사용자 가이드

AWS Glue

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Glue: 사용자 가이드

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 타사 제품 또는 서비스와 함께, 당사 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

AWS Glue란 무엇인가요? 1
AWS Glue 기능 2
AWS Glue의 혁신에 대해 알아보기 3
AWS Glue 시작하기 3
AWS Glue 액세스 4
관련 서비스 4
작동 방식 5
격리 중인 서버리스 ETL 작업 실행 6
개념 7
AWS Glue 용어 9
구성 요소 12
AWS Glue 콘솔 12
AWS Glue Data Catalog 12
AWS Glue 크롤러 및 분류자 13
AWS Glue ETL 연산 13
AWS Glue의 스트리밍 ETL 14
AWS Glue 작업 시스템 14
시각적 ETL 구성 요소 14
AWS Glue for Spark 및 AWS Glue for Ray 20
AWS Glue for Ray란 무엇인가요? 21
반구조화된 스키마를 관계형 스키마로 변환하기 21
AWS Glue 유형 23
AWS Glue 데이터 카탈로그 유형 23
Spark 스크립트가 포함된 AWS Glue의 유형 24
AWS Glue 크롤러 유형 24
시작하기 25
AWS Glue 사용 개요 25
IAM 권한 설정 27
다음 단계 32
AWS Glue Studio에 대한 설정 32
AWS Glue Studio에서 노트북 시작하기 45
사용 프로필 설정 47
사용 프로필 관리 49
사용 프로필 및 작업 61

AWS Glue Data Catalog 시작하기 61
개요 62
1단계: 데이터베이스 생성 62
단계 2. 테이블 생성 64
다음 단계 65
데이터 스토어에 대한 네트워크 액세스 설정 68
AWS Glue에서 PyPI에 연결하도록 VPC 설정 69
VPC에서 DNS 설정 71
암호화 설정 72
개발에 대한 네트워킹 설정 76
개발 엔드포인트에 대한 네트워크 설정 76
노트북 서버용 Amazon EC2 설정 78
데이터 검색 및 카탈로그 작성 80
데이터 카탈로그 채우기 82
AWS Glue 크롤러 사용 83
수동으로 메타데이터 정의 166
다른 AWS 서비스와 통합 184
데이터 카탈로그 설정 185
트랜잭션 테이블 채우기 및 관리 188
Iceberg 테이블 생성 188
Iceberg 테이블 최적화 192
Iceberg 테이블의 쿼리 성능 최적화 220
데이터 카탈로그 관리 222
스키마 업데이트 및 새 파티션 추가 223
열 통계를 사용한 쿼리 성능 최적화 231
데이터 카탈로그 암호화 254
Lake Formation을 사용한 데이터 카탈로그 보호 255
AWS Glue Data Catalog 뷰 작업 255
데이터 카탈로그 액세스 259
AWS Glue Iceberg REST 엔드포인트를 사용하여 Data Catalog에 연결 260
AWS Glue Iceberg REST 확장 엔드포인트를 사용하여 Data Catalog에 연결 263
Apache Iceberg용 AWS Glue REST API 264
독립 실행형 Spark 애플리케이션에서 Data Catalog에 연결 283
Amazon Redshift와 Apache Iceberg 간의 데이터 매핑 284
AWS Glue Iceberg REST 카탈로그 API 사용 시 고려 사항 및 제한 사항 285
데이터 카탈로그 모범 사례 286

Amazon CloudWatch에서 Data Catalog 사용량 지표 모니터링 287
Data Catalog 지표의 개요 287
CloudWatch 대시보드에 지표 추가 287
..... 289
AWS Glue 스키마 레지스트리 289
스키마 290
레지스트리 292
스키마 버전 관리 및 호환성 293
오픈 소스 Serde 라이브러리 297
Schema Registry의 할당량 298
작동 방식 298
시작하기 300
AWS Glue Schema Registry와 통합 322
AWS Glue Schema Registry로 마이그레이션 348
데이터에 연결 350
커넥터 및 연결 사용 개요 350
통합 연결 351
지원되는 인증 유형 354
고려 사항 354
사용 가능한 연결 355
제한 사항 358
AWS Glue 연결 속성 358
필수 연결 속성 359
JDBC 연결 속성 360
MongoDB 및 MongoDB Atlas 연결 속성 365
Salesforce 연결 속성 365
Snowflake 연결 366
Vertica 연결 367
SAP HANA 연결 368
Azure SQL 연결 369
Teradata Vantage 연결 369
OpenSearch Service 연결 370
Azure Cosmos 연결 371
SSL 연결 속성 372
인증을 위한 Kafka 연결 속성 374
Google BigQuery 연결 375

Vertica 연결 367
에 연결 보안 인증 정보 저장 AWS Secrets Manager 376
AWS Glue 연결 추가 377
Adobe Analytics에 연결 379
Adobe Marketo Engage에 연결 387
Amazon Redshift에 연결 395
Asana에 연결 400
Azure Cosmos DB에 연결 409
Azure SQL에 연결 411
Blackbaud Raiser's Edge NXT에 연결 414
CircleCI에 연결 423
Datadog에 연결 435
Docusign Monitor에 연결 442
Domo에 연결 449
Dynatrace에 연결 457
Facebook Ads에 연결 463
Facebook Page Insights에 연결 470
Freshdesk에 연결 491
Freshsales에 연결 501
Google Ads에 연결 507
Google Analytics에 연결 4 533
Google BigQuery에 연결 544
Google Search Console에 연결 548
Google Sheets에 연결 557
에 연결 HubSpot 563
Instagram Ads에 연결 588
Intercom에 연결 595
Jira Cloud에 연결 616
Kustomer에 연결 641
LinkedIn에 연결 668
Mailchimp에 연결 675
Microsoft Teams에 연결 685
Mixpanel에 연결 695
Monday에 연결 709
MongoDB에 연결 717
Oracle에 연결 NetSuite 721

OpenSearch Service에 연결 731
Okta에 연결 733
PayPal에 연결 741
Pendo에 연결 749
Pipedrive에 연결 756
Productboard에 연결 764
QuickBooks에 연결 773
Salesforce에 연결 780
Salesforce Marketing Cloud에 연결 791
Salesforce Commerce Cloud에 연결 815
Salesforce Marketing Cloud Account Engagement에 연결 825
SAP HANA에 연결 837
SAP OData에 연결 840
SendGrid에 연결 860
에 연결 ServiceNow 870
Slack에 연결 880
Smartsheet에 연결 890
Snapchat Ads에 연결 901
Snowflake에 연결 910
Stripe에 연결 914
Teradata에 연결 961
Twilio에 연결 964
Vertica에 연결 982
WooCommerce에 연결 986
Zendesk에 연결 1013
Zoho CRM에 연결 1040
Zoom Meetings에 연결 1049
자체 JDBC 드라이버를 사용하여 JDBC 연결 추가 1059
사용자 지정 커넥터 및 연결 사용 1064
사용자 정의 커넥터 생성 1066
커넥터에 대한 연결 생성 1070
사용자 정의 커넥터로 작업 작성 1078
커넥터 및 연결 관리 1085
사용자 정의 커넥터 개발 1088
AWS Glue Studio에서 커넥터와 연결을 사용하는 경우의 제한 사항 1090
AWS Glue 연결 테스트 1090

를 통과하도록 AWS 호출 구성 VPC 1091
의 JDBC 데이터 스토어에 연결 VPC 1092
탄력적 네트워크 인터페이스를 사용하여 VPC 데이터 액세스 1092
탄력적 네트워크 인터페이스 속성 1093
MongoDB 또는 MongoDB Atlas 연결 사용 1094
VPC 엔드포인트를 사용하여 Amazon S3 데이터 스토어 크롤링 1094
사전 조건 1095
Amazon S3에 대한 연결 생성 1096
Amazon S3에 대한 연결 테스트 1099
Amazon S3 데이터 스토어의 크롤러 생성 1101
Amazon S3 기반 데이터 카탈로그 테이블의 크롤러 생성 1103
크롤러 실행 1104
문제 해결 1104
AWS Lake Formation 보호 데이터 사용 1104
전체 테이블 액세스 1104
연결 문제 해결 1106
자습서: AWS Glue Connector for Elasticsearch 사용 1106
사전 조건 1107
1단계: (선택 사항) OpenSearch 클러스터 정보에 대한 AWS 보안 암호 생성 1107
2단계: 커넥터 구독 1108
3단계: AWS Glue Studio에서 커넥터 활성화 및 연결 생성 1109
4단계: ETL 작업에 대한 IAM 역할 구성 1110
5단계: OpenSearch 연결을 사용하는 작업 생성 1110
6단계: 작업 실행 1112
대화형 세션을 사용하여 AWS Glue 작업 구축 1113
AWS Glue 대화형 세션 개요 1113
제한 사항 1114
AWS Glue 대화형 세션 시작하기 1114
대화형 세션을 로컬에서 설정하기 위한 사전 조건 1114
Jupyter 및 AWS Glue 대화형 세션 Jupyter 커널 설치 1114
Jupyter 실행 1114
세션 자격 증명 및 리전 구성 1115
대화형 세션 평가판 업그레이드 1116
SageMaker AI Studio에서 대화형 세션 사용 1117
Microsoft Visual Studio Code에서 대화형 세션 사용 1117
IAM을 이용한 대화형 세션 1120

Jupyter 및 AWS Glue Studio 노트북용 AWS Glue 대화형 세션 구성 1128
Jupyter 매직 소개 1128
Jupyter용 AWS Glue 대화형 세션에서 지원되는 매직 1128
세션 이름 지정 1145
대화형 세션에 대한 IAM 역할 지정 1146
명명된 프로파일을 사용하여 세션 구성 1146
스크립트 또는 노트북을 AWS Glue 작업으로 변환 1147
AWS Glue 대화형 세션에서 스트리밍 작업 수행 1148
스트리밍 세션 유형 전환 1148
대화형 개발을 위한 샘플링 입력 스트림 1148
대화형 세션에서 스트리밍 애플리케이션 실행 1149
로컬로 개발 및 테스트 1150
AWS Glue Studio를 사용한 개발 1151
대화형 세션을 사용하여 개발 1151
Docker를 사용하여 로컬에서 AWS Glue 작업 개발 1151
dev 엔드포인트 1161
개발 엔드포인트에서 대화형 세션으로 마이그레이션 1162
개발 엔드포인트를 사용하여 스크립트 개발 1164
노트북 관리 1191
AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 1194
콘솔로 로그인합니다 1194
AWS Glue Studio에서 작업을 생성하기 위한 다음 단계 1194
AWS Glue Studio에서 시각적 ETL 작업 시작 1195
AWS Glue Studio에서 작업 시작 1195
작업 편집기 기능 1197
AWS Glue 관리형 변환으로 데이터 변환 1205
사용자 지정 시각적 변환으로 데이터 변환 1275
AWS Glue Studio에서 데이터 레이크 프레임워크 사용 1292
데이터 원본에 연결 1303
데이터 대상 노드 구성 1311
작업 스크립트 편집 또는 업로드 1317
작업 다이어그램에서 노드의 상위 노드 변경 1321
작업 다이어그램에서 노드 삭제 1322
AWS Glue 데이터 카탈로그 노드에 소스 및 대상 파라미터 추가 1326
AWS Glue에서 Git 버전 제어 시스템 사용 1328
AWS Glue Studio 노트북을 사용한 코드 작성 1336

제한 사항 1336
노트북 사용 개요 1337
AWS Glue Studio에서 노트북을 사용하여 ETL 작업 생성 1338
노트북 편집기 구성 요소 1339
노트북 및 작업 스크립트 저장 1340
노트북 세션 관리 1341
Amazon Q Developer를 AWS Glue Studio 노트북과 함께 사용하기 1342
작업 실행 모니터링 1343
작업 모니터링 대시보드에 액세스 1343
작업 모니터링 대시보드 개요 1343
작업 실행 보기 1343
작업 실행 로그 보기 1347
작업 실행의 세부 정보 보기 1348
Spark 작업 실행에 대한 Amazon CloudWatch 지표 보기 1351
Ray 작업 실행에 대한 Amazon CloudWatch 지표 보기 1351
민감한 데이터 감지 및 처리 1353
데이터를 스캔하는 방법 선택 1354
탐지할 PII 엔터티 선택 1355
탐지 민감도 수준 지정 1359
식별된 PII 데이터로 수행할 작업 선택 1359
세분화된 작업 오버라이드 추가 1360
작업 관리 1361
작업 실행 시작 1362
작업 실행 예약 1362
작업 일정 관리 1363
작업 실행 중지 1364
작업 보기 1364
최근 작업 실행에 대한 정보 보기 1365
작업 스크립트 보기 1366
작업 속성 수정 1366
작업 저장 1369
작업 복제 1371
작업 삭제 1371
작업 사용 1373
AWS Glue 버전 1374
AWS Glue 버전 1374

AWS Glue 버전 지원 정책 1390
AWS Glue 버전 5.0으로 AWS Glue for Spark 작업 마이그레이션 1392
AWS Glue 버전 4.0으로 AWS Glue for Spark 작업 마이그레이션 1406
AI를 사용하여 업그레이드 분석 1420
Spark 작업 사용 1436
작업 파라미터 1437
Spark 및 PySpark 작업 1446
스트리밍 ETL 작업 1581
FindMatches로 레코드 매칭 1595
Spark 프로그램 마이그레이션 1627
Ray 작업 사용 1634
AWS Glue for Ray 시작하기 1634
지원되는 Ray 런타임 환경 1635
Ray 작업에서 작업자 고려 1636
Ray 작업 파라미터 1637
Ray 작업 지표 1640
Python 쉘 작업 속성 구성 1641
제한 사항 1641
Python 셸 작업에 대한 작업 속성 정의 1642
Python 셸 작업에 지원되는 라이브러리 1644
자체 Python 라이브러리 제공 1646
AWS Glue에서 Python 쉘 작업과 함께 AWS CloudFormation 사용 1649
Python 쉘 3.6에서 Python 쉘 3.9로 마이그레이션 1650
AWS Glue Python 쉘 작업에서 마이그레이션 1650
모니터링 1652
AWS 태그 1653
EventBridge를 사용한 자동화 1658
리소스 모니터링 1660
CloudTrail을 사용하여 로깅 1663
작업 실행 상태 1667
AWS Glue 스트리밍 1670
스트리밍 사용 사례 1670
AWS Glue 스트리밍 사용의 이점은 무엇인가요? 1671
AWS Glue 스트리밍은 언제 사용하나요? 1671
지원되는 데이터 원본 1672
지원되는 데이터 대상 1673

자습서: AWS Glue Studio를 사용하여 첫 번째 스트리밍 워크로드 구축 1673
사전 조건 1673
Amazon Kinesis의 스트리밍 데이터 소비 1673
자습서: AWS Glue Studio 노트북을 사용하여 첫 번째 스트리밍 워크로드 구축 1684
사전 조건 1685
Amazon Kinesis의 스트리밍 데이터 소비 1685
스트리밍 개념 1692
AWS Glue 스트리밍 작업의 구조 1692
스트리밍 연결 1696
Kafka 연결 1696
Kinesis 연결 1702
AWS Glue 스트리밍 자동 크기 조정 1709
AWS Glue Studio에서 Auto Scaling 사용 1709
AWS CLI 또는 SDK를 사용하여 Auto Scaling 사용 설정 1710
작동 방식 1711
유지 관리 기간 1713
유지 관리 기간 설정 1713
유지 관리 기간 동작 1714
작업 모니터링 1715
데이터 손실 처리 1717
고급 AWS Glue 스트리밍 개념 1718
스트림 처리 시 시간 고려 사항 1718
기간 설정 1718
최신 데이터 및 워터마크 처리 1724
AWS Glue 스트리밍 작업 모니터링 1725
AWS Glue 스트리밍 지표 시각화 1726
AWS Glue 스트리밍 지표 사용 1727
최고의 성능을 얻는 방법 1732
제로 ETL 통합 1734
AWS Glue의 제로 ETL 기능 1734
사전 조건 1735
소스 리소스 설정 1735
대상 리소스 설정 1737
Amazon Redshift 데이터 웨어하우스 생성 1742
VPC 설정 1742
교차 계정 통합 설정 1745

소스 구성 1746
특수 SAP 엔터티 지원 1746
Amazon DynamoDB 소스 1746
Salesforce 소스 1748
Salesforce MCAE 소스 1748
추가 Salesforce 구성 1749
SAP OData 소스 1749
ServiceNow 소스 1749
Zendesk 소스 1750
Zoho CRM 소스 1750
Facebook Ads 소스 1751
Instagram Ads 소스 1752
지원되지 않는 Salesforce 필드 1752
지원되지 않는 ServiceNow 필드 1755
대상 구성 1760
S3 스토리지가 있는 SageMaker Lakehouse 1761
S3 Tables 대상 1761
Amazon Redshift 스토리지가 있는 SageMaker Lakehouse 1765
Amazon Redshift 데이터 웨어하우스 대상 1766
통합 대상 구성 1768
파티션 및 스키마 중첩 해제 1769
기본 중첩 해제 및 파티셔닝 동작 1769
스키마 중첩 해제 1770
제로 ETL 통합을 위한 데이터 파티셔닝 1773
제한 사항 1783
일반적인 작업 1783
통합 생성 1783
통합 수정 1785
통합 삭제 1785
API 사용 1786
통합 모니터링 1786
통합 상태 1786
로그 보기 1787
지표 보기 1789
이벤트 알림 1790
제한 사항 1791

AWS Glue Data Quality 1801
이점 및 주요 특징 1801
작동 방법 1802
AWS Glue Data Catalog에 대한 데이터 품질 1802
AWS Glue ETL 작업에 대한 데이터 품질 1802
AWS Glue Data Quality 진입점 비교 1803
고려 사항 1804
용어 1805
Limits 1806
AWS Glue Data Quality의 릴리즈 정보 1806
정식 출시: 새 기능 1806
2023년 11월 27일(미리 보기) 1806
2024년 3월 12일 1807
2024년 6월 26일 1807
2024년 8월 7일 1807
2024년 11월 22일 1807
2024년 12월 6일 1807
2025년 6월 30일 1808
AWS Glue Data Quality의 이상 탐지 1808
작동 방식 1809
이상 탐지 알고리즘의 세부 정보 1813
AWS Glue Data Quality에 대한 IAM 권한 1814
IAM 권한 1814
평가 실행을 예약하는 데 필요한 IAM 설정 1817
예제 IAM 정책 1818
Data Catalog에서 AWS Glue Data Quality 시작하기 1823
사전 조건 1823
단계별 예제 1824
규칙 권장 사항 생성 1824
모니터링 규칙 권장 사항 1825
권장 규칙 세트 편집 1826
새 규칙 세트 생성 1827
규칙 세트를 실행하여 데이터 품질 평가 1828
데이터 품질 점수 및 결과 보기 1829
관련 주제 1830
AWS Glue Studio에서 데이터 품질 평가 1830

이점 1831
AWS Glue Studio에서 ETL 작업에 대한 데이터 품질 평가 1831
데이터 품질 규칙 작성기 1841
AWS Glue ETL 작업에서 이상 탐지 구성 1846
데이터 품질 점수 및 이상 보기 1849
AWS Glue Studio 노트북에서 ETL 작업에 대한 데이터 품질 1852
사전 조건 1853
AWS Glue Studio에서 ETL 작업 생성 1853
데이터 품질 정의 언어(DQDL) 참조 1858
구문 1860
규칙 유형 참조 1875
API를 사용하여 데이터 품질 측정 및 관리 1930
사전 조건 1931
AWS Glue Data Quality 권장 사용 1931
AWS Glue Data Quality 규칙 세트 사용 1934
AWS Glue Data Quality 실행 사용 1936
AWS Glue Data Quality 결과 사용 1941
알림, 배포 및 예약 설정 1947
Amazon EventBridge 통합에서 알림 설정 1948
CloudWatch 통합에서 경보 및 알림 설정 1955
데이터 품질 결과 쿼리 1957
데이터 품질 규칙 배포 1960
데이터 품질 규칙 예약 1961
AWS Glue Data Quality의 저장 데이터 암호화 1961
AWS 소유 키 1961
고객 관리형 키 1961
고객 관리형 키 생성 1963
보안 구성 생성 1966
AWS Glue Data Quality 암호화 컨텍스트 1967
AWS Glue Data Quality의 암호화 키 모니터링 1967
자세히 알아보기 1972
AWS Glue Data Quality 오류 해결 1972
오류: 누락된 모듈 1973
오류: 권한 부족 1973
오류: 규칙 세트가 고유하지 않음 1973
오류: 특수 문자가 있는 테이블 1973

오류: 큰 규칙 세트로 인한 오버플로 1973
오류: 규칙 상태가 실패임 1974
AnalysisException: 기본 데이터베이스의 존재를 확인할 수 없음 1974
제공된 키 맵이 해당 데이터 프레임에 적합하지 않음 1974
java.lang.RuntimeException: 데이터를 가져오지 못했습니다. 1975
시작 오류: 버킷에 대해 S3에서 다운로드하는 중 오류 발생 1975
InvalidInputException(상태: 400): 데이터 품질 규칙을 구문 분석할 수 없음 1975
오류: Eventbridge에서 사용자가 설정한 일정에 따라 Glue DQ 작업을 트리거하지 않습니다. 1976
CustomSQL 오류 1976
동적 규칙 1977
사용자 클래스의 예외: org.apache.spark.sql.AnalysisException: org.apache.hadoop.hive.ql.metadata.HiveException 1979
UNCLASSIFIED_ERROR; IllegalArgumentException: Parsing Error: No rules or analyzers provided., no viable alternative at input 1979
AWS Glue의 Amazon Q 데이터 통합 1980
Amazon Q란 무엇인가요? 1980
AWS Glue의 Amazon Q 데이터 통합 1980
Amazon Q 데이터 통합을 사용하여 작업 1981
모범 사례 1982
서비스 개선 1983
고려 사항 1983
Amazon Q 데이터 통합 설정 1983
IAM 권한 구성 1984
지원되는 코드 생성 1986
상호 작용 예제 1987
Amazon Q 채팅 상호 작용 1987
AWS Glue Studio 노트북 상호 작용 1988
컨텍스트 인식 사용 1991
예: 상호 작용 1992
제한 사항 1996
오케스트레이션 1997
트리거를 사용하여 작업 및 크롤러 시작 1997
AWS Glue 트리거 1997
트리거 추가 2000
트리거 활성화 및 비활성화 2004

블루프린트 및 워크플로를 사용하여 복잡한 ETL 활동 수행 2004
워크플로우 개요 2005
워크플로 수동 생성 및 구축 2009
EventBridge 이벤트로 워크플로 시작 2013
워크플로를 시작한 EventBridge 이벤트 보기 2020
워크플로 실행 및 모니터링 2021
워크플로 실행 중지 2024
워크플로 실행 복구 및 재개 2025
워크플로 실행 속성 가져오기 및 설정 2031
AWS Glue API를 사용하여 워크플로 쿼리 2032
블루프린트 및 워크플로 제한 사항 2036
블루프린트 오류 해결 2037
블루프린트 페르소나 및 역할에 대한 권한 2042
블루프린트 개발 2046
블루프린트 개요 2047
블루프린트 개발 2050
블루프린트 등록 2074
블루프린트 보기 2076
블루프린트 업데이트 2078
블루프린트에서 워크플로 생성 2080
블루프린트 실행 보기 2082
AWS Glue용 AWS CloudFormation 2084
샘플 데이터베이스 2086
샘플 데이터베이스, 테이블 및 파티션 2087
샘플 Grok 분류자 2091
샘플 JSON 분류자 2092
샘플 XML 분류자 2093
샘플 Amazon S3 크롤러 2094
샘플 연결 2096
샘플 JDBC 크롤러 2097
Amazon S3에서 Amazon S3로의 샘플 작업 2100
JDBC에서 Amazon S3로의 샘플 작업 2101
샘플 온디맨드 트리거 2103
일정이 정해진 샘플 트리거 2104
샘플 조건부 트리거 2105
샘플 기계 학습 변환 2106

샘플 데이터 품질 규칙 세트 2108
EventBridge 스케줄러에서 샘플 데이터 품질 규칙 세트 2109
샘플 개발 엔드포인트 2111
AWS Glue 프로그래밍 안내서 2113
사용자 지정 스크립트 제공 2113
AWS Glue for Spark 2114
자습서: Spark 스크립트 작성 2114
PySpark에서 ETL 2127
Scala에서 ETL 2353
기능 및 최적화 2436
AWS Glue for Ray 2685
자습서: Ray 스크립트 작성 2686
AWS Glue for Ray에서 Ray Core 및 Ray Data 사용 2691
파일 및 Python 라이브러리 제공 2693
데이터에 연결 2698
AWS SDK 작업 2700
AWS Glue API 2702
보안 2729
- 데이터 유형 - 2729
DataCatalogEncryptionSettings 2730
EncryptionAtRest 2730
ConnectionPasswordEncryption 2730
EncryptionConfiguration 2731
S3Encryption 2732
CloudWatchEncryption 2732
JobBookmarksEncryption 2732
SecurityConfiguration 2733
GluePolicy 2733
DataQualityEncryption 2734
- 작업 - 2734
GetDataCatalogEncryptionSettings (get_data_catalog_encryption_settings) 2735
PutDataCatalogEncryptionSettings (put_data_catalog_encryption_settings) 2735
PutResourcePolicy(put_resource_policy) 2736
GetResourcePolicy(get_resource_policy) 2737
DeleteResourcePolicy(delete_resource_policy) 2738
CreateSecurityConfiguration (create_security_configuration) 2739

DeleteSecurityConfiguration (delete_security_configuration) 2740
GetSecurityConfiguration (get_security_configuration) 2741
GetSecurityConfigurations (get_security_configurations) 2741
GetResourcePolicies(get_resource_policies) 2742
카탈로그 객체 2743
카탈로그 2743
데이터베이스 수 2755
표 2765
파티션 2811
연결 2836
사용자 정의 함수 2877
Athena 카탈로그 가져오기 2884
테이블 옵티마이저 2886
- 데이터 유형 - 2886
TableOptimizer 2887
TableOptimizerConfiguration 2887
TableOptimizerVpcConfiguration 2888
CompactionConfiguration 2888
IcebergCompactionConfiguration 2889
TableOptimizerRun 2889
BatchGetTableOptimizerEntry 2891
BatchTableOptimizer 2891
BatchGetTableOptimizerError 2892
RetentionConfiguration 2892
IcebergRetentionConfiguration 2893
OrphanFileDeletionConfiguration 2893
IcebergOrphanFileDeletionConfiguration 2893
CompactionMetrics 2894
RetentionMetrics 2894
OrphanFileDeletionMetrics 2894
IcebergCompactionMetrics 2894
IcebergRetentionMetrics 2895
IcebergOrphanFileDeletionMetrics 2895
RunMetrics 2896
- 작업 - 2896
GetTableOptimizer (get_table_optimizer) 2897

BatchGetTableOptimizer (batch_get_table_optimizer) 2898
ListTableOptimizerRuns (list_table_optimizer_runs) 2899
CreateTableOptimizer (create_table_optimizer) 2900
DeleteTableOptimizer (delete_table_optimizer) 2901
UpdateTableOptimizer (update_table_optimizer) 2902
크롤러 및 분류자 2903
분류자 2904
크롤러 2918
열 통계값 2944
스케줄러 2960
ETL 스크립트 자동 생성 2963
- 데이터 유형 - 2963
CodeGenNode 2963
CodeGenNodeArg 2964
CodeGenEdge 2964
위치 2965
CatalogEntry 2965
MappingEntry 2965
- 작업 - 2966
CreateScript(create_script) 2966
GetDataflowGraph(get_dataflow_graph) 2967
GetMapping(get_mapping) 2968
GetPlan(get_plan) 2968
시각적 작업 API 2970
- 데이터 유형 - 2970
CodeGenConfigurationNode 2974
JDBCConnectorOptions 2980
StreamingDataPreviewOptions 2982
AthenaConnectorSource 2982
JDBCConnectorSource 2983
SparkConnectorSource 2984
CatalogSource 2984
MySQLCatalogSource 2985
PostgreSQLCatalogSource 2985
OracleSQLCatalogSource 2986
MicrosoftSQLServerCatalogSource 2986

CatalogKinesisSource 2986
DirectKinesisSource 2987
KinesisStreamingSourceOptions 2988
CatalogKafkaSource 2990
DirectKafkaSource 2991
KafkaStreamingSourceOptions 2992
RedshiftSource 2994
AmazonRedshiftSource 2994
AmazonRedshiftNodeData 2995
AmazonRedshiftAdvancedOption 2997
옵션 2997
S3CatalogSource 2998
S3SourceAdditionalOptions 2998
S3CsvSource 2999
DirectJDBCSource 3001
S3DirectSourceAdditionalOptions 3002
S3JsonSource 3002
S3ParquetSource 3004
S3DeltaSource 3005
S3CatalogDeltaSource 3006
CatalogDeltaSource 3006
S3HudiSource 3007
S3CatalogHudiSource 3008
S3ExcelSource 3008
CatalogHudiSource 3010
DynamoDBCatalogSource 3010
RelationalCatalogSource 3011
JDBCConnectorTarget 3011
SparkConnectorTarget 3012
BasicCatalogTarget 3013
MySQLCatalogTarget 3013
PostgreSQLCatalogTarget 3014
OracleSQLCatalogTarget 3014
MicrosoftSQLServerCatalogTarget 3015
RedshiftTarget 3015
AmazonRedshiftTarget 3016

UpsertRedshiftTargetOptions 3017
S3CatalogTarget 3017
S3GlueParquetTarget 3018
CatalogSchemaChangePolicy 3018
S3DirectTarget 3019
S3HudiCatalogTarget 3020
S3HudiDirectTarget 3020
S3DeltaCatalogTarget 3021
S3DeltaDirectTarget 3022
S3HyperDirectTarget 3023
S3IcebergDirectTarget 3024
DirectSchemaChangePolicy 3025
ApplyMapping 3026
매핑 3026
SelectFields 3027
DropFields 3027
RenameField 3028
Spigot 3028
조인 3029
JoinColumn 3029
SplitFields 3030
SelectFromCollection 3030
FillMissingValues 3031
Filter 3031
FilterExpression 3032
FilterValue 3032
CustomCode 3032
SparkSQL 3033
SqlAlias 3034
DropNullFields 3034
NullCheckBoxList 3035
NullValueField 3035
데이터 형식 3036
병합 3036
Union 3036
PIIDetection 3037

Aggregate 3038
DropDuplicates 3038
GovernedCatalogTarget 3039
GovernedCatalogSource 3040
AggregateOperation 3040
GlueSchema 3041
GlueStudioSchemaColumn 3041
GlueStudioColumn 3041
DynamicTransform 3042
TransformConfigParameter 3043
EvaluateDataQuality 3044
DQResultsPublishingOptions 3044
DQStopJobOnFailureOptions 3045
EvaluateDataQualityMultiFrame 3045
레시피 3046
RecipeReference 3046
SnowflakeNodeData 3047
SnowflakeSource 3049
SnowflakeTarget 3049
ConnectorDataSource 3050
ConnectorDataTarget 3051
RecipeStep 3052
RecipeAction 3052
ConditionExpression 3052
작업 3053
작업 3053
작업 실행 3078
트리거 3096
통합 API 3110
- 데이터 유형 - 3110
통합 3110
IntegrationConfig 3112
IntegrationPartition 3112
IntegrationError 3113
IntegrationFilter 3114
InboundIntegration 3114

SourceProcessingProperties 3115
TargetProcessingProperties 3115
SourceTableConfig 3116
TargetTableConfig 3116
- 작업 - 3117
CreateIntegration(create_integration) 3117
ModifyIntegration(modify_integration) 3120
DescribeIntegrations(describe_integrations) 3123
DeleteIntegration(delete_integration) 3124
DescribeInboundIntegrations(describe_inbound_integrations) 3126
CreateIntegrationTableProperties(create_integration_table_properties) 3127
UpdateIntegrationTableProperties(update_integration_table_properties) 3128
GetIntegrationTableProperties(get_integration_table_properties) 3130
DeleteIntegrationTableProperties(delete_integration_table_properties) 3131
CreateIntegrationResourceProperty(create_integration_resource_property) 3132
UpdateIntegrationResourceProperty(update_integration_resource_property) 3133
GetIntegrationResourceProperty(get_integration_resource_property) 3134
UntagResource(untag_resource) 3135
ListTagsForResource(list_tags_for_resource) 3136
— 예외 — 3136
ResourceNotFoundException 3136
InternalServerException 3137
IntegrationAlreadyExistsFault 3137
IntegrationConflictOperationFault 3137
IntegrationQuotaExceededFault 3137
KMSKeyNotAccessibleFault 3138
IntegrationNotFoundFault 3138
TargetResourceNotFound 3138
InvalidIntegrationStateFault 3138
대화형 세션 3139
- 데이터 유형 - 3139
세션 3139
SessionCommand 3141
문 3142
StatementOutput 3143
StatementOutputData 3143

ConnectionsList 3143
- 작업 - 3144
CreateSession(create_session) 3144
StopSession(stop_session) 3147
DeleteSession(delete_session) 3148
GetSession(get_session) 3149
ListSessions(list_sessions) 3150
RunStatement(run_statement) 3151
CancelStatement(cancel_statement) 3152
GetStatement(get_statement) 3153
ListStatements(list_statements) 3154
DevEndpoints 3155
- 데이터 유형 - 3155
DevEndpoint 3155
DevEndpointCustomLibraries 3159
- 작업 - 3159
CreateDevEndpoint(create_dev_endpoint) 3160
UpdateDevEndpoint(update_dev_endpoint) 3165
DeleteDevEndpoint(delete_dev_endpoint) 3167
GetDevEndpoint(get_dev_endpoint) 3167
GetDevEndpoints(get_dev_endpoints) 3168
BatchGetDevEndpoints(batch_get_dev_endpoints) 3169
ListDevEndpoints(list_dev_endpoints) 3170
Schema Registry 3171
- 데이터 유형 - 3171
RegistryId 3171
RegistryListItem 3172
MetadataInfo 3173
OtherMetadataValueListItem 3173
SchemaListItem 3173
SchemaVersionListItem 3174
MetadataKeyValuePair 3175
SchemaVersionErrorItem 3175
ErrorDetails 3176
SchemaVersionNumber 3176
SchemaId 3176

- 작업 - 3177
CreateRegistry(create_registry) 3177
CreateSchema(create_schema) 3179
GetSchema(get_schema) 3183
ListSchemaVersions(list_schema_versions) 3185
GetSchemaVersion(get_schema_version) 3186
GetSchemaVersionsDiff(get_schema_versions_diff) 3187
ListRegistries(list_registries) 3188
ListSchemas(list_schemas) 3189
RegisterSchemaVersion(register_schema_version) 3190
UpdateSchema(update_schema) 3191
CheckSchemaVersionValidity(check_schema_version_validity) 3193
UpdateRegistry(update_registry) 3194
GetSchemaByDefinition(get_schema_by_definition) 3195
GetRegistry(get_registry) 3196
PutSchemaVersionMetadata(put_schema_version_metadata) 3197
QuerySchemaVersionMetadata(query_schema_version_metadata) 3199
RemoveSchemaVersionMetadata(remove_schema_version_metadata) 3200
DeleteRegistry(delete_registry) 3202
DeleteSchema(delete_schema) 3203
DeleteSchemaVersions(delete_schema_versions) 3204
워크플로 3205
- 데이터 유형 - 3205
JobNodeDetails 3205
CrawlerNodeDetails 3206
TriggerNodeDetails 3206
Crawl 3206
노드 3207
Edge 3207
워크플로 3208
WorkflowGraph 3209
WorkflowRun 3209
WorkflowRunStatistics 3211
StartingEventBatchCondition 3212
블루프린트 3212
BlueprintDetails 3213

LastActiveDefinition 3213
BlueprintRun 3214
- 작업 - 3215
CreateWorkflow(create_workflow) 3216
UpdateWorkflow(update_workflow) 3217
DeleteWorkflow(delete_workflow) 3219
GetWorkflow(get_workflow) 3219
ListWorkflows(list_workflows) 3220
BatchGetWorkflows(batch_get_workflows) 3221
GetWorkflowRun(get_workflow_run) 3222
GetWorkflowRuns(get_workflow_runs) 3222
GetWorkflowRunProperties(get_workflow_run_properties) 3223
PutWorkflowRunProperties(put_workflow_run_properties) 3224
CreateBlueprint(create_blueprint) 3225
UpdateBlueprint(update_blueprint) 3227
DeleteBlueprint(delete_blueprint) 3227
ListBlueprints(list_blueprints) 3228
BatchGetBlueprints(batch_get_blueprints) 3229
StartBlueprintRun(start_blueprint_run) 3230
GetBlueprintRun(get_blueprint_run) 3231
GetBlueprintRuns(get_blueprint_runs) 3231
StartWorkflowRun(start_workflow_run) 3232
StopWorkflowRun(stop_workflow_run) 3233
ResumeWorkflowRun(resume_workflow_run) 3234
사용 프로필 3235
- 데이터 유형 - 3235
ProfileConfiguration 3235
ConfigurationObject 3236
UsageProfileDefinition 3237
- 작업 - 3237
CreateUsageProfile(create_usage_profile) 3237
GetUsageProfile(get_usage_profile) 3238
UpdateUsageProfile(update_usage_profile) 3239
DeleteUsageProfile(delete_usage_profile) 3240
ListUsageProfiles(list_usage_profiles) 3241
기계 학습 3242

- 데이터 유형 - 3242
TransformParameters 3243
EvaluationMetrics 3243
MLTransform 3243
FindMatchesParameters 3246
FindMatchesMetrics 3247
ConfusionMatrix 3249
GlueTable 3249
TaskRun 3250
TransformFilterCriteria 3251
TransformSortCriteria 3252
TaskRunFilterCriteria 3252
TaskRunSortCriteria 3253
TaskRunProperties 3253
FindMatchesTaskRunProperties 3254
ImportLabelsTaskRunProperties 3254
ExportLabelsTaskRunProperties 3255
LabelingSetGenerationTaskRunProperties 3255
SchemaColumn 3255
TransformEncryption 3256
MLUserDataEncryption 3256
ColumnImportance 3256
- 작업 - 3257
CreateMLTransform(create_ml_transform) 3257
UpdateMLTransform(update_ml_transform) 3261
DeleteMLTransform(delete_ml_transform) 3263
GetMLTransform(get_ml_transform) 3264
GetMLTransforms(get_ml_transforms) 3267
ListMLTransforms(list_ml_transforms) 3268
StartMLEvaluationTaskRun(start_ml_evaluation_task_run) 3269
StartMLLabelingSetGenerationTaskRun(start_ml_labeling_set_generation_task_run) 3270
GetMLTaskRun(get_ml_task_run) 3271
GetMLTaskRuns(get_ml_task_runs) 3272
CancelMLTaskRun(cancel_ml_task_run) 3274
StartExportLabelsTaskRun(start_export_labels_task_run) 3275
StartImportLabelsTaskRun(start_import_labels_task_run) 3276

데이터 품질 3277
- 데이터 유형 - 3277
DataSource 3278
DataQualityRulesetListDetails 3278
DataQualityTargetTable 3279
DataQualityRulesetEvaluationRunDescription 3279
DataQualityRulesetEvaluationRunFilter 3280
DataQualityEvaluationRunAdditionalRunOptions 3280
DataQualityRuleRecommendationRunDescription 3281
DataQualityRuleRecommendationRunFilter 3281
DataQualityResult 3282
DataQualityAnalyzerResult 3283
DataQualityObservation 3284
MetricBasedObservation 3284
DataQualityMetricValues 3285
DataQualityRuleResult 3285
DataQualityResultDescription 3286
DataQualityResultFilterCriteria 3287
DataQualityRulesetFilterCriteria 3287
DataQualityAggregatedMetrics 3288
StatisticAnnotation 3289
TimestampedInclusionAnnotation 3289
AnnotationError 3290
DatapointInclusionAnnotation 3290
StatisticSummaryList 3291
StatisticSummary 3291
RunIdentifier 3292
StatisticModelResult 3292
- 작업 - 3293
StartDataQualityRulesetEvaluationRun(start_data_quality_ruleset_evaluation_run) 3294
CancelDataQualityRulesetEvaluationRun(cancel_data_quality_ruleset_evaluation_run) 3296
GetDataQualityRulesetEvaluationRun(get_data_quality_ruleset_evaluation_run) 3296
ListDataQualityRulesetEvaluationRuns(list_data_quality_ruleset_evaluation_runs) 3298
StartDataQualityRuleRecommendationRun(start_data_quality_rule_recommendation_run) 3299
CancelDataQualityRuleRecommendationRun(cancel_data_quality_rule_recommendation_run) 3301
GetDataQualityRuleRecommendationRun(get_data_quality_rule_recommendation_run) ... 3301

ListDataQualityRuleRecommendationRuns(list_data_quality_rule_recommendation_runs) . 3303
GetDataQualityResult(get_data_quality_result) 3304
BatchGetDataQualityResult(batch_get_data_quality_result) 3306
ListDataQualityResults(list_data_quality_results) 3307
CreateDataQualityRuleset(create_data_quality_ruleset) 3307
DeleteDataQualityRuleset(delete_data_quality_ruleset) 3309
GetDataQualityRuleset(get_data_quality_ruleset) 3310
ListDataQualityRulesets(list_data_quality_rulesets) 3311
UpdateDataQualityRuleset(update_data_quality_ruleset) 3312
ListDataQualityStatistics(list_data_quality_statistics) 3313
TimestampFilter 3314
CreateDataQualityRulesetRequest 3314
GetDataQualityRulesetResponse 3316
GetDataQualityResultResponse 3316
StartDataQualityRuleRecommendationRunRequest 3318
GetDataQualityRuleRecommendationRunResponse 3319
BatchPutDataQualityStatisticAnnotation(batch_put_data_quality_statistic_annotation) 3320
GetDataQualityModel(get_data_quality_model) 3321
GetDataQualityModelResult(get_data_quality_model_result) 3322
ListDataQualityStatisticAnnotations(list_data_quality_statistic_annotations) 3323
PutDataQualityProfileAnnotation(put_data_quality_profile_annotation) 3324
민감한 데이터 3325
- 데이터 유형 - 3325
CustomEntityType 3325
- 작업 - 3325
CreateCustomEntityType(create_custom_entity_type) 3326
DeleteCustomEntityType(delete_custom_entity_type) 3327
GetCustomEntityType(get_custom_entity_type) 3328
BatchGetCustomEntityTypes(batch_get_custom_entity_types) 3329
ListCustomEntityTypes(list_custom_entity_types) 3329
API 태그 지정 3330
- 데이터 유형 - 3330
태그 3330
- 작업 - 3331
TagResource(tag_resource) 3331
UntagResource(untag_resource) 3332

GetTags(get_tags) 3332
공통 데이터 형식 3333
태그 3333
DecimalNumber 3334
ErrorDetail 3334
PropertyPredicate 3334
ResourceUri 3335
ColumnStatistics 3335
ColumnStatisticsError 3336
ColumnError 3336
ColumnStatisticsData 3336
BooleanColumnStatisticsData 3337
DateColumnStatisticsData 3338
DecimalColumnStatisticsData 3338
DoubleColumnStatisticsData 3339
LongColumnStatisticsData 3339
StringColumnStatisticsData 3340
BinaryColumnStatisticsData 3340
문자열 패턴 3341
예외 3343
AccessDeniedException 3343
AlreadyExistsException 3344
ConcurrentModificationException 3344
ConcurrentRunsExceededException 3344
CrawlerNotRunningException 3344
CrawlerRunningException 3345
CrawlerStoppingException 3345
EntityNotFoundException 3345
FederationSourceException 3345
FederationSourceRetryableException 3346
GlueEncryptionException 3346
IdempotentParameterMismatchException 3346
IllegalWorkflowStateException 3347
InternalServiceException 3347
InvalidExecutionEngineException 3347
InvalidInputException 3347

InvalidStateException 3348
InvalidTaskStatusTransitionException 3348
JobDefinitionErrorException 3348
JobRunInTerminalStateException 3348
JobRunInvalidStateTransitionException 3349
JobRunNotInTerminalStateException 3349
LateRunnerException 3350
NoScheduleException 3350
OperationTimeoutException 3350
ResourceNotReadyException 3350
ResourceNumberLimitExceededException 3351
SchedulerNotRunningException 3351
SchedulerRunningException 3351
SchedulerTransitioningException 3351
UnrecognizedRunnerException 3352
ValidationException 3352
VersionMismatchException 3352
AWS Glue API 코드 예제 3353
기본 사항 3363
AWS Glue 시작 3364
기본 사항 알아보기 3373
작업 3509
보안 3648
데이터 보호 3648
저장 데이터 암호화 3649
전송 중 데이터 암호화 3666
FIPS 규정 준수 3667
키 관리 3667
기타 AWS 서비스에서 AWS Glue 종속성 3667
개발 엔드포인트 3668
자격 증명 및 액세스 관리 3668
대상 3669
ID를 통한 인증 3670
정책을 사용하여 액세스 관리 3673
AWS Glue의 작동 방식 IAM 3675
AWS Glue에 대한 IAM 권한 구성 3682

AWS Glue 액세스 제어 정책 예제 3713
AWS 관리형 정책 부여 3738
AWS Glue 리소스 ARN 지정 3745
교차 계정 액세스 권한 부여 3754
문제 해결 3761
AWS Lake Formation 액세스 제어 모델 3763
전체 테이블 액세스를 위해 AWS Lake Formation에서 AWS Glue 사용 3763
FGAC를 위한 Lake Formation 3775
AWS Glue를 통해 Amazon S3 Access Grants 사용 3786
AWS Glue가 S3 Access Grants에서 작동하는 방법 3787
AWS Glue에서의 S3 Access Grants 고려 사항 3787
AWS Glue에서 S3 Access Grants 설정 3788
로깅 및 모니터링 3790
규정 준수 확인 3791
복원성 3792
인프라 보안 3792
AWS Glue에 대한 인터페이스 VPC 엔드포인트(AWS PrivateLink) 구성 3793
공유 Amazon VPC 구성 3795
AWS Glue 문제 해결 3796
AWS Glue 문제 해결 정보 모으기 3796
Spark 오류 해결 3797
오류: 사용 가능하지 않는 리소스 3798
오류: VPC에서 서브넷 ID용 S3 엔드포인트 또는 NAT 게이트웨이를 찾을 수 없습니다. 3798
오류: 보안 그룹의 인바운드 규칙이 필요합니다. 3798
오류: 보안 그룹의 아웃바운드 규칙이 필요합니다. 3799
오류: 작업 실행에 실패했습니다. 그 이유는 전달된 역할에는 AWS Glue 서비스에 대한 역할 수임 권한이 주어져야 하기 때문입니다. 3799
오류: DescribeVpcEndpoints 작업이 승인되지 않았습니다. VPC ID vpc-id를 검증할 수 없습니다. 3799
오류: DescribeRouteTables 작업이 승인되지 않았습니다. VPC ID: vpc-id에서 서브넷 ID: 서브넷-id를 검증할 수 없습니다. 3799
오류: ec2:DescribeSubnets 호출이 실패했습니다. 3799
오류: ec2:DescribeSecurityGroups 호출이 실패했습니다. 3800
오류: AZ용 서브넷을 찾을 수 없습니다. 3800
오류: JDBC 대상으로 작성할 경우의 작업 실행 예외 3800
오류: Amazon S3: 객체의 스토리지 클래스에 대해 작업이 유효하지 않습니다. 3801

오류: Amazon S3 시간 제한 3801
오류: Amazon S3 액세스가 거부되었습니다. 3801
오류: Amazon S3 액세스 키 ID가 없습니다. 3801
오류: s3a:// URI를 사용해 Amazon S3에 액세스할 때 작업 실행이 실패합니다. 3802
오류: Amazon S3 서비스 토큰이 만료되었습니다. 3804
오류: 네트워크 인터페이스용 프라이빗 DNS를 찾을 수 없습니다. 3804
오류: 개발 엔드포인트 프로비저닝이 실패했습니다. 3804
오류: 노트북 서버 생성이 실패했습니다. 3804
오류: 로컬 노트북 시작이 실패했습니다. 3805
오류: 크롤러 실행이 실패했습니다. 3805
오류: 파티션이 업데이트되지 않았습니다. 3805
오류: 버전 불일치로 인해 작업 북마크 업데이트 실패 3806
오류: 작업 북마크가 사용 설정된 경우 작업이 데이터를 재처리합니다. 3806
오류: AWS Glue에서 VPC 간 장애 조치 동작 3807
크롤러가 Lake Formation 권한을 사용하는 경우 발생하는 크롤러 오류 3808
오류: The S3 location: s3://examplepath is not registered 3808
오류: User/Role is not authorized to perform: lakeformation:GetDataAccess on resource .. 3808
오류: Insufficient Lake Formation permission(s) on (Database name: exampleDatabase, Table Name: exampleTable) 3809
오류: Insufficient Lake Formation permission(s) on s3://examplepath 3809
Lake Formation 보안 인증을 사용한 크롤러 구성에 대해 자주 묻는 질문 3809
Ray 오류 해결 3810
Ray 작업 로그 검사 3811
Ray 작업 오류 해결 3811
AWS Glue 기계 학습 예외 사항 3813
CancelMLTaskRunActivity 3813
CreateMLTaskRunActivity 3813
DeleteMLTransformActivity 3815
GetMLTaskRunActivity 3815
GetMLTaskRunsActivity 3815
GetMLTransformActivity 3815
GetMLTransformsActivity 3816
GetSaveLocationForTransformArtifactActivity 3816
GetTaskRunArtifactActivity 3816
PublishMLTransformModelActivity 3817
PullLatestMLTransformModelActivity 3818

PutJobMetadataForMLTransformActivity 3818
StartExportLabelsTaskRunActivity 3819
StartImportLabelsTaskRunActivity 3819
StartMLEvaluationTaskRunActivity 3820
StartMLLabelingSetGenerationTaskRunActivity 3821
UpdateMLTransformActivity 3821
AWS Glue 할당량 3822
AWS Glue 성능 개선 3823
작업 유형에 맞는 튜닝 전략 3823
Spark 성능 개선 3823
푸시다운을 사용한 읽기 최적화 3824
Amazon S3에 저장된 파일에서 조건자 푸시다운 3824
JDBC 소스를 사용할 때 푸시다운 3825
AWS Glue에서 푸시다운에 대한 참고 및 제한 사항 3828
에 자동 조정 사용 AWS Glue 3828
요구 사항 3829
에서 Auto Scaling 활성화 AWS Glue Studio 1709
또는 를 AWS CLI 사용하여 Auto Scaling 활성화 SDK 1710
대화형 세션을 사용하여 Auto Scaling 활성화 3831
팁 및 고려 사항 3831
Amazon CloudWatch 지표를 사용한 Auto Scaling 모니터링 3832
Amazon CloudWatch Logs를 사용한 Auto Scaling 모니터링 3833
Spark UI로 Auto Scaling 모니터링 3834
Auto Scaling 작업 실행 DPU 사용량 모니터링 3834
제한 사항 3835
제한된 실행을 통한 워크로드 분할 3835
워크로드 분할 사용 3835
작업을 자동으로 실행하도록 AWS Glue 트리거 설정 3837
알려진 문제 3838
작업 간 데이터 액세스 방지 3838
문서 기록 3841
이전 업데이트 3890
AWS 용어집 3892

AWS Glue란 무엇인가요?

AWS Glue는 분석 사용자가 여러 소스의 데이터를 쉽게 검색, 준비, 이동, 통합할 수 있도록 하는 서버리스 데이터 통합 서비스입니다. 분석, 기계 학습 및 애플리케이션 개발에 사용할 수 있습니다. 또한 작성, 작업 실행, 비즈니스 워크플로 구현을 위한 추가 생산성 및 데이터 운영 도구도 포함됩니다.

AWS Glue를 사용하면 70개 이상의 다양한 데이터 소스를 검색하여 연결하고 중앙 집중식 데이터 카탈로그에서 데이터를 관리할 수 있습니다. 추출, 변환, 로드(ETL) 파이프라인을 시각적으로 생성, 실행, 모니터링하여 데이터 레이크에 데이터를 로드할 수 있습니다. 또한 Amazon Athena, Amazon EMR, Amazon Redshift Spectrum을 사용하여 카탈로그화된 데이터를 즉시 검색하고 쿼리할 수 있습니다.

AWS Glue는 주요 데이터 통합 기능을 단일 서비스로 통합합니다. 여기에는 데이터 검색, 최신 ETL, 정제, 변환, 중앙 집중식 카탈로그화가 포함됩니다. 또한 서버리스이므로 관리할 인프라가 없습니다. ETL, ELT, 스트리밍과 같은 모든 워크로드를 하나의 서비스에서 유연하게 지원하므로 AWS Glue는 다양한 워크로드 및 사용자 유형에 걸쳐 사용자를 지원합니다.

또한 AWS Glue를 사용하면 아키텍처 전반에 걸쳐 데이터를 쉽게 통합할 수 있습니다. AWS 분석 서비스 및 Amazon S3 데이터 레이크와 통합됩니다. AWS Glue는 개발자에서 비즈니스 사용자에 이르기까지 모든 사용자가 사용하기 쉬운 통합 인터페이스 및 작업 작성 도구를 보유하고 있으며 다양한 기술 세트에 대한 맞춤형 솔루션을 제공합니다.

필요에 따라 확장할 수 있는 기능을 통해 AWS Glue는 데이터의 가치를 극대화하는 고부가가치 활동에 집중할 수 있도록 도와줍니다. 모든 데이터 크기에 맞게 확장되며 모든 데이터 유형 및 스키마 변형을 지원합니다. 민첩성을 높이고 비용을 최적화하기 위해 AWS Glue는 기본 제공 고가용성 및 사용한 만큼 지불하는 청구서를 제공합니다.

요금 정보는 AWS Glue 요금을 참조하세요.

AWS Glue Studio

AWS Glue Studio는 AWS Glue에서 데이터 통합 작업을 쉽게 생성, 실행, 모니터링할 수 있는 그래픽 인터페이스입니다. 데이터 변환 워크플로를 시각적으로 구성하고 AWS Glue의 Apache Spark 기반 서버리스 ETL 엔진에서 원활하게 실행할 수 있습니다.

AWS Glue Studio를 사용하면 데이터를 수집, 변환, 정리하는 작업을 생성하고 관리할 수 있습니다. 또한 AWS Glue Studio를 사용하여 작업 스크립트 문제를 해결하고 작업 스크립트를 편집할 수 있습니다.

주제

- AWS Glue 기능

- AWS Glue의 혁신에 대해 알아보기
- AWS Glue 시작하기
- AWS Glue 액세스
- 관련 서비스

AWS Glue 기능

AWS Glue의 기능은 크게 세 가지 범주로 분류됩니다.

- 데이터 검색 및 구성
- 분석을 위한 데이터 변환, 준비, 정리
- 데이터 파이프라인 구축 및 모니터링

데이터 검색 및 구성

- 여러 데이터 스토어에 걸친 통합 및 검색 - AWS의 모든 데이터를 카탈로그로 분류하여 여러 데이터 소스 및 싱크에서 저장, 인덱싱, 검색할 수 있습니다.
- 데이터 자동 검색 - AWS Glue 크롤러를 사용하여 스키마 정보를 자동으로 추론하고 AWS Glue Data Catalog에 통합합니다.
- 스키마 및 권한 관리 - 데이터베이스와 테이블에 대한 액세스를 검증하고 제어합니다.
- 다양한 데이터 소스에 연결 - 데이터 레이크를 구축하기 위해 AWS Glue 연결을 사용하여 온프레미스와 AWS 모두에서 여러 데이터 소스를 활용합니다.

분석을 위한 데이터 변환, 준비, 정리

- 작업 캔버스 인터페이스를 사용하여 데이터를 시각적으로 변환 - 시각적 작업 편집기에서 ETL 프로세스를 정의하고 코드를 자동으로 생성하여 데이터를 추출, 변환, 로드할 수 있습니다.
- 간단한 작업 스케줄링으로 복잡한 ETL 파이프라인을 구축 - 일정에 따라 또는 요청 시 또는 이벤트를 기반으로 AWS Glue 작업을 호출합니다.
- 전송 중인 스트리밍 데이터 정리 및 변환 - 데이터를 지속적으로 사용하고 전송 중에 데이터를 정리 및 변환할 수 있습니다. 이렇게 하면 대상 데이터 스토어에서 몇 초 만에 분석할 수 있습니다.
- 기본 제공 기계 학습으로 데이터 중복 제거 및 정제 - `FindMatches` 기능을 사용하여 기계 학습 전문가가 아니더라도 분석을 위해 데이터를 정리하고 준비할 수 있습니다. 이 기능은 중복 데이터를 제거하고 서로 불완전하게 일치하는 레코드를 찾습니다.

- 기본 제공 작업 노트북 - AWS Glue 작업 노트북은 AWS Glue에서 최소한의 설정으로 서버리스 노트북을 제공하므로 빠르게 시작할 수 있습니다.
- ETL 코드 편집, 디버그, 테스트 - AWS Glue 대화형 세션을 통해 대화형 방식으로 데이터를 탐색하고 준비할 수 있습니다. 선택한 IDE 또는 노트북을 사용하여 대화형 방식으로 데이터를 탐색, 실험, 처리할 수 있습니다.
- 민감한 데이터의 정의, 탐지, 문제 해결 – AWS Glue 민감한 데이터 탐지를 통해 데이터 파이프라인과 데이터 레이크에서 민감한 데이터를 정의, 식별, 처리할 수 있습니다.

데이터 파이프라인 구축 및 모니터링

- 워크로드에 따라 자동 확장 - 워크로드에 따라 리소스를 동적으로 확장 및 축소할 수 있습니다. 이렇게 하면 필요한 경우에만 작업자를 작업에 할당합니다.
- 이벤트 기반 트리거로 작업 자동화 - 크롤러를 시작하거나 이벤트 기반 트리거를 사용하여 AWS Glue 작업을 수행하고 종속 작업 및 크롤러 체인을 설계합니다.
- 작업 실행 및 모니터링 - 원하는 엔진(Spark 또는 Ray)에서 AWS Glue 작업을 실행합니다. 자동 모니터링 도구, AWS Glue 작업 실행 인사이트, AWS CloudTrail 등을 사용하여 모니터링합니다. Apache Spark UI를 사용하여 Spark 지원 작업의 모니터링을 개선합니다.
- ETL 및 통합 활동을 위한 워크플로 정의 - 여러 크롤러, 작업, 트리거에 대한 ETL 및 통합 활동을 위한 워크플로를 정의합니다.

AWS Glue의 혁신에 대해 알아보기

AWS Glue의 최신 혁신에 대해 알아보고 고객이 조직 전체에서 셀프 서비스 데이터 준비를 지원하는 데 AWS Glue를 사용하는 방법을 알아보세요.

고객이 기존 설정 이상으로 AWS Glue 규모를 조정하는 방법과 작업 모니터링 및 성능을 위해 AWS Glue를 구성하는 방법도 알아보세요.

AWS Glue 시작하기

다음 단원부터 시작하는 것이 좋습니다.

- AWS Glue 사용 개요
- AWS Glue개념
- AWS Glue에 대한 IAM 권한 설정

- AWS Glue Data Catalog 시작하기
- AWS Glue의 작업 작성
- AWS Glue 대화형 세션 시작하기
- AWS Glue에서 오케스트레이션

AWS Glue 액세스

다음 인터페이스 중 하나를 사용하여 AWS Glue 작업을 생성하고, 액세스하고, 관리할 수 있습니다.

- AWS Glue콘솔 - AWS Glue 작업을 생성, 확인 및 관리할 수 있는 웹 인터페이스를 제공합니다. 콘솔에 액세스하려면 AWS Glue 단원을 참조하십시오.
- AWS Glue Studio - AWS Glue 작업을 시각적으로 만들고 편집할 수 있는 그래픽 인터페이스를 제공합니다. 자세한 내용은 AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 단원을 참조하십시오.
- AWS CLI 참조의 AWS Glue 섹션 - AWS Glue에서 사용할 수 있는 AWS CLI 명령을 제공합니다. 자세한 내용은 AWS Glue에 대한 AWS CLI 참조를 참조하세요.
- AWS Glue API - 개발자를 위한 전체 API 레퍼런스를 제공합니다. 자세한 내용은 AWS Glue API를 참조하세요.

관련 서비스

AWS Glue의 사용자는 또한 다음을 사용합니다.

- AWS Lake Formation - AWS Glue Data Catalog의 리소스에 대한 세분화된 액세스 제어를 제공하는 권한 부여 계층인 서비스입니다.
- AWS Glue DataBrew - 코드를 작성하지 않고 데이터를 정리하고 정규화하는 데 사용할 수 있는 시각적 데이터 준비 도구입니다.

AWS Glue: 작동 방식

AWS Glue는 다른 AWS 서비스를 통해 추출, 변환, 로드 작업을 오케스트레이션하여 데이터 웨어하우스 및 데이터 레이크를 빌드하고 출력 스트림을 생성합니다. AWS Glue는 API 작업을 호출하여 데이터를 변환하고, 런타임 로그를 생성하고, 작업 로직을 저장하고, 작업 실행을 모니터링하는 데 도움이 되는 알림을 생성합니다. AWS Glue 콘솔은 이런 서비스를 관리형 애플리케이션으로 연결하여 ETL 작업을 생성하고 모니터링하는 데 집중할 수 있습니다. 콘솔은 사용자를 대신하여 관리 및 작업 개발 작업을 수행합니다. 자격 증명 및 기타 속성을 AWS Glue에 제공하여 데이터 원본에 액세스하고 데이터 대상에 작성합니다.

AWS Glue는 워크로그가 실행되도록 필요한 리소스를 프로비저닝 및 관리를 담당합니다. ETL 도구를 사용하기 위해서 인프라를 생성할 필요가 없는 이유는 AWS Glue가 대신하기 때문입니다. 리소스가 필요한 경우, 시작 시간을 줄이기 위해 AWS Glue는 인스턴스의 준비된 풀에서 인스턴스를 사용하여 워크로드를 실행합니다.

AWS Glue를 통해 데이터 카탈로그에서 테이블 정의를 사용하여 작업을 생성합니다. 작업은 원하는 데이터 변환 작업을 실행하는 명령이 포함된 스크립트로 구성됩니다. 트리거를 사용하여 특정 이벤트의 일정에 따라 혹은 결과에 따라 작업을 시작합니다. 대상 데이터가 있는 곳과 어떤 소스 데이터가 대상으로 채우는지 결정합니다. 입력을 기반으로 AWS Glue에서는 소스에서 대상 형식으로 데이터를 변환합니다. 또는 AWS Glue 콘솔 또는 API에서 사용자 지정 스크립트를 제공하여 특정 요구 사항에 따라 데이터를 처리할 수도 있습니다.

데이터 소스 및 대상

AWS Glue for Spark를 사용하면 다음을 포함한 여러 시스템 및 데이터베이스에서 데이터를 읽고 쓸 수 있습니다.

- Amazon S3
- Amazon DynamoDB
- Amazon Redshift
- Amazon Relational Database Service(Amazon RDS)
- 타사 JDBC에서 액세스할 수 있는 데이터베이스
- MongoDB 및 Amazon DocumentDB(MongoDB와 호환)
- 기타 마켓플레이스 커넥터 및 Apache Spark 플러그인

데이터 스트림

AWS Glue for Spark는 다음 시스템에서 데이터를 스트리밍할 수 있습니다.

- Amazon Kinesis Data Streams
- Apache Kafka

AWS Glue는 여러 AWS 리전에서 사용할 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS Regions and endpoints를 참조하세요.

주제

- 격리 중인 서버리스 ETL 작업 실행
- AWS Glue 개념
- AWS Glue 구성 요소
- AWS Glue for Spark 및 AWS Glue for Ray
- 반구조화된 스키마를 AWS Glue가 포함된 관계형 스키마로 변환하기
- AWS Glue 유형 시스템

격리 중인 서버리스 ETL 작업 실행

AWS Glue는 선택한 엔진(Spark 또는 Ray)을 사용하여 서버리스 환경에서 ETL 작업을 실행합니다. AWS Glue는 이런 작업을 가상 리소스에서 실행하여 자체 서비스 계정을 프로비저닝하고 관리합니다.

AWS Glue는 다음과 같이 하도록 설계되어 있습니다.

- 사용자 지점 데이터 분리
- 전송 및 저장 시 사용자 데이터 보호
- 사용자 요청에 대항 응답으로만 임시 범위가 축소된 자격 증명 또는 계정의 IAM 역할에 대한 사용자 동의에 따라 사용자 데이터에 액세스합니다.

ETL 작업을 프로비저닝하는 중에 가상 사설 클라우드(VPC)에 입력 데이터 원본 및 출력 데이터 대상을 제공합니다. 또한, 데이터 원본 및 대상에 필요한 IAM 역할, VPC ID, 서브넷 ID 및 보안 그룹을 제공합니다. 각 튜플(사용자 계정 ID, IAM 역할, 서브넷 ID 및 보안 그룹)을 위해 AWS Glue는 AWS Glue 서비스 계정 내 다른 모든 환경의 네트워크 및 관리 수준이 격리된 새로운 환경을 생성합니다.

AWS 계정 내에서 Data Catalog, 작업 및 크롤러와 같은 AWS Glue 리소스를 생성하고 구성합니다. 그런 다음, 이러한 리소스는 생성 프로세스 중에 지정한 IAM 역할 및 네트워크 설정(서브넷 및 보안 그룹)과 연결됩니다.

AWS Glue는 프라이빗 IP 주소를 사용하여 탄력적 네트워크 인터페이스를 서브넷에 생성합니다. 작업은 탄력적 네트워크 인터페이스를 사용하여 데이터 소스와 데이터 대상에 액세스합니다. VPC 및 네트워크 정책은 다음 한 가지를 제외한 작업 실행 환경 내부 및 외부에 있는 트래픽을 관리합니다. 한 가지 예외 사항은 AWS Glue 라이브러리를 직접 호출하여 프록시 트래픽에서 AWS Glue API 작업을 AWS Glue VPC를 통해 가능하게 한 경우입니다. 모든 AWS Glue API 호출은 로깅됩니다. 따라서 데이터 소유자는 감사 로그를 계정으로 전달하는 AWS CloudTrail를 활성화하여 API 액세스를 감사할 수 있습니다.

ETL 작업을 실행하는 AWS Glue 관리형 환경은 뒤이어 다른 AWS 서비스가 따르는 동일한 보안 과정을 통해 보호됩니다. 사례 및 공유 보안 책임에 대한 개요는 AWS 보안 프로세스 소개 백서를 참조하세요.

AWS Glue 개념

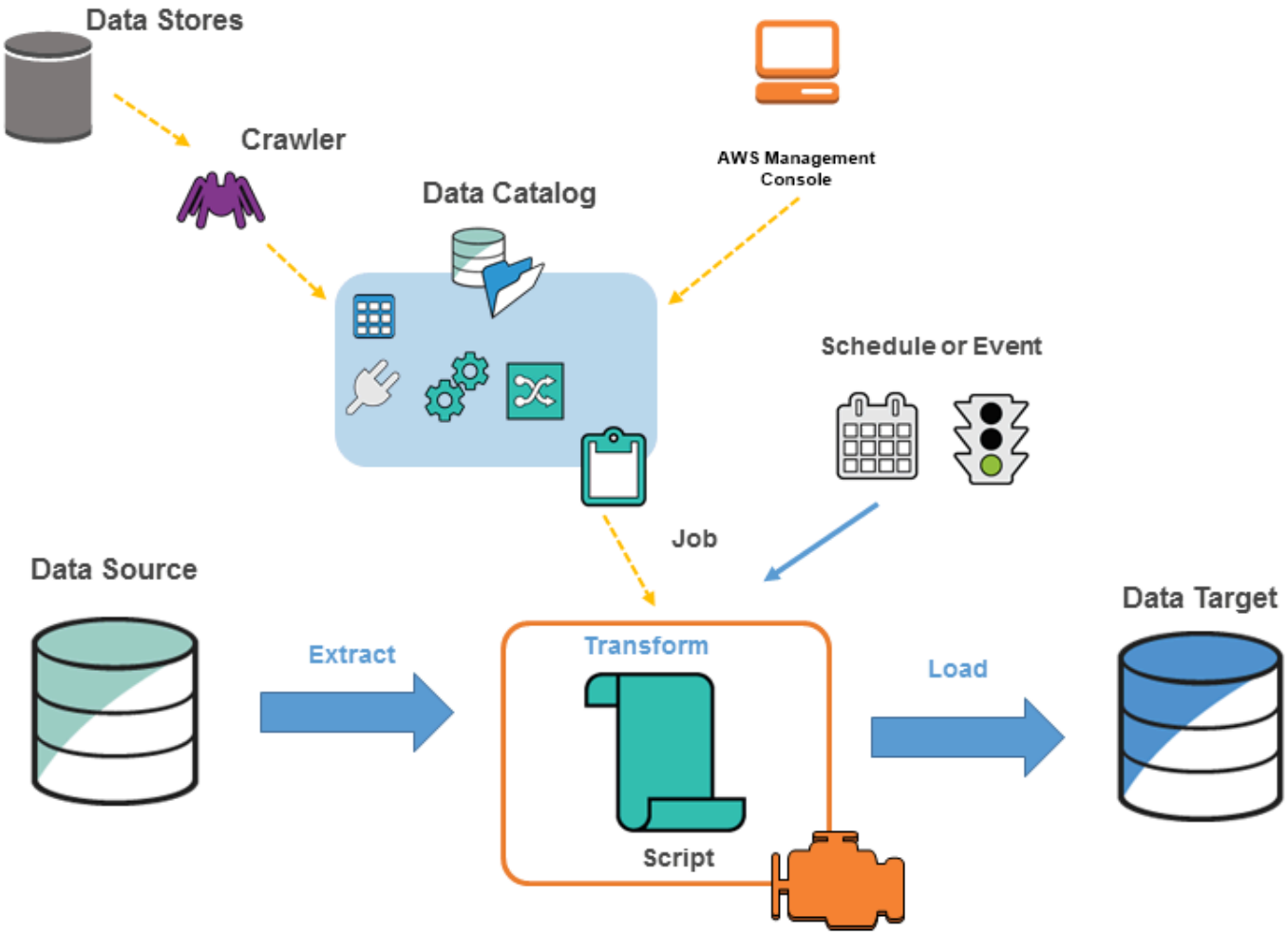
AWS Glue은(는) 완전 관리형 ETL(추출, 변환, 로드) 서비스로서 다양한 데이터 소스 및 타겟 간에 데이터를 쉽게 이동할 수 있습니다. 주요 구성 요소는 다음과 같습니다.

- 데이터 카탈로그: ETL 워크플로를 위한 테이블 정의, 작업 정의 및 기타 제어 정보가 포함되어 있는 메타데이터 저장소입니다.
- 크롤러: 데이터 소스에 연결하고, 데이터 스키마를 추론하고, 데이터 카탈로그에서 메타데이터 테이블 정의를 만드는 프로그램입니다.
- ETL 작업: 소스에서 데이터를 추출하고, Apache Spark 스크립트를 사용하여 데이터를 변환하고, 타겟에 로드하는 비즈니스 로직입니다.
- 트리거: 일정이나 이벤트를 기반으로 작업 실행을 시작하는 메커니즘.

일반적인 워크플로에는 다음 사항이 포함됩니다.

1. 데이터 카탈로그에서 데이터 소스 및 타겟을 정의합니다.
2. 크롤러를 사용하여 데이터 소스의 테이블 메타데이터로 데이터 카탈로그를 채웁니다.
3. 변환 스크립트로 ETL 작업을 정의하여 데이터를 이동하고 처리합니다.
4. 온디맨드 또는 트리거를 기반으로 작업을 실행합니다.
5. 대시보드를 사용하여 작업 성과를 모니터링합니다.

다음 다이어그램은 AWS Glue 환경의 아키텍처를 나타냅니다.



AWS Glue에서 작업을 정의하여 데이터 원본에서 데이터 대상으로 데이터를 추출, 변환, 로드하는 데 필요한 작업을 수행합니다. 일반적으로 다음 작업을 수행할 수 있습니다.

- 데이터 스토어 소스의 경우 크롤러를 정의하여 메타데이터 테이블 정의로 AWS Glue Data Catalog를 채웁니다. 데이터 스토어에서 크롤러를 포인트하면 크롤러는 Data Catalog에서 테이블 정의를 생성합니다. 스트리밍 소스의 경우 Data Catalog 테이블을 수동으로 정의하고 데이터 스트림 속성을 지정합니다.

  테이블 정의에 관련해 추가하자면 AWS Glue Data Catalog는 다른 필수 메타데이터를 포함하여 ETL 작업을 정의합니다. 작업을 정의하여 데이터를 변환할 경우 이 메타데이터를 사용합니다.
- AWS Glue는 스크립트를 생성하여 데이터를 변환할 있습니다. 또는 스크립트를 AWS Glue 콘솔이나 API에 제공합니다.

- 필요시 작업하거나 지정된 trigger가 발생하면 작업을 설정하여 시작할 수 있습니다. 트리거는 시간 기반 스케줄이나 이벤트일 수 있습니다.

  작업이 진행되면 스크립트는 데이터 원본에서 데이터를 추출하고 데이터를 변환한 다음 데이터 대상으로 로드합니다. 스크립트는 AWS Glue의 Apache Spark 환경에서 실행됩니다.

> **Important**
>
> AWS Glue에 있는 테이블과 데이터베이스는 AWS Glue Data Catalog의 객체입니다. 이 객체들은 메타데이터를 포함하지만 데이터 스토어의 데이터는 포함하지 않습니다.

CSV와 같은 텍스트 기반 데이터를 처리하려면 AWS Glue용으로 **UTF-8**에 인코딩해야 합니다. 자세한 내용은 Wikipedia의 UTF-8을 참조하세요.

AWS Glue 용어

AWS Glue에서는 여러 구성 요소의 상호 작용에 의존하여 추출, 전환, 적재(ETL) 워크플로를 생성하고 관리합니다.

AWS Glue Data Catalog

AWS Glue의 영구적 메타데이터 스토어입니다. 테이블 정의, 작업 정의 및 기타 관리 정보를 포함하여 AWS Glue 환경을 관리합니다. 각 AWS 계정에는 리전당 AWS Glue Data Catalog 하나가 있습니다.

분류자

데이터 스키마를 결정합니다. AWS Glue는 CSV, JSON, AVRO, XML 등과 같은 일반 파일 형식에 대한 분류자를 제공합니다. JDBC 연결을 사용한 일반 관계형 데이터베이스 관리 시스템을 위한 분류자를 제공합니다. a grok 패턴을 사용하거나 XML 문서에 행 태그를 지정하여 자체 분류자를 작성할 수 있습니다.

연결

특정 데이터 스토어에 연결하는 데 필요한 속성을 포함하는 Data Catalog 객체입니다.

크롤러

데이터 스토어(소스 또는 대상)에 연결하는 프로그램은 분류자의 우선 순위 지정 목록을 통해 데이터의 스키마를 결정한 다음 AWS Glue Data Catalog에 메타데이터 테이블을 생성합니다.

데이터베이스

논리 그룹으로 구성된 일련의 연결된 Data Catalog 테이블 정의입니다.

데이터 스토어, 데이터 원본, 데이터 대상

데이터 스토어는 데이터를 영구적으로 저장하기 위한 리포지토리입니다. 예를 들면 Amazon S3 버킷과 관계형 데이터베이스가 있습니다. 데이터 원본은 프로세스 또는 변환의 입력값으로 사용되는 데이터 스토어입니다. 데이터 대상은 프로세스 또는 변환에서 쓰기를 수행하는 대상 데이터 스토어입니다.

개발 엔드포인트

AWS Glue ETL 스크립트를 개발하고 테스트하는 데 사용할 수 있는 환경입니다.

동적 프레임

구조 및 배열과 같은 중첩 데이터를 지원하는 분산 테이블입니다. 각 레코드는 자기 설명적이며 반정형 데이터가 있는 스키마 유연성을 위해 설계되었습니다. 각 레코드에는 데이터와 해당 데이터를 설명하는 스키마가 모두 포함되어 있습니다. ETL 스크립트에서 동적 프레임과 Apache Spark DataFrame을 모두 사용하여 이 두 프레임 간에 변환할 수 있습니다. 동적 프레임은 데이터 정리 및 ETL에 필요한 고급 변환 세트를 지원합니다.

작업

작업은 ETL 작업을 수행하는 데 필요한 비즈니스 로직입니다. 변환 스크립트, 데이터 원본 및 데이터 대상으로 구성됩니다. 작업은 트리거에 의해 시작되고, 트리거는 예약되거나 이벤트 기반으로 작동할 수 있습니다.

작업 성능 대시보드

AWS Glue는 ETL 작업에 대한 포괄적인 실행 대시보드를 제공합니다. 대시보드에는 특정 기간의 작업 실행에 대한 정보가 표시됩니다.

노트북 인터페이스

간편한 작업 작성 및 데이터 탐색을 위해 원클릭 설정으로 향상된 노트북 환경. 노트북과 연결이 자동으로 구성됩니다. Jupyter Notebook 기반의 노트북 인터페이스를 통해 AWS Glue 서버리스 Apache

Spark ETL 인프라를 사용하여 스크립트와 워크플로를 대화형으로 개발, 디버깅, 배포할 수 있습니다. 노트북 환경에서 임시 쿼리, 데이터 분석, 시각화(예: 테이블 및 그래프)를 수행할 수도 있습니다.

Script

소스에서 데이터를 추출하고 변환하고 대상으로 로드하는 코드입니다. AWS Glue는 PySpark 또는 Scala 스크립트를 생성합니다.

표

데이터를 나타내는 메타데이터 정의입니다. 데이터가 Amazon Simple Storage Service(Amazon S3) 파일, Amazon Relational Database Service(Amazon RDS) 테이블 또는 다른 데이터 집합에 있는지에 관계없이 테이블은 데이터의 스키마를 정의합니다. AWS Glue Data Catalog의 테이블은 열 이름, 데이터 유형 정의, 파티션 정보 및 베이스 데이터 세트에 대한 기타 메타데이터로 구성됩니다. 데이터의 스키마는 AWS Glue 테이블 정의에 나타납니다. 실제 데이터는 파일 혹은 관계형 데이터베이스 테이블 어디에 있든지 기존 데이터 스토어에 남겨집니다. AWS Glue는 파일과 관계형 데이터베이스 테이블 목록을 AWS Glue Data Catalog에 작성합니다. ETL 작업을 생성할 경우 소스와 대상로써 사용됩니다.

변환

다른 포맷으로 데이터를 바꾸는 코드 로직.

트리거

ETL 작업 시작. 트리거는 정해진 시간 혹은 이벤트 기반으로 정의할 수 있습니다.

시각적 작업 편집기

시각적 작업 편집기는 AWS Glue에서 추출, 전환, 적재(ETL) 작업을 쉽게 생성, 실행, 모니터링할 수 있게 해주는 그래픽 인터페이스입니다. 데이터 변환 워크플로를 시각적으로 구성하고 AWS Glue의 Apache Spark 기반 서버리스 ETL 엔진에서 원활하게 실행하며 작업의 각 단계에서 스키마와 데이터 결과를 검사할 수 있습니다.

작업자

AWS Glue를 사용하면 ETL 작업을 실행하는 데 걸리는 시간 요금만 결제하면 됩니다. 관리할 리소스와 사전 투자 비용이 없으며 시작 시간 또는 종료 시간에 대한 요금이 부과되지 않습니다. ETL 작업을 실행하는 데 사용된 데이터 처리 단위(DPU) 수에 따라 시간당 요금이 청구됩니다. 단일 데이터 처리 단위(DPU)를 작업자라고도 합니다. AWS Glue에는 작업 대기 시간 및 비용 요구 사항을 충족하는 구성을 선택하는 데 도움이 되는 세 가지 작업자 유형이 있습니다. 작업자는 표준, G.1X, G.2X 및 G.025X 구성으로 제공됩니다.

AWS Glue 구성 요소

AWS Glue는 콘솔 및 API 작업을 제공하여 워크로드를 추출, 변환 및 로드(ETL)하는 것을 설정하고 관리합니다. 특정 언어의 SDK 및 AWS Command Line Interface(AWS CLI)를 통해 API 작업을 사용할 수 있습니다. AWS CLI 사용에 대한 자세한 내용은 AWS CLI Command Reference를 참조하세요.

AWS Glue는 AWS Glue Data Catalog를 사용하여 데이터 원본, 변환 및 대상의 메타데이터를 저장합니다. Data Catalog는 Apache Hive Metastore의 드롭인 교체물입니다. AWS Glue Jobs system는 데이터의 ETL 작업을 정의, 일정 관리 및 실행하기 위한 관리된 인프라를 제공합니다. AWS Glue API에 대한 자세한 내용은 AWS Glue API을 참조하십시오.

AWS Glue 콘솔

AWS Glue 콘솔을 사용하여 ETL 워크플로우를 정의하고 관리할 수 있습니다. 콘솔은 AWS Glue Data Catalog 및 AWS Glue Jobs system에서 몇 가지 API 작업을 호출하여 다음 작업을 실행합니다.

- 작업, 테이블, 크롤러 및 연결과 같은 AWS Glue 객체를 정의합니다.
- 크롤러를 실행하는 일정
- 작업 트리거를 위한 이벤트 또는 일정을 정의합니다.
- AWS Glue 객체 목록을 검색하고 필터링합니다.
- 변환 스크립트를 편집합니다.

AWS Glue Data Catalog

AWS Glue Data Catalog는 AWS 클라우드에 있는 영구적 기술 메타데이터 스토어입니다.

각 AWS 계정에는 AWS 리전당 AWS Glue Data Catalog가 하나씩 있습니다. 각 데이터 카탈로그는 데이터베이스로 구성된 확장성이 뛰어난 테이블 모음입니다. 테이블은 Amazon RDS, Apache Hadoop 분산 파일 시스템, Amazon OpenSearch Service 등의 소스에 저장된 정형 또는 반정형 데이터 모음을 메타데이터로 표현한 것입니다. AWS Glue Data Catalog는 일정한 리포지토리를 제공합니다. 그러면 전혀 다른 시스템들이 메타데이터를 저장하고 탐색하여 데이터 사일로에서 데이터를 추적할 수 있습니다. 그런 다음 메타데이터를 사용하여 다양한 애플리케이션에서 일관된 방식으로 해당 데이터를 쿼리하고 변환할 수 있습니다.

데이터 카탈로그를 AWS Identity and Access Management 정책 및 Lake Formation과 함께 사용하여 테이블 및 데이터베이스에 대한 액세스를 제어합니다. 이렇게 하면 기업의 여러 그룹이 더 광범위한 조직에 데이터를 안전하게 게시하면서 매우 세분화된 방식으로 민감한 정보를 보호할 수 있습니다.

데이터 카탈로그는 CloudTrail 및 Lake Formation과 함께 스키마 변경 추적 및 데이터 액세스 제어와 함께 포괄적인 감사 및 거버넌스 기능도 제공합니다. 이렇게 하면 데이터가 부적절하게 수정되거나 실수로 공유되지 않도록 보장할 수 있습니다.

AWS Glue Data Catalog 보안 및 감사에 대한 자세한 내용은 다음을 참조하세요.

- AWS Lake Formation – 자세한 내용은 AWS Lake Formation 개발자 안내서의 AWS Lake Formation이란 무엇입니까?를 참조하세요.
- CloudTrail – 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail이란 무엇입니까?를 참조하세요.

다음은 AWS Glue Data Catalog를 사용하는 기타 AWS 서비스 및 오픈 소스 프로젝트입니다.

- Amazon Athena – 자세한 내용은 Amazon Athena 사용 설명서의 테이블, 데이터베이스, 데이터 카탈로그 이해를 참조하세요.
- Amazon Redshift Spectrum – 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 Amazon Redshift Spectrum을 사용하여 외부 데이터 쿼리를 참조하세요.
- Amazon EMR – 자세한 내용은 Amazon EMR 관리 안내서의 AWS Glue Data Catalog에 대한 Amazon EMR 액세스에 리소스 기반 정책 사용을 참조하세요.
- Apache Hive 메타스토어용 AWS Glue Data Catalog 클라이언트 – 이 GitHub 프로젝트에 대한 자세한 내용은 AWS Glue Data Catalog Client for Apache Hive Metastore를 참조하세요.

AWS Glue 크롤러 및 분류자

AWS Glue는 모든 종류의 리포지토리에서 데이터를 스캔하고 분류하며, 스키마 정보를 추출하고, AWS Glue Data Catalog에서 자동적으로 메타데이터를 저장하는 크롤러를 설정할 수 있습니다. AWS Glue Data Catalog를 사용하여 ETL 작업을 관리할 수 있습니다.

크롤러 및 분류자 설정 방법에 대한 자세한 내용은 크롤러를 사용하여 데이터 카탈로그 채우기 단원을 참조하십시오. AWS Glue API를 사용하는 크롤러 및 분류자를 프로그래밍하는 방법에 대한 자세한 내용은 크롤러 및 분류자 API 단원을 참조하십시오.

AWS Glue ETL 연산

Data Catalog에서 메타데이터를 사용하여 AWS Glue는 다양한 ETL 작업을 실행할 때 사용하고 수정할 수 있는 AWS Glue 확장자를 붙여 Scala 또는 PySpark(Apache Spark용 Python API) 스크립트를 자

동으로 생성할 수 있습니다. 예를 들어, 원 데이터를 추출, 정리 및 변환한 다음 결과가 쿼리되고 분석될 수 있는 다른 리포지토리에 결과를 저장할 수 있습니다. 이러한 스크립트는 CSV 파일을 관계 형식으로 변환하고 Amazon Redshift에 저장할 수 있습니다.

AWS Glue ETL 기능을 사용하는 방법에 대한 자세한 내용은 Spark 스크립트 프로그래밍 단원을 참조하십시오.

AWS Glue의 스트리밍 ETL

AWS Glue를 사용하면 지속적으로 실행되는 작업을 사용하여 스트리밍 데이터에 대해 ETL 작업을 수행할 수 있습니다. AWS Glue 스트리밍 ETL은 Apache Spark Structured Streaming 엔진을 기반으로 하며 Amazon Kinesis Data Streams, Apache Kafka 및 Amazon Managed Streaming for Apache Kafka (Amazon MSK)에서 스트림을 수집할 수 있습니다. 스트리밍 ETL은 스트리밍 데이터를 정리하고 변환하여 Amazon S3 또는 JDBC 데이터 스토어에 로드할 수 있습니다. AWS Glue의 스트리밍 ETL을 사용하여 IoT 스트림, 클릭스트림 및 네트워크 로그와 같은 이벤트 데이터를 처리합니다.

스트리밍 데이터 원본의 스키마를 알고 있는 경우 Data Catalog 테이블에서 지정할 수 있습니다. 그렇지 않은 경우 스트리밍 ETL 작업에서 스키마 감지를 사용할 수 있습니다. 그런 다음 작업은 들어오는 데이터에서 스키마를 자동으로 결정합니다.

스트리밍 ETL 작업은 AWS Glue 기본 제공 변환과 Apache Spark Structured Streaming에 대한 기본 변환을 모두 사용할 수 있습니다. 자세한 내용은 Apache Spark 웹 사이트의 Operations on streaming DataFrames/Datasets를 참조하세요.

자세한 내용은 the section called “스트리밍 ETL 작업” 단원을 참조하십시오.

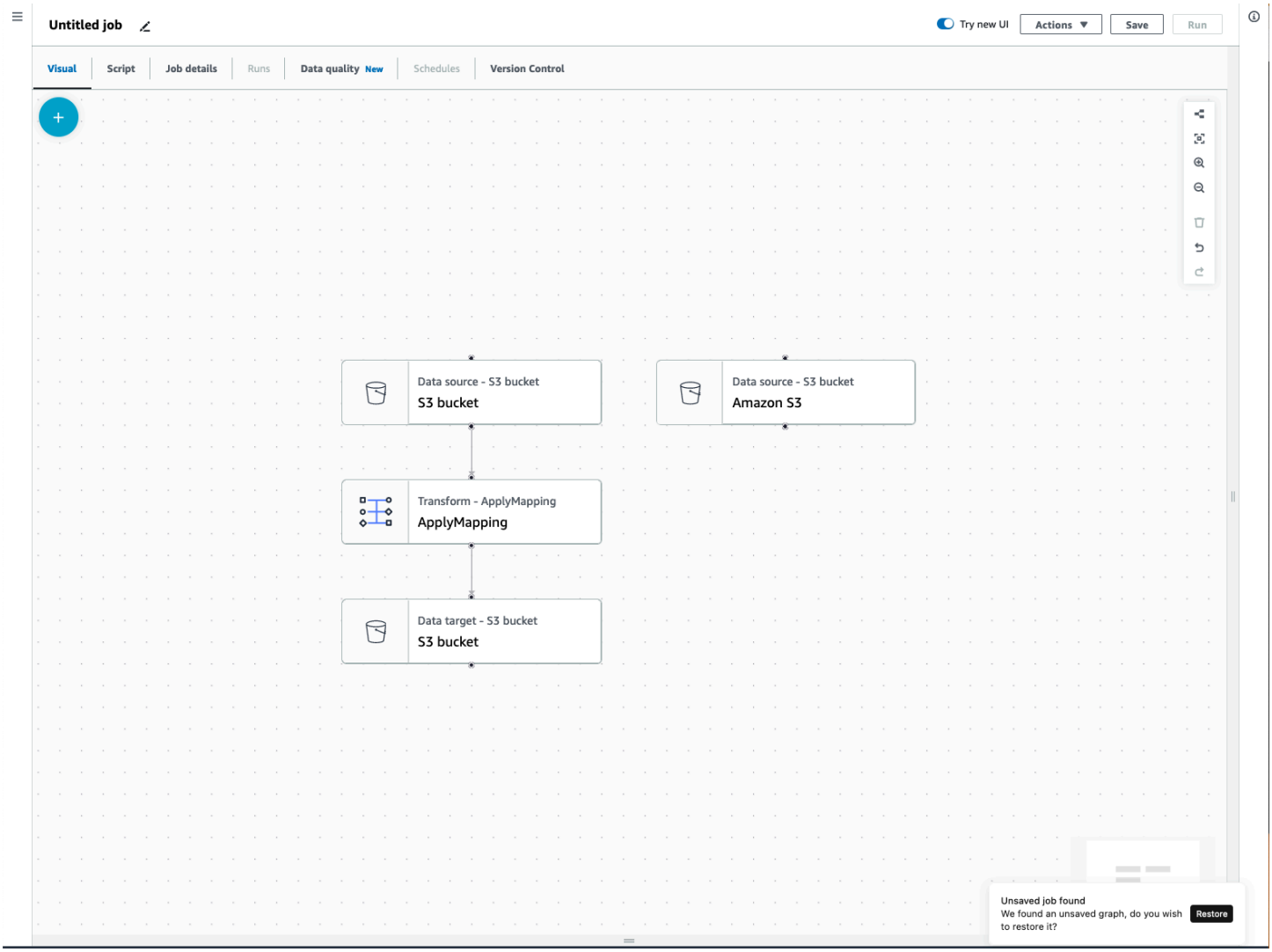
AWS Glue 작업 시스템

AWS Glue Jobs system는 관리된 인프라를 사용하여 ETL 워크플로우를 관리할 수 있습니다. 사용하는 스크립트를 통해 자동적으로 데이터를 다른 위치로 추출, 변환 및 전송할 수 있는 작업을 AWS Glue에 생성할 수 있습니다. 작업은 일정이 정해지고 모을 수 있고 새로운 데이터가 도착하는 것과 같은 이벤트에 의해 촉발될 수 있습니다.

AWS Glue Jobs system 사용에 대한 자세한 내용은 AWS Glue 모니터링 단원을 참조하십시오. AWS Glue Jobs system API를 사용하여 프로그래밍하는 것에 대한 자세한 내용은 작업 API 단원을 참조하십시오.

시각적 ETL 구성 요소

AWS Glue에서는 사용자가 조작할 수 있는 시각적 캔버스를 통해 ETL 작업을 생성할 수 있습니다.



ETL 작업 메뉴

캔버스 상단의 메뉴 옵션을 통해 작업에 대한 다양한 보기 및 구성 세부 정보에 액세스할 수 있습니다.

- 시각적 - 시각적 작업 편집기 캔버스입니다. 여기서 노드를 추가하여 작업을 생성할 수 있습니다.
- 스크립트 - ETL 작업의 스크립트 표현입니다. AWS Glue는 작업의 시각적 표현을 기반으로 스크립트를 생성합니다. 스크립트를 편집하거나 다운로드할 수도 있습니다.

> **Note**
>
> 스크립트를 편집하려는 경우 작업 작성 환경이 스크립트 전용 모드로 영구적으로 전환됩니다. 이후에는 더 이상 시각적 편집기를 사용하여 작업을 편집할 수 없습니다. 스크립트 편집

을 선택하기 전에 모든 작업 소스, 변환 및 대상을 추가하고 시각적 편집기를 사용하여 필요한 모든 변경 사항을 수행해야 합니다.

- 작업 세부 정보 - 작업 세부 정보 탭에서는 작업 속성을 설정하여 작업을 구성할 수 있습니다. 여기에는 기본 속성(예: 작업 이름 및 설명, IAM 역할, 작업 유형, AWS Glue 버전, 언어, 작업자 유형, 작업자 유형, 작업자 수, 작업 북마크, Flex 실행, 사용 중지 횟수, 작업 제한 시간 초과)과 함께, 연결, 라이브러리, 작업 파라미터, 태그와 같은 고급 속성도 있습니다.
- 실행 - 작업을 실행한 후 이 탭에 액세스하여 이전 작업 실행을 볼 수 있습니다.
- 데이터 품질 - 데이터 품질에서는 데이터 자산의 품질을 평가하고 모니터링합니다. 이 탭에서 데이터 품질을 사용하는 방법에 대해 자세히 알아보고 작업에 데이터 품질 변환을 추가할 수 있습니다.
- 일정 - 예약한 작업이 이 탭에 표시됩니다. 이 작업에 연결된 일정이 없는 경우 이 탭에는 액세스할 수 없습니다.
- 버전 제어 - 작업을 Git 리포지토리로 구성하여 작업에서 Git를 사용할 수 있습니다.

시각적 ETL 패널

캔버스에서 작업하는 경우 노드를 구성하거나 데이터를 미리 보고 출력 스키마를 보는 데 도움이 되는 여러 패널을 사용할 수 있습니다.

- 속성 - 캔버스에서 노드를 선택하면 속성 패널이 나타납니다.
- 데이터 미리 보기 - 데이터 미리 보기 패널에서는 데이터 출력의 미리 보기를 제공하므로 작업을 실행하기 전에 결정을 내리고 출력을 살펴볼 수 있습니다.
- 출력 스키마 - 출력 스키마 탭을 사용하면 변환 노드의 스키마를 보고 편집할 수 있습니다.

패널 크기 조정

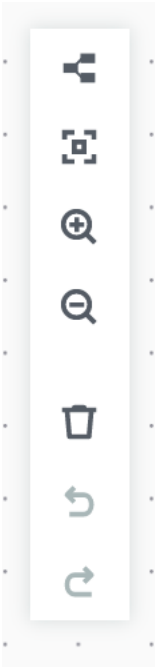
데이터 미리 보기 및 출력 스키마 탭이 포함된 하단 패널과 화면 오른쪽에 있는 속성 패널은 패널 가장자리를 클릭한 상태로 좌우 또는 위아래로 끌어 크기를 조정할 수 있습니다.

- 속성 패널 - 화면 오른쪽의 캔버스 가장자리를 클릭한 상태로 끌어 속성 패널의 크기를 조정한 후 왼쪽으로 끌어 너비를 확장합니다. 기본적으로 패널은 축소된 상태이며, 노드를 선택하면 속성 패널이 기본 크기로 열립니다.
- 데이터 미리 보기 및 출력 스키마 패널 - 화면 하단에서 캔버스 하단 가장자리를 클릭한 상태로 끌어 하단 패널의 크기를 조정한 후 위로 끌어 높이를 확장합니다. 기본적으로 패널은 축소된 상태이며, 노드를 선택하면 하단 패널이 기본 크기로 열립니다.

작업 캔버스

시각적 ETL 캔버스에서 직접 노드를 추가, 제거, 이동/재정렬할 수 있습니다. 작업 캔버스는 데이터 소스로 시작하고 데이터 대상으로 끝날 수 있는 완전한 기능을 갖춘 ETL 작업을 생성할 수 있는 작업 공간과 같습니다.

캔버스에서 노드에 대한 작업을 수행할 때 확대 및 축소, 노드 제거, 노드 간 연결 설정 또는 편집, 작업 흐름 방향 변경, 작업 실행 취소 또는 재실행 등을 도와주는 도구 모음이 있습니다.

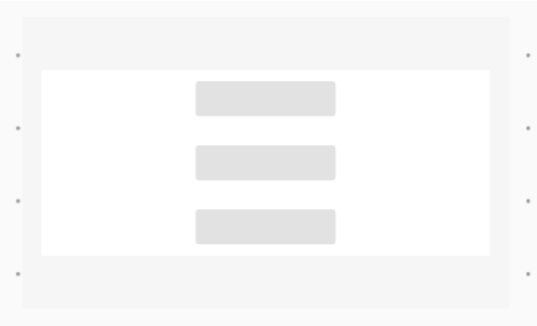


부동 도구 모음은 캔버스의 오른쪽 상단 크기로 고정되어 있으며 다음과 같이 작업을 수행하는 여러 이미지를 포함합니다.

- 레이아웃 아이콘 - 도구 모음의 첫 번째 아이콘은 레이아웃 아이콘입니다. 기본적으로 시각적 작업의 방향은 위에서 아래 방향입니다. 노드를 왼쪽에서 오른쪽으로 가로로 정렬하여 시각적 작업의 방향을 재정렬할 수 있습니다. 레이아웃 아이콘을 다시 클릭하면 방향이 위에서 아래로 다시 바뀝니다.
- 가운데 재정렬 아이콘 - 가운데 재정렬 아이콘은 캔버스 보기를 가운데에 배치하여 변경합니다. 대규모 작업의 경우 이 아이콘을 사용하여 가운데 위치로 돌아갈 수 있습니다.
- 확대 아이콘 - 확대 아이콘은 캔버스의 노드 크기를 확대합니다.
- 축소 아이콘 - 축소 아이콘은 캔버스의 노드 크기를 축소합니다.
- 휴지통 아이콘 - 휴지통 아이콘은 시각적 작업에서 노드를 제거합니다. 먼저 노드를 선택해야 합니다.
- 실행 취소 아이콘 - 실행 취소 아이콘은 시각적 작업에서 수행한 마지막 작업을 되돌립니다.

- 다시 실행 아이콘 - 다시 실행 아이콘은 시각적 작업에서 수행한 마지막 작업을 반복합니다.

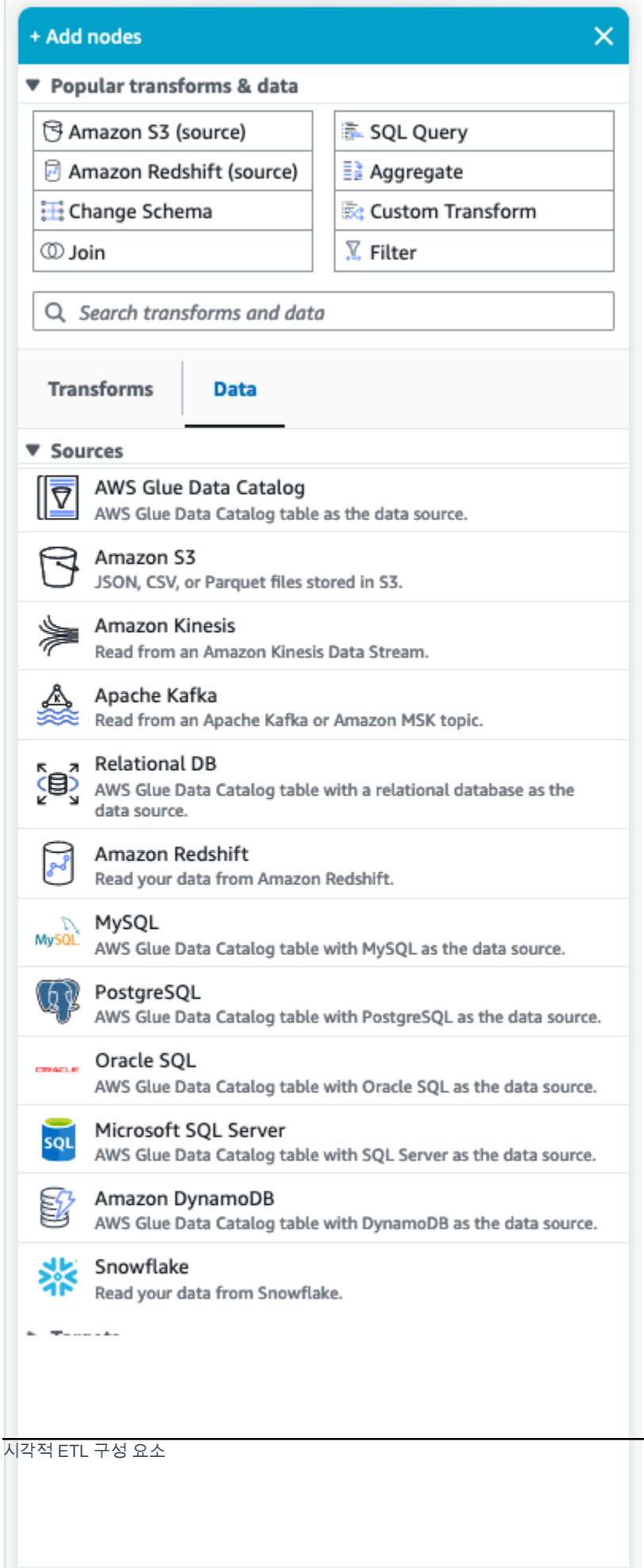
미니 맵 사용



리소스 패널

리소스 패널에는 사용자가 사용할 수 있는 모든 데이터 소스, 변환 작업 및 연결이 포함되어 있습니다. '+' 아이콘을 클릭하여 캔버스에서 리소스 패널을 엽니다. 그러면 리소스 패널이 열립니다.

리소스 패널을 닫으려면 리소스 패널의 오른쪽 상단에 있는 X를 클릭합니다. 그러면 패널을 다시 열 준비가 될 때까지 패널을 숨깁니다.


+ Add nodes
Popular transforms & data
Amazon S3 (source)
Amazon Redshift (source)
Change Schema
Join
SQL Query
Aggregate
Custom Transform
Filter
Search transforms and data
Transforms
Data
Sources
AWS Glue Data Catalog
AWS Glue Data Catalog table as the data source.
Amazon S3
JSON, CSV, or Parquet files stored in S3.
Amazon Kinesis
Read from an Amazon Kinesis Data Stream.
Apache Kafka
Read from an Apache Kafka or Amazon MSK topic.
Relational DB
AWS Glue Data Catalog table with a relational database as the data source.
Amazon Redshift
Read your data from Amazon Redshift.
MySQL
AWS Glue Data Catalog table with MySQL as the data source.
PostgreSQL
AWS Glue Data Catalog table with PostgreSQL as the data source.
Oracle SQL
AWS Glue Data Catalog table with Oracle SQL as the data source.
Microsoft SQL Server
AWS Glue Data Catalog table with SQL Server as the data source.
Amazon DynamoDB
AWS Glue Data Catalog table with DynamoDB as the data source.
Snowflake
Read your data from Snowflake.

자주 사용하는 변환 및 데이터

패널 상단에는 자주 사용하는 변환 및 데이터 컬렉션이 있습니다. 이 노드는 일반적으로 AWS Glue에서 사용됩니다. 하나를 선택하여 캔버스에 추가합니다. 자주 사용하는 변환 및 데이터 제목 옆의 삼각형을 클릭하여 자주 사용하는 변환 및 데이터를 숨길 수도 있습니다.

자주 사용하는 변환 및 데이터 섹션 아래에서 변환 및 데이터 소스 노드를 검색할 수 있습니다. 내용을 입력하면 결과가 표시됩니다. 검색 쿼리를 길게 입력할수록 결과 목록이 작아집니다. 검색 결과는 노드 이름 및/또는 설명으로 채워집니다. 노드를 선택하여 캔버스에 추가합니다.

변환 및 데이터

노드를 변환 및 데이터로 구성하는 두 개의 탭이 있습니다.

변환 - 변환 탭을 선택하면 사용 가능한 모든 변환을 선택할 수 있습니다. 변환을 선택하여 캔버스에 추가합니다. 변환 목록 하단에서 변환 추가를 선택할 수도 있습니다. 그러면 사용자 지정 시각적 변환 생성에 관한 설명서의 새 페이지가 열립니다. 단계를 따라 직접 변환을 생성할 수 있습니다. 그러면 변환이 사용 가능한 변환 목록에 표시됩니다.

데이터 - 데이터 탭에는 소스 및 대상의 모든 노드가 포함되어 있습니다. 소스 또는 대상 제목 옆의 삼각형을 클릭하여 소스 및 대상을 숨길 수 있습니다. 삼각형을 다시 클릭하면 숨긴 소스 및 대상을 다시 표시할 수 있습니다. 소스 또는 대상 노드를 선택하여 캔버스에 추가합니다. 연결 관리를 선택하여 새 연결을 추가할 수도 있습니다. 그러면 콘솔에서 커넥터 페이지가 열립니다.

AWS Glue for Spark 및 AWS Glue for Ray

AWS Glue on Apache Spark(AWS Glue ETL)에서 PySpark를 사용하여 대규모 데이터를 처리하는 Python 코드를 작성할 수 있습니다. Spark는 이 문제에 대한 친숙한 솔루션이지만 Python 중심 배경을 갖춘 데이터 엔지니어는 이 전환이 직관적이지 않다고 생각할 수 있습니다. Spark DataFrame 모델은 이 모델의 기반이 되는 Scala 언어와 Java 런타임을 반영하는 “Pythonic”이 원활하지 않습니다.

AWS Glue에서는 Python 쉘 작업을 사용하여 기본 Python 데이터 통합을 실행할 수 있습니다. 이러한 작업은 단일 Amazon EC2 인스턴스에서 실행되며 해당 인스턴스의 용량에 따라 제한됩니다. 따라서 처리할 수 있는 데이터의 처리량이 제한되고 빅 데이터를 처리할 때 유지 관리 비용이 많이 듭니다.

AWS Glue for Ray를 사용하면 Spark 학습에 많은 투자를 하지 않고도 Python 워크로드를 확장할 수 있습니다. Ray가 더 잘 작동하는 특정 시나리오를 활용할 수 있습니다. 선택권을 제공하여 Spark와 Ray의 장점을 모두 활용할 수 있습니다.

AWS Glue ETL과 AWS Glue for Ray는 기본적으로 다르므로 서로 다른 기능을 지원합니다. 지원되는 기능은 설명서를 참조하세요.

AWS Glue for Ray란 무엇인가요?

Ray는 Python을 중심으로 워크로드를 확장하는 데 사용될 수 있는 오픈 소스 분산 계산 프레임워크입니다. Ray에 대한 자세한 내용은 Ray 웹사이트를 참조하세요. AWS Glue Ray 작업 및 대화형 세션을 활용하면 Ray를 AWS Glue 내에서 사용할 수 있습니다.

AWS Glue for Ray를 사용하여 여러 시스템에서 병렬로 실행되는 계산을 위한 Python 스크립트를 작성할 수 있습니다. Ray 작업 및 대화형 세션에서는 pandas와 같은 친숙한 Python 라이브러리를 사용하여 워크플로를 쉽게 작성하고 실행할 수 있습니다. Ray 데이터 세트에 대한 자세한 내용은 Ray 설명서의 Ray 데이터 세트를 참조하세요. Pandas에 대한 자세한 내용은 Pandas 웹 사이트를 참조하세요.

AWS Glue for Ray를 사용하면 단 몇 줄의 코드만으로 엔터프라이즈 규모의 빅 데이터에 대해 Pandas 워크플로를 실행할 수 있습니다. AWS Glue 콘솔 또는 AWS SDK에서 Ray 작업을 생성할 수 있습니다. AWS Glue 대화형 세션을 열어 서버리스 Ray 환경에서 코드를 실행할 수도 있습니다. AWS Glue Studio의 시각적 작업은 아직 지원되지 않습니다.

AWS Glue for Ray 작업을 통해 일정에 따라 또는 Amazon EventBridge의 이벤트에 대한 응답으로 스크립트를 실행할 수 있습니다. 작업은 스크립트의 상태와 신뢰성을 이해하는 데 도움이 되는 로그 정보 및 CloudWatch의 모니터링 통계를 저장합니다. AWS Glue 작업 시스템에 대한 자세한 내용은 the section called “Ray 작업 사용” 섹션을 참조하세요.

Ray는 부하에 따라 실시간으로 재구성하는 시스템 클러스터에 처리를 분산하여 Python 코드 크기 조정 작업을 자동화합니다. 따라서 특정 워크로드의 가격 대비 성능이 향상될 수 있습니다. Ray 작업에서는 auto Scaling이 AWS Glue 작업 모델에 기본적으로 제공되므로 이 기능을 최대한 활용할 수 있습니다. Ray 작업은 AWS Graviton에서 실행되므로 전반적인 가격 대비 성능이 향상됩니다.

비용 절감 이외에 기본 auto Scaling을 사용하여 클러스터 유지 관리, 조정 및 관리에 시간을 투자하지 않고도 Ray 워크로드를 실행할 수 있습니다. Pandas와 같은 친숙한 기존 오픈 소스 라이브러리와 AWS SDK for Pandas를 사용할 수 있습니다. 그러면 AWS Glue for Ray에서 개발하는 동안 반복 속도가 향상됩니다. AWS Glue for Ray를 사용하면 비용 효율적인 데이터 통합 워크로드를 신속하게 개발하여 실행할 수 있습니다.

반구조화된 스키마를 AWS Glue가 포함된 관계형 스키마로 변환하기

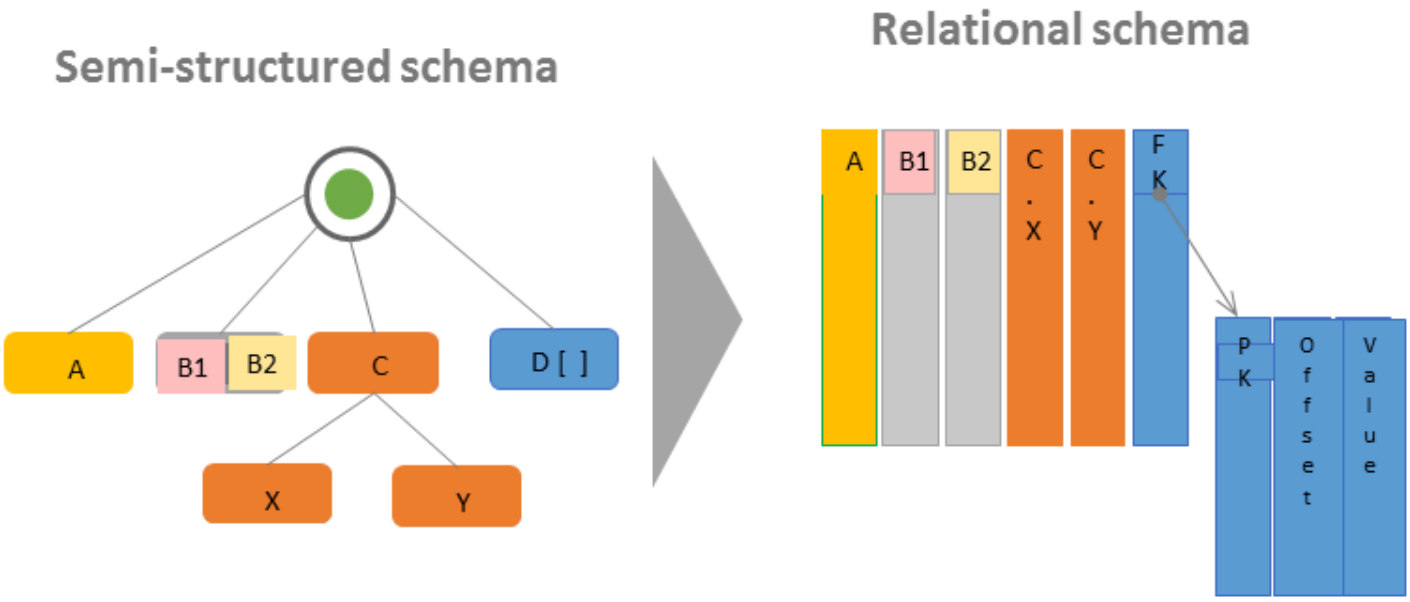
일반적으로 반구조화된 데이터를 관계형 테이블로 변환하고자 할 것입니다. 개념적으로는 계층적 스키마를 관계형 스키마로 평면화하는 것입니다. AWS Glue는 이 변환을 바로 실행할 수 있습니다.

반구조화된 데이터는 일반적으로 데이터 내 개체를 식별하기 위해서 마크업을 포함합니다. 고정된 스키마없이 중첩된 데이터 구조를 가질 수 있습니다. 반구조화된 데이터에 대한 자세한 내용은 Wikipedia에서 반구조화된 데이터를 참조하십시오.

관계형 데이터는 행과 열로 조직된 테이블이 보여줍니다. 테이블간 관계는 기본 키(PK)와 외래 키(FK) 관계에 의해 표현됩니다. 자세한 내용은 Wikipedia의 관계형 데이터베이스를 참조하십시오.

AWS Glue는 크롤러를 사용하여 반구조화된 데이터용 스키마를 유추합니다. ETL(추출, 변환 및 로드) 작업을 사용하여 데이터를 관계형 스키마로 변환합니다. 예를 들어, JSON 데이터를 Amazon Simple Storage Service(Amazon S3) 소스 파일에서 Amazon Relational Database Service (Amazon RDS) 테이블로 구문 분석해야 할 경우가 있습니다. AWS Glue가 어떻게 스키마간 차이를 다루는지 알면 변환 절차를 이해할 수 있습니다.

이 다이어그램은 AWS Glue가 어떻게 반구조화된 스키마를 관계형 스키마로 변환하는지 보여줍니다.



다이어그램은 다음을 보여 줍니다.

- 단일 값 A는 관계형 열로 바로 변환됩니다.
- B1 및 B2인 값 페어는 두 개의 관계형 열로 변환됩니다.
- 하위 X 및 Y가 있는 C 구조는 두 개의 관계형 열로 변환됩니다.
- D[] 배열은 다른 관계형 테이블을 가리키는 외래 키(FK)가 있는 관계형 열로 변환합니다. 기본 키(PK)와 함께 두 번째 관계형 테이블은 배열 아이템의 오프셋과 값을 포함하는 열을 보유하고 있습니다.

AWS Glue 유형 시스템

AWS Glue는 여러 유형 시스템을 사용하여 매우 다른 방식으로 데이터를 저장하는 데이터 시스템에 대해 다양한 인터페이스를 제공합니다. 이 문서는 AWS Glue 유형 시스템 및 데이터 표준을 명확하게 설명합니다.

AWS Glue 데이터 카탈로그 유형

데이터 카탈로그는 다양한 데이터 시스템인 메타스토어에 저장된 테이블 및 필드의 레지스트리입니다. AWS Glue 크롤러 및 Spark를 사용한 AWS Glue 작업과 같은 AWS Glue 구성 요소가 데이터 카탈로그에 기록하면 내부 유형 시스템을 사용하여 필드 유형을 추적합니다. 이러한 값은 AWS Glue 콘솔에 있는 테이블 스키마의 데이터 유형 열에 표시됩니다. 이 유형 시스템은 Apache Hive의 유형 시스템을 기반으로 합니다. Apache Hive 유형 시스템에 대한 자세한 내용은 Apache Hive 위키의 유형을 참조하세요. 특정 유형 및 지원에 대한 자세한 내용은 스키마 빌더의 일부로 AWS Glue Console에서 예제를 참조하세요.

검증, 호환성 및 기타 용도

데이터 카탈로그는 유형 필드에 기록된 유형을 검증하지 않습니다. AWS Glue 구성 요소는 데이터 카탈로그를 읽고 쓸 때 서로 호환됩니다. AWS Glue 구성 요소는 또한 Hive 유형과의 높은 호환성을 유지하는 것을 목표로 합니다. 그러나 AWS Glue 구성 요소가 모든 Hive 유형과의 호환성을 보장하지는 않습니다. 이를 통해 데이터 카탈로그의 테이블로 작업할 때 Athena DDL과 같은 도구와의 상호 운용성이 가능합니다.

데이터 카탈로그는 유형을 검증하지 않으므로 다른 서비스에서는 데이터 카탈로그를 사용하여 Hive 유형 시스템 또는 기타 시스템을 엄격하게 준수하는 시스템을 사용하여 유형을 추적할 수 있습니다.

Spark 스크립트가 포함된 AWS Glue의 유형

Spark 스크립트가 포함된 AWS Glue가 데이터세트를 해석하거나 변환할 때 스크립트에서 사용된 데이터세트의 인 메모리 표현인 DynamicFrame을 제공합니다. DynamicFrame의 목표는 Spark DataFrame의 목표와 유사합니다. Spark가 데이터에 대한 변환을 예약하고 실행할 수 있도록 데이터세트를 모델링하는 것입니다. toDF 및 fromDF 메서드를 제공하여 DynamicFrame의 유형 표현이 DataFrame과 상호 호환되도록 보장합니다.

유형 정보가 유추되거나 DataFrame에 제공될 수 있는 경우 달리 문서화되지 않는 한 이를 유추하거나DynamicFrame에 제공할 수 있습니다. 특정 데이터 형식에 최적화된 리더 또는 라이터를 제공하는 경우, Spark가 사용자의 데이터를 읽거나 쓸 수 있는 경우 제공된 리더 및 라이터는 문서화된 제한에 따라 데이터를 읽거나 쓸 수 있습니다. 리더 및 라이터에 관한 자세한 내용은 the section called “데이터 포맷 옵션” 섹션을 참조하세요.

선택 유형

DynamicFrames는 디스크에서 행 간에 유형이 일치하지 않을 수 있는 값인 데이터 세트의 필드를 모델링하는 메커니즘을 제공합니다. 예를 들어 필드는 특정 행에는 문자열로 저장된 숫자를, 다른 행에는 정수를 포함할 수 있습니다. 이 메커니즘은 Choice라고 하는 인 메모리 유형입니다. 선택 열을 구체적인 유형으로 해결하기 위한 ResolveChoice 메서드와 같은 변환을 제공합니다. AWS Glue ETL은 일반적인 작업 과정에서 선택 유형을 데이터 카탈로그에 기록하지 않습니다. 선택 유형은 데이터 세트의 DynamicFrame 메모리 모델 컨텍스트에서만 존재합니다. 선택 유형 사용 예는 the section called “데이터 준비 예” 섹션을 참조하세요.

AWS Glue 크롤러 유형

크롤러는 데이터세트에 대해 일관되고 사용 가능한 스키마를 생성한 다음 다른 AWS Glue 구성 요소 및 Athena에서 사용할 수 있도록 데이터 카탈로그에 저장하는 것을 목표로 합니다. 크롤러는 데이터 카탈로그의 이전 섹션 the section called “AWS Glue 데이터 카탈로그 유형”에서 정의된 유형을 다룹니다. 열에 두 개 이상의 유형 값이 포함된 '선택' 유형 시나리오에서 사용 가능한 유형을 생성하기 위해 크롤러는 잠재적 유형을 모델링하는 struct 유형을 만듭니다.

AWS Glue 시작하기

다음 섹션에서는 AWS Glue 설정에 대한 정보를 제공합니다. AWS Glue 사용을 시작하기 위해 모든 설정 섹션이 필요한 것은 아닙니다. VPC 환경을 사용하여 데이터 스토어에 액세스하거나 대화형 세션을 사용하는 경우 IAM 권한, 암호화, DNS를 설정하는 데 필요한 지침을 사용할 수 있습니다.

주제

- AWS Glue 사용 개요
- AWS Glue에 대한 IAM 권한 설정
- AWS Glue 사용 프로필 설정
- AWS Glue Data Catalog 시작하기
- 데이터 스토어에 대한 네트워크 액세스 설정
- AWS Glue에서 암호화 설정
- AWS Glue의 개발에 대한 네트워킹 설정

AWS Glue 사용 개요

AWS Glue로 AWS Glue Data Catalog에 메타데이터를 저장합니다. 이 메타데이터를 사용하여 데이터 원본을 변환하고 데이터 웨어하우스 또는 데이터 레이크를 로드하는 ETL 작업을 오케스트레이션할 수 있습니다. 다음 단계는 일반 작업 흐름과 AWS Glue와 작업할 때 정한 선택에 대해 설명합니다.

> **Note**
>
> 다음 단계를 사용하거나 1~3단계를 자동으로 수행하는 워크플로를 생성할 수 있습니다. 자세한 내용은 the section called “블루프린트 및 워크플로를 사용하여 복잡한 ETL 활동 수행” 단원을 참조하십시오.

1. 테이블 정의로 AWS Glue Data Catalog를 채웁니다.

   콘솔에서 영구 데이터 스토어의 경우 크롤러를 추가하여 AWS Glue Data Catalog를 채울 수 있습니다. 테이블 목록 또는 크롤러 목록에서 [Add crawler(크롤러 추가)] 마법사를 시작합니다. 크롤러에 액세스하도록 하나 이상의 스토어를 선택합니다. 일정을 생성하여 크롤러 실행 빈도수를 결정합니다. 데이터 스트림의 경우 테이블 정의를 수동으로 생성하고 스트림 속성을 정의할 수 있습니다.

데이터 스키마를 추론하는 사용자 분류자를 제공할 수도 있습니다. grok 패턴을 사용하여 사용자 지정 분류자를 생성합니다. 하지만 AWS Glue는 사용자 분류자가 데이터를 인식하지 않으면 크롤러에서 자동적으로 사용된 기본 설정 분류자를 제공합니다. 크롤러를 정의할 때 분류자를 선택할 필요가 없습니다. AWS Glue의 분류자에 대한 자세한 내용은 분류자 정의 및 관리 단원을 참조하십시오.

몇 가지 데이터 스토어 유형을 크롤하려면 인증 및 위치 정보가 제공되는 연결이 필요합니다. 필요하면 AWS Glue 콘솔에 필요한 정보를 제공하는 연결을 생성할 수 있습니다.

크롤러가 데이터 스토어를 읽고 데이터 정의와 이름 붙여진 테이블을 AWS Glue Data Catalog에 생성합니다. 이런 테이블은 선택에 따라 데이터베이스에 조직됩니다. 수동으로 생성한 테이블로 Data Catalog를 채울 수도 있습니다. 이러 방법을 통해 스키마 및 기타 메타데이터를 제공하여 Data Catalog에 테이블 정의를 생성합니다. 이런 방법은 조금 짜증나고 오류가 쉽게 날 수 있기 때문에 크롤러가 테이블 정의를 생성하는 방법이 더 좋습니다.

테이블 정의로 AWS Glue Data Catalog을 채우는 방법에 대한 자세한 내용은 테이블 생성 단원을 참조하십시오.

2. 원본에서 대상으로의 데이터 변환을 설명하는 작업을 정의합니다.

   일반적으로 다음과 같은 선택을 통해 작업을 생성합니다.

   - AWS Glue Data Catalog에서 작업의 소스로 사용할 테이블을 선택합니다. 작업은 이 테이블 정의를 사용하여 데이터 원본에 액세스하고 데이터 포맷을 해석합니다.
   - AWS Glue Data Catalog에서 작업의 대상으로 사용할 테이블 또는 위치를 선택합니다. 작업은 이 정보를 사용하여 데이터 스토어에 액세스합니다.
   - AWS Glue에 스크립트를 생성하여 소스를 대상으로 변환하도록 지시합니다. AWS Glue는 코드를 생성하여 기본 제공 변환을 직접 호출한 다음 소스 스키마에서 대상 스키마 형식으로 데이터를 전환합니다. 이 변환은 데이터 복사, 열 이름 바꾸기 및 데이터 필터링 후 필요에 따라 데이터 변환의 작업을 실행합니다. AWS Glue 콘솔의 스크립트를 수정할 수 있습니다.

   AWS Glue의 작업을 정의하는 방법에 대한 자세한 내용은 AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 단원을 참조하십시오.

3. 작업을 실행하여 데이터를 변환합니다.

   필요할 때 작업하거나, 트리거 유형 중 하나에 따라 시작할 수 있습니다.

   - Cron 일정에 따른 트리거.
   - 트리거는 이벤트 기반입니다. 예를 들어, 작업을 성공적으로 완료했다면 AWS Glue 작업을 시작할 수 있습니다.

- 필요 시 트리거는 작업을 시작합니다.

  AWS Glue의 트리거에 대한 자세한 내용은 트리거를 사용하여 작업 및 크롤러 시작 단원을 참조하십시오.

4. 예정된 크롤러와 촉발된 작업을 모니터합니다.

   AWS Glue 콘솔을 사용하여 다음을 봅니다.

   - 작업 실행 상세 정보 및 오류
   - 크롤러 실행 상세 정보 및 오류
   - AWS Glue 동작에 대한 알림

   AWS Glue의 크롤러 및 작업 모니터링에 대한 자세한 내용은 AWS Glue 모니터링 단원을 참조하십시오.

AWS Glue에 대한 IAM 권한 설정

이 주제의 지침은 AWS Glue에 대해 AWS Identity and Access Management(IAM) 권한을 빠르게 설정하는 데 도움이 됩니다. 다음 작업을 완료합니다.

- IAM 자격 증명에 AWS Glue 리소스에 대한 액세스 권한을 부여합니다.
- 작업 실행, 데이터 액세스, AWS Glue Data Quality 작업 실행을 위한 서비스 역할을 생성합니다.

AWS Glue에 대한 IAM 권한을 사용자 지정하는 데 사용할 수 있는 자세한 지침은 AWS Glue에 대한 IAM 권한 구성 섹션을 참조하세요.

AWS Management Console에서 AWS Glue에 대한 IAM 권한을 설정하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 시작하기를 선택합니다.
3. AWS Glue에 대한 계정 준비에서 IAM 권한 설정을 선택합니다.
4. AWS Glue 권한을 부여하려는 IAM 자격 증명(역할 또는 사용자)을 선택합니다. AWS Glue에서는 이 자격 증명에 `AWSGlueConsoleFullAccess` 관리형 정책을 연결합니다. 이러한 권한을 수동으로 설정하거나 기본 서비스 역할만 설정하려는 경우 이 단계를 건너뛸 수 있습니다.
5. 다음을 선택합니다.

6. 역할 및 사용자에게 필요한 Amazon S3 액세스 수준을 선택합니다. 이 단계에서 선택한 옵션은 선택한 모든 자격 증명에 적용됩니다.

   a. S3 위치 선택에서 액세스 권한을 부여하려는 Amazon S3 위치를 선택합니다.

   b. 다음으로, 이전에 선택한 위치에 대해 자격 증명에 읽기 전용(권장) 또는 읽기 및 쓰기 액세스 권한을 부여할지 선택합니다. AWS Glue에서는 선택한 위치 및 읽기 또는 쓰기 권한의 조합을 기반으로 자격 증명에 권한 정책을 추가합니다.

      다음 테이블에는 Amazon S3 액세스를 위해 AWS Glue에서 연결하는 권한이 나와 있습니다.

사용자 선택 항목...	AWS Glue에서 연결하는 항목...
변경 없음	해당 권한이 없습니다. AWS Glue에서는 자격 증명 권한은 변경하지 않습니다.

사용자 선택 항목...	AWS Glue에서 연결하는 항목...

특정 Amazon S3 위치에 대한 액세스 권한 부여(읽기 전용)	고객 관리형 정책(AWSGlueConsole-S3-read-only-policy)은 읽기 전용 권한이 있는 특정 Amazon S3 위치에 대한 액세스 권한을 부여합니다.
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"s3:GetObject"
	],
	"Resource": [
	"arn:aws:
	s3:::amzn-s3-demo-bucket1/*",
	"arn:aws:
	s3:::amzn-s3-demo-bucket2/*",
	"arn:aws:
	s3:::amzn-s3-demo-bucket3",
	"arn:aws:
	s3:::amzn-s3-demo-bucket"
	],
	"Condition": {
	"StringEquals": {
	"aws:Reso
	urceAccount":"0000000000"
	}
	}
	}
	]
	}

사용자 선택 항목...	AWS Glue에서 연결하는 항목...

특정 Amazon S3 위치에 대한 액세스 권한 부여(읽기 및 쓰기)	`AWSGlueConsole-S3-read-and-write-policy` 는 특정 Amazon S3 위치에 대한 읽기 및 쓰기 권한과 함께 액세스 권한을 부여합니다.
	
	{
	"Version": "2012-10-17",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"s3:GetObject",
	"s3:PutObject"
	],
	"Resource": [
	"arn:aws:
	s3:::aes-siem-00000000000-log/
	*",
	"arn:aws:
	s3:::aes-siem-00000000000-s
	napshot/*",
	"arn:aws:
	s3:::aes-siem-00000000000-l
	og",
	"arn:aws:
	s3:::aes-siem-00000000000-s
	napshot"
	],
	"Condition": {
	"StringEquals": {
	"aws:Reso
	urceAccount":"00000000000"
	}
	}
	}
	]
	}

7. 다음을 선택합니다.

8. 사용자 계정의 기본 AWS Glue 서비스 역할을 선택합니다. 서비스 역할은 사용자를 대신하여 다른 AWS 서비스의 리소스에 액세스하기 위해 AWS Glue에서 사용하는 IAM 역할입니다. 자세한 내용은 AWS Glue의 서비스 역할 섹션을 참조하세요.

   - 표준 AWS Glue 서비스 역할을 선택하면 AWS Glue는 AWS 계정에서 이름이 `AWSGlueServiceRole`이고 다음 관리형 정책이 연결된 새 IAM 역할을 생성합니다. 사용자 계정에 이름이 `AWSGlueServiceRole`인 IAM 역할이 이미 있는 경우 AWS Glue는 이러한 정책을 기존 역할에 연결합니다.

     - AWSGlueServiceRole - 이 관리형 정책은 AWS Glue에서 사용자를 대신하여 리소스에 액세스하고 관리하는 데 필요합니다. 이를 통해 AWS Glue 작업, 크롤러 및 연결과 같은 다양한 리소스를 AWS Glue에서 생성, 업데이트 및 삭제할 수 있습니다. 또한 이 정책은 로깅 목적으로 Amazon CloudWatch 로그에 액세스할 수 있는 권한을 AWS Glue에 부여합니다. 시작하기 위해 AWS Glue를 사용하는 방법을 알아보도록 이 정책을 사용하는 것이 좋습니다. AWS Glue에 더 익숙해지면 필요에 따라 리소스에 대한 액세스를 미세 조정할 수 있는 정책을 생성할 수 있습니다.

     - AWSGlueConsoleFullAccess - 이 관리형 정책은 AWS Management Console을 통해 AWS Glue 서비스에 대한 전체 액세스 권한을 부여합니다. 이 정책은 AWS Glue 내에서 모든 작업을 수행할 수 있는 권한을 부여하므로 필요에 따라 모든 AWS Glue 리소스를 생성, 수정 및 삭제할 수 있습니다. 그러나 이 정책은 ETL 프로세스에 관여할 수 있는 기본 데이터 스토어 또는 기타 AWS 서비스에 액세스할 수 있는 권한을 부여하지 않는다는 점에 유의해야 합니다. `AWSGlueConsoleFullAccess` 정책은 광범위한 권한을 부여하므로 최소 권한 원칙에 따라 신중하게 할당해야 합니다. 가능하다면 특정 사용 사례와 요구 사항에 맞게 더욱 세부적인 정책을 만들고 사용하는 것이 좋습니다.

     - AWSGlueConsole-S3-read-only-policy – 이 정책은 AWS Glue가 지정된 Amazon S3 버킷에서 데이터를 읽을 수 있도록 허용하지만 Amazon S3에서 데이터를 쓰거나 수정할 수 있는 권한은 부여하지 않습니다.

       AWSGlueConsole-S3-read-and-write – 이 정책은 AWS Glue가 ETL 프로세스의 일부로 지정된 Amazon S3 버킷에 데이터를 읽고 쓸 수 있도록 허용합니다.

   - 기존 IAM 역할을 선택하면 AWS Glue는 역할을 기본값으로 설정하지만 역할에 `AWSGlueServiceRole` 권한을 추가하지는 않습니다. AWS Glue에 대한 서비스 역할로 사용하도록 역할을 구성했는지 확인합니다. 자세한 내용은 1단계: AWS Glue 서비스를 위한 IAM 정책 생성 및 2단계: AWS Glue에 대한 IAM 역할 생성(을)를 참조하세요.

9. 다음을 선택합니다.

10. 마지막으로, 선택한 권한을 검토하고 변경 사항 적용을 선택합니다. 변경 사항을 적용하면 AWS Glue는 선택한 자격 증명에 IAM 권한을 추가합니다. 새 권한은 IAM 콘솔(https://console.aws.amazon.com/iam/)에서 확인하거나 수정할 수 있습니다.

이제 AWS Glue에 대한 최소 IAM 권한 설정을 완료했습니다. 프로덕션 환경에서는 사용 사례에 사용할 AWS 리소스를 확보할 수 있도록 AWS Glue의 보안 및 AWS Glue의 Identity and Access Management에 익숙해지는 것이 좋습니다.

다음 단계

이제 IAM 권한을 설정했으므로 AWS Glue 사용을 시작하기 위해 다음 주제를 탐색해볼 수 있습니다.

- Getting Started with AWS Glue in AWS Skill Builder
- AWS Glue Data Catalog 시작하기

AWS Glue Studio에 대한 설정

시각적 ETL에 대한 AWS Glue을(를) 처음 사용하는 경우 이 섹션의 작업을 완료합니다.

주제

- AWS Glue Studio 사용자에게 필요한 IAM 권한 검토
- ETL 작업에 필요한 IAM 권한 검토
- AWS Glue Studio에 대한 IAM 권한 설정
- ETL 작업에 사용할 VPC 구성

AWS Glue Studio 사용자에게 필요한 IAM 권한 검토

AWS Glue Studio를 사용하려면 사용자가 다양한 AWS 리소스에 액세스할 수 있어야 합니다. 사용자는 Amazon S3 버킷, IAM 정책 및 역할, AWS Glue Data Catalog 객체를 보고 선택할 수 있어야 합니다.

AWS Glue 서비스 권한

AWS Glue Studio는 AWS Glue 서비스의 작업과 리소스를 사용합니다. AWS Glue Studio를 효과적으로 사용하려면 사용자에게 이러한 작업과 리소스에 대한 권한이 필요합니다. AWS Glue Studio 사용자에게 `AWSGlueConsoleFullAccess` 관리형 정책을 부여하거나 더 작은 권한 집합으로 사용자 지정 정책을 생성할 수 있습니다.

> **Important**
>
> 보안 모범 사례에 따라 Amazon S3 버킷 및 Amazon CloudWatch 로그 그룹에 대한 액세스를 추가로 제한하는 정책을 강화하여 액세스를 제한하는 것이 좋습니다. Amazon S3 정책 예제는 IAM 정책 작성하기: Amazon S3 버킷으로의 액세스를 보장하는 방법을 참조하세요.

AWS Glue Studio에 대한 사용자 지정 IAM 정책 생성

AWS Glue Studio에 대한 더 작은 권한 집합을 포함하는 사용자 지정 정책을 생성할 수 있습니다. 이 정책에서는 객체 또는 작업의 하위 집합에 대한 권한을 부여할 수 있습니다. 사용자 지정 정책을 생성할 때는 다음 정보를 사용합니다.

AWS Glue Studio API를 사용하려면 IAM 권한 내의 작업 정책에 `glue:UseGlueStudio`를 포함합니다. `glue:UseGlueStudio`를 사용하면 시간이 지남에 따라 API에 더 많은 작업이 추가되는 경우에도 모든 AWS Glue Studio 작업을 수행할 수 있습니다.

AWS Glue에서 정의한 작업에 대한 자세한 내용은 AWS Glue에서 정의한 작업을 참조하세요.

데이터 준비 작성 작업

- SendRecipeAction
- GetRecipeAction

DAG(방향성 비순환 그래프) 작업

- CreateDag
- UpdateDag
- GetDag
- DeleteDag

작업

- SaveJob
- GetJob
- CreateJob

- DeleteJob
- GetJobs
- UpdateJob

작업 실행 옵션

- StartJobRun
- GetJobRuns
- BatchStopJobRun
- GetJobRun
- QueryJobRuns
- QueryJobs
- QueryJobRunsAggregated

스키마 작업

- GetSchema
- GetInferredSchema

데이터베이스 작업

- GetDatabases

계획 작업

- GetPlan

테이블 작업

- SearchTables
- GetTables
- GetTable

연결 작업

- CreateConnection
- DeleteConnection
- UpdateConnection
- GetConnections
- GetConnection

매핑 작업

- GetMapping

S3 프록시 작업

- ListBuckets
- ListObjectsV2
- GetBucketLocation

보안 구성 작업

- GetSecurityConfigurations

스크립트 작업

- CreateScript(AWS Glue에서 같은 이름의 API와 다름)

AWS Glue Studio API에 액세스

AWS Glue Studio에 액세스하려면 IAM 권한 내의 작업 정책 목록에 `glue:UseGlueStudio`를 추가합니다.

아래 예제에서는 `glue:UseGlueStudio`가 작업 정책에 포함되어 있지만 AWS Glue Studio API는 개별적으로 식별되지 않습니다. 그 이유는 `glue:UseGlueStudio`를 포함하면 IAM 권한 내의 개별 AWS Glue Studio API를 지정하지 않아도 내부 API에 대한 액세스 권한이 사용자에게 자동으로 부여되기 때문입니다.

이 예제에서 나열된 추가 작업 정책(예: `glue:SearchTables`)은 AWS Glue Studio API가 아니며, 따라서 필요에 따라 IAM 권한에 포함되어야 합니다. 부여할 Amazon S3 액세스 수준을 지정할 때

Amazon S3 프록시 작업을 포함할 수도 있습니다. 아래 예제 정책에서는 AWS Glue Studio를 열고, 시각적 작업을 생성하고 선택한 IAM 역할에 충분한 액세스 권한이 있는 경우 이를 저장/실행하는 액세스 권한을 제공합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "glue:UseGlueStudio",
                "iam:ListRoles",
                "iam:ListUsers",
                "iam:ListGroups",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "glue:SearchTables",
                "glue:GetConnections",
                "glue:GetJobs",
                "glue:GetTables",
                "glue:BatchStopJobRun",
                "glue:GetSecurityConfigurations",
                "glue:DeleteJob",
                "glue:GetDatabases",
                "glue:CreateConnection",
                "glue:GetSchema",
                "glue:GetTable",
                "glue:GetMapping",
                "glue:CreateJob",
                "glue:DeleteConnection",
                "glue:CreateScript",
                "glue:UpdateConnection",
                "glue:GetConnection",
                "glue:StartJobRun",
                "glue:GetJobRun",
                "glue:UpdateJob",
                "glue:GetPlan",
                "glue:GetJobRuns",
                "glue:GetTags",
                "glue:GetJob",
                "glue:QueryJobRuns",
```

```
                "glue:QueryJobs",
                "glue:QueryJobRunsAggregated",
                "glue:SendRecipeAction",
                "glue:GetRecipeAction"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

노트북 및 데이터 미리 보기 권한

데이터 미리 보기와 노트북을 사용하면 작업을 실행하지 않고도 작업의 모든 스테이지(읽기, 변환, 쓰기)에서 데이터 샘플을 볼 수 있습니다. 데이터에 액세스할 때 사용할 AWS Glue Studio에 대한 AWS Identity and Access Management(IAM) 역할을 지정합니다. IAM 역할은 수임 가능하도록 설계되었으며 암호 또는 액세스 키와 같은 표준 장기 자격 증명이 연결되어 있지 않습니다. 대신, AWS Glue Studio가 역할을 수임할 때 IAM은 임시 보안 자격 증명을 제공합니다.

데이터 미리 보기와 노트북 명령이 올바르게 작동하려면 이름이 `AWSGlueServiceRole` 문자열로 시작하는 역할을 사용합니다. 역할에 다른 이름을 사용하려는 경우 `iam:passrole` 권한을 추가하고 IAM에서 해당 역할에 대한 정책을 구성해야 합니다. 자세한 내용은 역할에 대해 'AWSGlueServiceRole*'이라는 이름이 아닌 IAM 정책 생성 단원을 참조하십시오.

> **Warning**
>
> 역할이 노트북에 대한 `iam:passrole` 권한을 부여하고 역할 체인을 구현하는 경우 사용자가 의도하지 않게 노트북에 액세스하게 될 수 있습니다. 노트북에 대한 액세스 권한이 부여된 사용자를 모니터링할 수 있는 감사는 현재 구현되어 있지 않습니다.

IAM ID가 데이터 미리 보기 세션을 생성하지 못하게 하려면 the section called “ID가 데이터 미리 보기 세션을 생성하지 못하게 하기” 섹션의 다음 예제를 참조하세요.

Amazon CloudWatch 권한

AWS Glue에서 원시 데이터를 수집한 후 판독이 가능한 지표로 실시간에 가깝게 처리하는 Amazon CloudWatch를 사용하여 AWS Glue Studio 작업을 모니터링할 수 있습니다. 기본적으로 AWS Glue 지표 데이터는 CloudWatch에 자동으로 전송됩니다. 자세한 내용은 Amazon CloudWatch User Guide의 What Is Amazon CloudWatch?와 AWS Glue Developer Guide의 AWS Glue Metrics를 참조하세요.

CloudWatch 대시보드에 액세스하려면 AWS Glue Studio에 액세스하는 사용자에게 다음 중 하나가 필요합니다.

- `AdministratorAccess` 정책
- `CloudWatchFullAccess` 정책
- 다음과 같은 특정 권한 중 하나 이상을 포함하는 사용자 지정 정책:
  - 대시보드 보기를 위한 `cloudwatch:GetDashboard` 및 `cloudwatch:ListDashboards`
  - 대시보드를 생성하거나 수정하는 `cloudwatch:PutDashboard`
  - 대시보드를 삭제하는 `cloudwatch:DeleteDashboards`

정책을 사용하여 IAM 사용자의 권한 변경에 대한 자세한 내용은 IAM User Guide의 Changing Permissions for an IAM User를 참조하세요.

ETL 작업에 필요한 IAM 권한 검토

를 사용하여 작업을 생성할 때 AWS Glue Studio, 작업은 생성할 때 지정하는 IAM 역할의 권한을 맡습니다. 이 IAM 역할에는 데이터 소스에서 데이터를 추출하고, 대상에 데이터를 쓰고, AWS Glue 리소스에 액세스할 수 있는 권한이 있어야 합니다.

작업에 대해 생성한 역할의 이름은 문자열로 시작해야 에서 올바르게 사용할 `AWSGlueServiceRole` 수 있습니다.AWS Glue Studio. 예를 들어 역할 이름을 로 지정할 수 있습니다`AWSGlueServiceRole-FlightDataJob`.

데이터 원본 및 데이터 대상 권한

원래 요청 ping에 대한 AWS Glue Studio 작업에 사용하는 모든 소스, 대상, 스크립트 및 임시 디렉터리에 대해 Amazon S3에 액세스할 수 있어야 합니다. 특정 Amazon S3 리소스에 대한 세분화된 액세스를 제공하는 정책을 생성할 수 있습니다.

- 데이터 원본은 `s3:ListBucket` 및 `s3:GetObject` 권한을 요구합니다.
- 데이터 대상은 `s3:ListBucket`, `s3:PutObject` 및 `s3:DeleteObject` 권한을 요구합니다.

> **Note**
>
> IAM 정책은 AWS Glue 변환 호스팅`s3:GetObject`에 사용되는 특정 버킷을 허용해야 합니다. 다음 버킷은 AWS 서비스 계정에서 소유하며 전 세계적으로 읽을 수 있습니다. 이러한 버킷은 AWS Glue Studio 시각적 편집기를 통해 액세스할 수 있는 변환의 하위 집합과 관련된 소스 코드의 리포지토리 역할을 합니다. 버킷에 대한 권한은 버킷에 대한 다른 API 작업을 거부하도록 설정됩니다. 변환을 위해 제공하는 스크립트는 누구나 읽을 수 있지만, 서비스 팀 외부의 누구도 스크립트에 내용을 “입력”할 수 없습니다. AWS Glue 작업이 실행되면 파일이 로컬 컨테이너로 다운로드되도록 해당 파일을 로컬 가져오기로 가져옵니다. 이후에는 해당 계정과 더 이상 통신할 수 없습니다.

지역: 버킷 이름

- af-south-1: aws-glue-studio-transforms-762339736633-prod-af-south-1
- ap-east-1: aws-glue-studio-transforms-125979764932-prod-ap-east-1
- ap-northeast-2: aws-glue-studio-transforms-673535381443-prod-ap-northeast-2
- ap-northeast-3: aws-glue-studio-transforms-149976050262-prod-ap-northeast-3
- ap-south-1: aws-glue-studio-transforms-584702181950-prod-ap-south-1
- ap-south-2: aws-glue-studio-transforms-380279651983-prod-ap-south-2
- ap-southeast-1: aws-glue-studio-transforms-737106620487-prod-ap-southeast-1
- ap-southeast-2: aws-glue-studio-transforms-234881715811-prod-ap-southeast-2
- ap-southeast-3: aws-glue-studio-transforms-151265630221-prod-ap-southeast-3

- ap-southeast-4: aws-glue-studio-transforms-052235663858-prod-ap-southeast-4
- ca-central-1: aws-glue-studio-transforms-622716468547-prod-ca-central-1
- ca-west-1: aws-glue-studio-transforms-915795495192-prod-ca-west-1
- eu-central-1: aws-glue-studio-transforms-560373232017-prod-eu-central-1
- eu-central-2: aws-glue-studio-transforms-907358657121-prod-eu-central-2
- eu-north-1: aws-glue-studio-transforms-312557305497-prod-eu-north-1
- eu-south-1: aws-glue-studio-transforms-939684186351-prod-eu-south-1
- eu-south-2: aws-glue-studio-transforms-239737454084-prod-eu-south-2
- eu-west-1: aws-glue-studio-transforms-244479516193-prod-eu-west-1
- eu-west-2: aws-glue-studio-transforms-804222392271-prod-eu-west-2
- eu-west-3: aws-glue-studio-transforms-371299348807-prod-eu-west-3
- il-central-1: aws-glue-studio-transforms-806964611811-prod-il-central-1
- me-central-1: aws-glue-studio-transforms-733304270342-prod-me-central-1
- me-south-1: aws-glue-studio-transforms-112120182341-prod-me-south-1
- sa-east-1: aws-glue-studio-transforms-881619130292-prod-sa-east-1
- us-east-1: aws-glue-studio-transforms-510798373988-prod-us-east-1
- us-east-2: aws-glue-studio-transforms-251189692203-prod-us-east-2
- us-west-1: aws-glue-studio-transforms-593230150239-prod-us-west-1
- us-west-2: aws-glue-studio-transforms-818035625594-prod-us-west-2
- ap-northeast-1: aws-glue-studio-transforms-200493242866-prod-ap-northeast-1
- cn-north-1: aws-glue-studio-transforms-071033555442-prod-cn-north-1
- cn-northwest-1: aws-glue-studio-transforms-070947029561-prod-cn-northwest-1
- us-gov-west-1: aws-glue-studio-transforms-227493901923-prod-us-gov-west-1-2604

데이터 소스 Amazon Redshift 로 를 선택한 경우 클러스터 권한에 대한 역할을 제공할 수 있습니다. Amazon Redshift 클러스터에 대해 실행되는 작업은 임시 자격 증명을 사용하여 임시 스토리지를 위해 Amazon S3에 액세스하는 명령을 내리게 됩니다. 작업이 1시간 이상 실행되면 이러한 자격 증명이 만료되어 작업이 실패합니다. 이 문제를 방지하기 위해 임시 자격 증명을 사용하여 작업에 필요한 권한을 부여하는 역할을 Amazon Redshift 클러스터 자체에 할당할 수 있습니다. 자세한 내용은 AWS Glue Developer Guide의 Moving Data to and from Amazon Redshift를 참조하세요.

작업이 Amazon S3 이외의 데이터 소스 또는 대상을 사용하는 경우 이러한 데이터 소스 및 대상에 액세스하는 데 작업에서 사용하는 IAM 역할에 필요한 권한을 연결해야 합니다. 자세한 내용은 AWS Glue Developer Guide의 Setting Up Your Environment to Access Data Stores를 참조하세요.

데이터 스토어에 커넥터 및 연결을 사용하는 경우 the section called “커넥터 사용에 필요한 권한”에 설명된 대로 추가 권한이 필요합니다.

작업 삭제에 필요한 권한

In AWS Glue Studio 콘솔에서 삭제할 작업을 여러 개 선택할 수 있습니다. 이 작업을 수행하려면 `glue:BatchDeleteJob` 권한이 있어야 합니다. 이는 와 다릅니다.AWS Glue 콘솔 - 작업을 삭제할 수 있는 `glue:DeleteJob` 권한이 필요합니다.

AWS Key Management Service 권한

AWS Key Management Service (AWS KMS)를 사용하여 서버 측 암호화를 사용하는 Amazon S3 소스 및 대상에 액세스하려면 정책을 에 연결합니다.AWS Glue Studio 작업이 데이터를 복호화할 수 있도록 하는 작업에서 사용하는 역할입니다. 작업 역할에는 `kms:ReEncrypt`, `kms:GenerateDataKey` 및 `kms:DescribeKey` 권한이 필요합니다. 또한 작업 역할에는 AWS KMS 고객 마스터 키()로 암호화된 Amazon S3 객체를 업로드하거나 다운로드할 수 있는 `kms:Decrypt` 권한이 필요합니다CMK.

사용 시 추가 요금이 부과됩니다 AWS KMS CMKs. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS Key Management Service 개념 - 고객 마스터 키(CMKs) 및 AWS Key Management Service 요금을 참조하세요.

커넥터 사용에 필요한 권한

를 사용하는 경우 AWS Glue Custom Connector 및 데이터 스토어에 액세스하기 위한 연결, 를 실행하는 데 사용되는 역할 AWS Glue ETL 작업에 추가 권한이 연결되어야 합니다.

- 에서 구매한 커넥터에 액세스하기 `AmazonEC2ContainerRegistryReadOnly` 위한 AWS 관리형 정책입니다 AWS Marketplace.
- `glue:GetJob` 및 `glue:GetJobs` 권한.
- AWS Secrets Manager 연결에 사용되는 보안 암호에 액세스할 수 있는 권한입니다. 예제: 예제 정책의 보안 암호 값을 검색할 수 있는 권한을 참조하세요. IAM

가 AWS Glue ETL 작업은 실행 VPC 중인 Amazon 내에서 실행되며VPC, 그런 다음 에 설명된 대로 를 구성해야 VPC 합니다the section called “ETL 작업에 사용할 VPC 구성”.

AWS Glue Studio에 대한 IAM 권한 설정

AWS 관리자 사용자를 사용하여 역할을 생성하고 사용자 및 작업 역할에 정책을 할당할 수 있습니다.

AWSGlueConsoleFullAccess AWS 관리형 정책을 사용하여 AWS Glue Studio 콘솔 사용에 필요한 권한을 제공할 수 있습니다.

자체 정책을 생성하려면 AWS Glue Developer Guide의 Create an IAM Policy for the AWS Glue Service에 설명된 단계를 따릅니다. 이전에 AWS Glue Studio 사용자에게 필요한 IAM 권한 검토에서 설명한 IAM 권한을 포함합니다.

주제

- AWS Glue Studio 사용자에게 정책을 연결합니다.
- 역할에 대해 'AWSGlueServiceRole*'이라는 이름이 아닌 IAM 정책 생성

AWS Glue Studio 사용자에게 정책을 연결합니다.

AWS Glue Studio 콘솔에 로그인하는 모든 AWS 사용자에게 특정 리소스에 액세스할 수 있는 권한이 있어야 합니다. 사용자에게 IAM 정책을 할당하여 해당 권한을 제공합니다.

AWSGlueConsoleFullAccess 관리형 정책을 사용자에 연결

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 AWSGlueConsoleFullAccess 옆의 확인란을 선택합니다. [Filter] 메뉴와 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 조치를 선택한 후 연결을 선택합니다.
5. 정책을 연결하려는 사용자를 선택합니다. 필터 메뉴와 검색 상자를 사용하면 보안 주체 개체 목록을 필터링할 수 있습니다. 정책을 추가할 사용자를 선택한 다음 [Attach policy(정책 추가)]를 선택합니다.
6. 필요에 따라 이전 단계를 반복하여 사용자에게 추가 정책을 연결합니다.

역할에 대해 'AWSGlueServiceRole*'이라는 이름이 아닌 IAM 정책 생성

AWS Glue Studio에서 사용하는 역할에 대한 IAM 정책 구성

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 새 IAM 정책을 추가합니다. 기존 정책에 추가하거나 새 IAM 인라인 정책을 생성할 수 있습니다. IAM 정책 생성
   1. 정책을 선택한 후 정책 생성을 선택합니다. 시작 버튼이 표시되면 이 버튼을 선택한 다음 정책 생성을 선택합니다.
   2. [Create Your Own Policy] 옆의 [Select]를 선택합니다.
   3. 정책 이름에 나중에 쉽게 참조할 수 있는 값을 입력합니다. 선택적으로, 설명에 설명을 입력합니다.
   4. 정책 문서에 다음 형식의 정책 설명을 입력한 다음, 정책 생성을 선택합니다.
3. 다음 블록을 복사하여 'Statement' 배열 아래의 정책에 붙여넣습니다. 이때 *my-interactive-session-role-prefix*를 AWS Glue에 대한 권한과 연결할 모든 공통 역할의 접두사로 바꿉니다.

```
{
    "Action": [
        "iam:PassRole"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/my-interactive-session-role-prefix*",
    "Condition": {
        "StringLike": {
            "iam:PassedToService": [
                "glue.amazonaws.com "
            ]
        }
    }
}
```

다음은 정책에 포함된 Version 및 Statement 배열의 전체 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
        "Action": [
            "iam:PassRole"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iam::*:role/my-interactive-session-role-prefix*",
        "Condition": {
            "StringLike": {
                "iam:PassedToService": [
                    "glue.amazonaws.com "
                ]
            }
        }
    }
  ]
}
```

4. 사용자에게 정책을 활성화하려면 사용자를 선택합니다.
5. 정책을 연결하려는 사용자를 선택합니다.

ETL 작업에 사용할 VPC 구성

Amazon Virtual Private Cloud(Amazon VPC)를 사용하면 AWS 클라우드 내에서 논리적으로 격리된 자체 영역에 Virtual Private Cloud(VPC)라고 하는 가상 네트워크를 정의할 수 있습니다. 인스턴스와 같은 AWS 리소스를 VPC에서 시작할 수 있습니다. VPC는 고객의 자체 데이터 센터에서 운영하는 기존 네트워크와 매우 유사하지만 AWS의 확장 가능한 인프라를 사용한다는 이점을 제공합니다. 해당 IP 주소 범위를 선택하고, 서브넷을 만든 후 라우팅 테이블, 네트워크 게이트웨이 및 보안 설정을 구성하여 VPC를 구성할 수 있습니다. VPC의 인스턴스를 인터넷에 연결합니다. VPC를 사내 데이터 센터에 연결하여 AWS 클라우드에서 데이터 센터를 확장할 수 있습니다. 각의 서브넷에서 리소스를 보호하기 위해 보안 그룹 및 네트워크 액세스 제어 목록을 포함한 다중 보안 계층을 사용할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

커넥터를 사용할 때 VPC 내에서 실행되도록 AWS Glue ETL 작업을 구성할 수 있습니다. 필요에 따라 다음에 대해 VPC를 구성해야 합니다.

- AWS에 없는 데이터 스토어에 대한 퍼블릭 네트워크 액세스. 작업이 액세스한 모든 데이터 스토어는 VPC 서브넷에서 사용 가능해야 합니다.
- 작업에서 VPC 리소스와 퍼블릭 인터넷에 모두 액세스해야 할 경우 VPC 내부 네트워크 주소 변환(NAT) 게이트웨이가 VPC에 있어야 합니다.

자세한 내용은 AWS Glue Developer Guide의 Setting Up Your Environment to Access Data Stores를 참조하세요.

AWS Glue Studio에서 노트북 시작하기

AWS Glue Studio를 통해 노트북을 시작하는 경우 데이터를 탐색하고 몇 초 만에 작업 스크립트 개발을 시작할 수 있도록 모든 구성 단계가 완료됩니다.

다음 섹션에서는 ETL 작업에 대해 AWS Glue Studio에서 노트북을 사용하기 위해 역할을 생성하고 적절한 권한을 부여하는 방법을 설명합니다.

AWS Glue에서 정의한 작업에 대한 자세한 내용은 AWS Glue에서 정의한 작업을 참조하세요.

주제

- IAM 역할에 권한 부여

IAM 역할에 권한 부여

AWS Glue Studio 설정은 노트북을 사용하기 위한 전제 조건입니다.

AWS Glue에서 노트북을 사용하려면 역할에 다음이 필요합니다.

- `sts:AssumeRole` 작업(및 태그 지정을 원할 경우 `sts:TagSession` 작업)을 수행하기 위한 AWS Glue와의 신뢰 관계
- 노트북, AWS Glue 및 대화형 세션에 대한 모든 권한을 포함하는 IAM 정책.
- 전달 역할에 대한 IAM 정책(역할이 노트북에서 대화형 세션으로 전달될 수 있어야 하기 때문)

예를 들어 새 역할을 생성할 때 역할에 `AWSGlueConsoleFullAccessRole`과 같은 표준 AWS 관리형 정책을 추가한 후 노트북 작업에 대한 새 정책과 IAM PassRole 정책에 대한 다른 새 정책을 추가할 수 있습니다.

AWS Glue와의 신뢰 관계에 필요한 작업

노트북 세션을 시작할 때 노트북에 전달되는 역할의 신뢰 관계에 `sts:AssumeRole`을 추가해야 합니다. 세션에 태그가 포함되어 있는 경우 `sts:TagSession` 작업도 전달해야 합니다. 이러한 작업이 없으면 노트북 세션을 시작할 수 없습니다.

예:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

노트북의 IAM 권한을 포함하는 정책

다음 샘플 정책은 노트북에 대한 필수적인 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartNotebook",
                "glue:TerminateNotebook",
                "glue:GlueNotebookRefreshCredentials",
                "glue:DeregisterDataPreview",
                "glue:GetNotebookInstanceStatus",
                "glue:GlueNotebookAuthorize"
            ],
            "Resource": "*"
        }
    ]
}
```

다음 IAM 정책을 사용하여 특정 리소스에 대한 액세스를 허용할 수 있습니다.

- AwsGlueSessionUserRestrictedNotebookServiceRole: 세션을 제외한 모든 AWS Glue 리소스에 대한 전체 액세스 권한을 제공합니다. 사용자가 자신과 연결된 노트북 세션만 생성하고 사용할 수 있도

록 허용합니다. 이 정책에는 다른 AWS 서비스에서 AWS Glue 리소스를 관리하는 데 AWS Glue에서 필요한 기타 권한도 포함됩니다.

- AwsGlueSessionUserRestrictedNotebookPolicy: 사용자가 자신과 연결된 노트북 세션만 생성하고 사용할 수 있도록 허용하는 권한을 제공합니다. 이 정책에는 제한된 AWS Glue 세션 역할을 사용자가 전달할 수 있도록 명시적으로 허용하는 권한도 포함되어 있습니다.

역할을 전달하는 IAM 정책

역할이 있는 노트북을 생성하면 해당 역할이 대화형 세션으로 전달되므로 두 위치에서 동일한 역할을 사용할 수 있습니다. 따라서 `iam:PassRole` 권한은 역할의 정책의 일부여야 합니다.

다음 예제를 사용하여 역할에 대한 새 정책을 생성합니다. 계정 번호를 사용자의 고유한 역할 이름으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::090000000210:role/<role_name>"
        }
    ]
}
```

AWS Glue 사용 프로필 설정

클라우드 플랫폼 사용의 주요 이점 중 하나는 유연성입니다. 하지만 이렇게 컴퓨팅 리소스를 쉽게 생성할 경우 관리하지 않고 가드레일 없이 방치하면 클라우드 비용이 급증할 위험이 있습니다. 따라서 관리자는 높은 인프라 비용을 피하는 동시에 사용자가 불필요한 마찰 없이 작업할 수 있도록 균형을 맞춰야 합니다.

AWS Glue 사용 프로필을 통해 관리자는 개발자, 테스터, 제품 팀 등 계정 내의 다양한 사용자 계층에 대한 여러 프로필을 생성할 수 있습니다. 각 프로필은 다양한 유형의 사용자에게 할당할 수 있는 고유한 파라미터 세트입니다. 예를 들어 개발자는 더 많은 작업자가 필요할 수 있고 최대 작업자 수가 많을 수 있지만, 제품 팀의 경우 필요한 작업자 수가 더 적고 필요한 제한 시간 또는 유휴 제한 시간 값이 더 낮을 수 있습니다.

작업 및 작업 실행 동작 예제

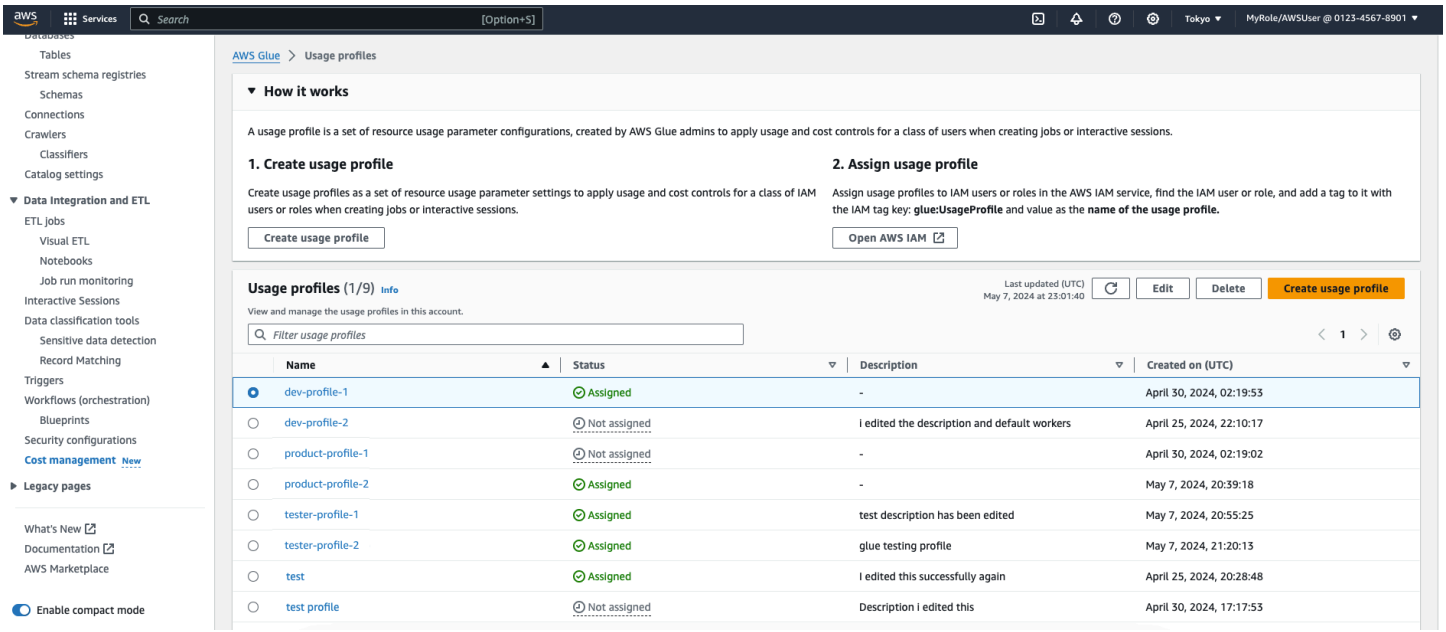
사용자 A가 프로필 A를 사용하여 작업을 생성한다고 가정합니다. 이때 작업은 특정 파라미터 값과 함께 저장됩니다. 프로필 B를 사용하는 사용자 B가 작업을 실행하려고 시도합니다.

사용자 A가 작업을 작성했을 때 특정 작업자 수를 설정하지 않은 경우 사용자 A의 프로필에 기본 설정이 적용되고 작업의 정의와 함께 저장됩니다.

사용자 B가 작업을 실행하면 저장된 값으로 실행됩니다. 사용자 B의 자체 프로필이 더 제한적이고 많은 작업자를 대상으로 실행할 수 없는 경우 작업 실행이 실패합니다.

리소스로의 사용 프로필

AWS Glue 사용 프로필은 Amazon 리소스 이름(ARN)으로 식별되는 리소스입니다. 작업 기반 및 리소스 기반 권한 부여를 포함하여 모든 기본 Identity and Access Management(IAM) 제어가 적용됩니다. 관리자는 AWS Glue 리소스를 생성하는 사용자의 IAM 정책을 업데이트하여 프로필을 사용할 수 있는 액세스 권한을 부여해야 합니다.



주제

- 사용 프로필 생성 및 관리
- 사용 프로필 및 작업

사용 프로필 생성 및 관리

AWS Glue 사용 프로필 생성

관리자는 사용 프로필을 생성한 다음 다양한 사용자에게 할당해야 합니다. 사용 프로필을 생성할 때에는 다양한 작업 및 세션 파라미터에 허용되는 값 범위뿐 아니라 기본값도 지정합니다. 작업 또는 대화형 세션에 대해 파라미터를 하나 이상 구성해야 합니다. 작업에 파라미터 값이 제공되지 않을 경우 사용할 기본값을 사용자 지정하거나, 이 프로필을 사용할 때 사용자가 파라미터 값을 제공하는 경우 검증을 하기 위한 범위 한도 또는 허용되는 값 세트를 설정할 수 있습니다.

기본값은 관리자가 작업 작성자를 지원하기 위해 설정하는 모범 사례입니다. 사용자가 새 작업을 생성하고 제한 시간 값을 설정하지 않으면 사용 프로필의 기본 제한 시간이 적용됩니다. 작성자에게 프로필이 없는 경우 AWS Glue 서비스 기본값이 적용되어 작업 정의에 저장됩니다. 런타임에 AWS Glue는 프로필에 설정된 한도(최소, 최대, 허용된 작업자)을 적용합니다.

파라미터가 구성되면 다른 모든 파라미터는 선택 사항이 됩니다. 작업 또는 대화형 세션에 대해 사용자 정의할 수 있는 파라미터는 다음과 같습니다.

- 작업자 수 - 컴퓨팅 리소스의 과도한 사용을 방지하기 위해 작업자 수를 제한합니다. 기본값, 최소값, 최대값을 설정할 수 있습니다. 최소값은 1입니다.
- 작업자 유형 - 워크로드의 관련 작업자 유형을 제한합니다. 사용자 프로필에 대해 기본 유형을 설정하고 작업자 유형을 허용할 수 있습니다.
- 제한 시간 - 작업 또는 대화형 세션이 종료되기 전에 실행 및 리소스를 소비할 수 있는 최대 시간을 정의합니다. 작업이 오래 실행되지 않도록 제한 시간 값을 설정합니다.

  기본값, 최소값, 최대값을 분 단위로 설정할 수 있습니다. 최소값은 1(분)입니다. AWS Glue 기본 제한 시간은 2,880분이지만 사용 프로필에서 원하는 기본값을 설정할 수 있습니다.

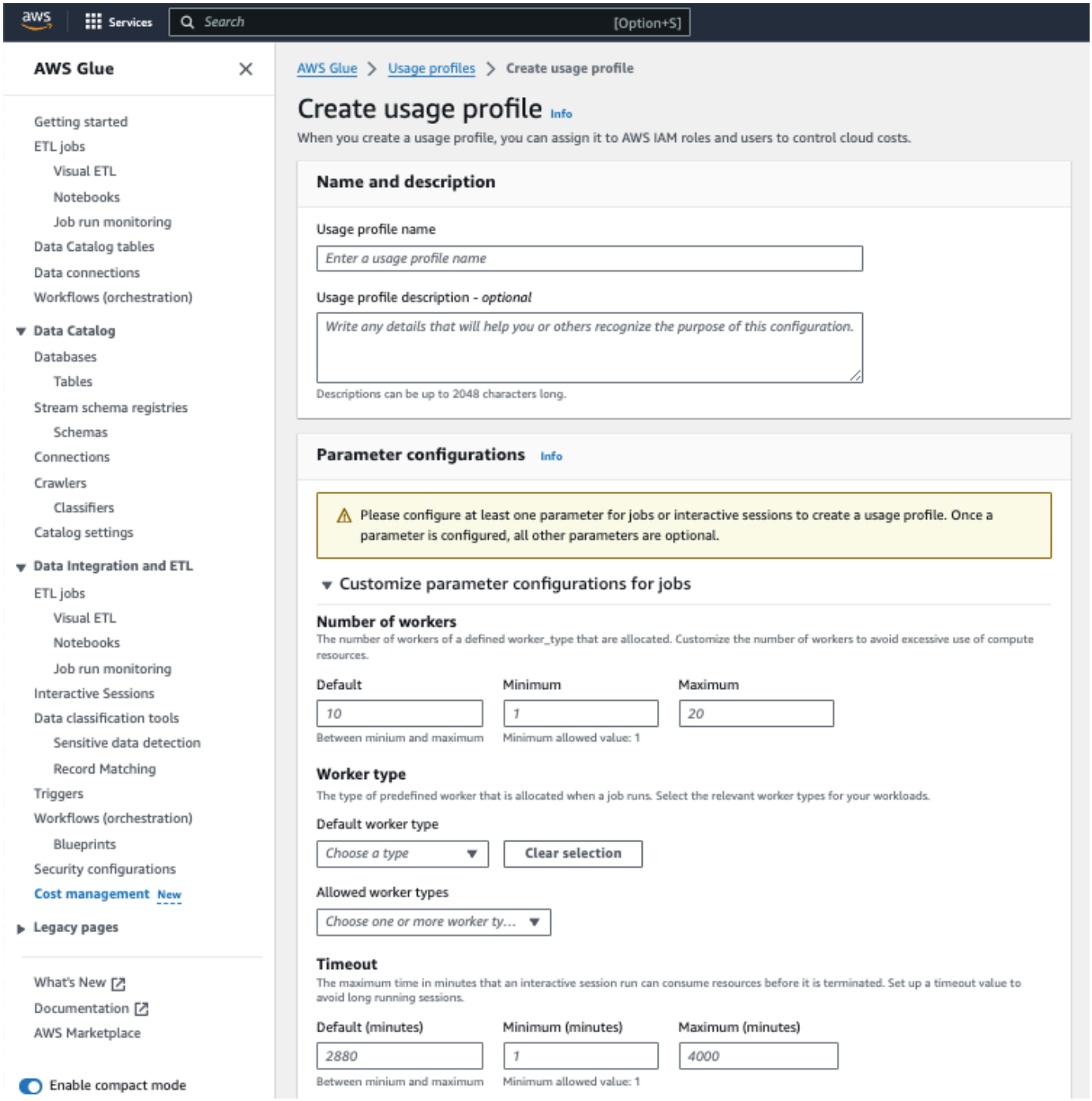
  '기본값'의 값을 설정하는 것이 좋습니다. 이 값은 사용자가 값을 설정하지 않은 경우에 작업 또는 세션 생성에 사용됩니다.
- 유휴 제한 시간 - 셀이 실행된 후 시간 초과되기 전까지 대화형 세션이 비활성 상태로 유지되는 시간(분)을 정의합니다. 작업이 완료된 후 종료할 대화형 세션의 유휴 제한 시간을 정의합니다. 유휴 제한 시간 범위는 제한 시간 내에 있어야 합니다.

  기본값, 최소값, 최대값을 분 단위로 설정할 수 있습니다. 최소값은 1(분)입니다. AWS Glue 기본 제한 시간은 2,880분이지만 사용 프로필에서 원하는 기본값을 설정할 수 있습니다.

  '기본값'의 값을 설정하는 것이 좋습니다. 이 값은 사용자가 값을 설정하지 않은 경우에 세션 생성에 사용됩니다.

관리자로 AWS Glue 사용 프로필을 생성하려면(콘솔)

1. 왼쪽 탐색 창에서 비용 관리를 선택합니다.
2. 사용 프로필 생성을 선택합니다.
3. 사용 프로필의 사용 프로필 이름을 입력합니다.
4. 다른 사람들이 사용 프로필의 용도를 쉽게 알 수 있도록 설명(선택 사항)을 입력합니다.
5. 프로필에 파라미터를 하나 이상 정의합니다. 양식의 모든 필드는 파라미터입니다. 세션 유휴 제한 시간 최소값을 예로 들 수 있습니다.
6. 사용 프로필에 적용되는 선택적 태그를 모두 정의합니다.
7. Save(저장)를 선택합니다.



사용 프로필을 생성하려면(AWS CLI)

1. 다음 명령을 입력합니다.

```
aws glue create-usage-profile --name profile-name --configuration file://
config.json --tags list-of-tags
```

여기서 config.json은 대화형 세션(SessionConfiguration) 및 작업(JobConfiguration)에 대한 파라미터 값을 정의할 수 있습니다.

```
//config.json (There is a separate blob for session/job configuration
{
    "SessionConfiguration": {
        "timeout": {
            "DefaultValue": "2880",
            "MinValue": "100",
            "MaxValue": "4000"
        },
        "idleTimeout": {
            "DefaultValue": "30",
            "MinValue": "10",
            "MaxValue": "4000"
        },
        "workerType": {
            "DefaultValue": "G.2X",
            "AllowedValues": [
                "G.2X",
                "G.4X",
                "G.8X"
            ]
        },
        "numberOfWorkers": {
            "DefaultValue": "10",
            "MinValue": "1",
            "MaxValue": "10"
        }
    },
    "JobConfiguration": {
        "timeout": {
            "DefaultValue": "2880",
            "MinValue": "100",
            "MaxValue": "4000"
        },
        "workerType": {
            "DefaultValue": "G.2X",
            "AllowedValues": [
                "G.2X",
                "G.4X",
                "G.8X"
```

```
            ]
        },
        "numberOfWorkers": {
            "DefaultValue": "10",
            "MinValue": "1",
            "MaxValue": "10"
        }
    }
}
```

2. 다음 명령을 입력하여 사용 프로필을 생성합니다.

```
aws glue get-usage-profile --name profile-name
```

응답은 다음과 같습니다.

```
{
    "ProfileName": "foo",
    "Configuration": {
        "SessionConfiguration": {
            "numberOfWorkers": {
                "DefaultValue": "10",
                "MinValue": "1",
                "MaxValue": "10"
            },
            "workerType": {
                "DefaultValue": "G.2X",
                "AllowedValues": [
                    "G.2X",
                    "G.4X",
                    "G.8X"
                ]
            },
            "timeout": {
                "DefaultValue": "2880",
                "MinValue": "100",
                "MaxValue": "4000"
            },
            "idleTimeout": {
                "DefaultValue": "30",
                "MinValue": "10",
                "MaxValue": "4000"
            }
```

```
        },
        "JobConfiguration": {
            "numberOfWorkers": {
                "DefaultValue": "10",
                "MinValue": "1",
                "MaxValue": "10"
            },
            "workerType": {
                "DefaultValue": "G.2X",
                "AllowedValues": [
                    "G.2X",
                    "G.4X",
                    "G.8X"
                ]
            },
            "timeout": {
                "DefaultValue": "2880",
                "MinValue": "100",
                "MaxValue": "4000"
            }
        }
    },
    "CreatedOn": "2024-01-19T23:15:24.542000+00:00"
}
```

사용 프로필을 관리하는 데 사용되는 추가 CLI 명령:

- aws glue list-usage-profiles
- aws glue update-usage-profile --name *profile-name* --configuration *file://config.json*
- aws glue delete-usage-profile --name *profile-name*

사용 프로필 편집

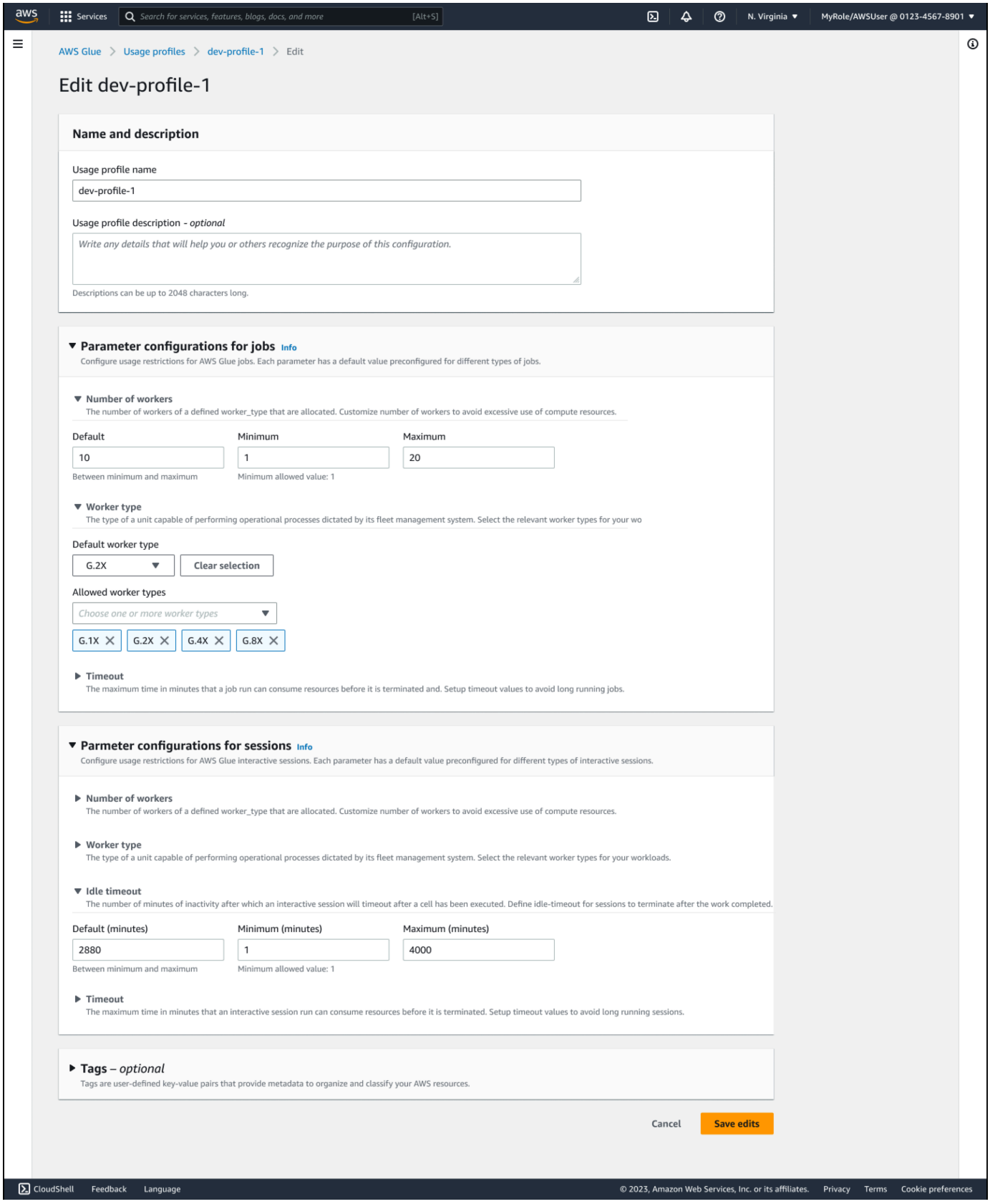
관리자는 자신이 생성한 사용 프로필을 편집하여 작업 및 대화형 세션의 프로필 파라미터 값을 변경할 수 있습니다.

사용 프로필을 편집하려면:

관리자로 AWS Glue 사용 프로필을 편집하려면(콘솔)

1. 왼쪽 탐색 창에서 비용 관리를 선택합니다.

2. 편집 권한이 있는 사용자 프로필을 선택하고 편집을 선택합니다.
3. 필요에 따라 프로필을 변경합니다. 기본적으로, 이미 값이 있는 파라미터가 확장됩니다.
4. 편집 내용 저장을 선택합니다.


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
MyRole/AWSUser @ 0123-4567-8901
AWS Glue > Usage profiles > dev-profile-1 > Edit
Edit dev-profile-1
Name and description
Usage profile name
dev-profile-1
Usage profile description - optional
Write any details that will help you or others recognize the purpose of this configuration.
Descriptions can be up to 2048 characters long.
Parameter configurations for jobs Info
Configure usage restrictions for AWS Glue jobs. Each parameter has a default value preconfigured for different types of jobs.
Number of workers
The number of workers of a defined worker_type that are allocated. Customize number of workers to avoid excessive use of compute resources.
Default
10
Between minimum and maximum
Minimum
1
Minimum allowed value: 1
Maximum
20
Worker type
The type of a unit capable of performing operational processes dictated by its fleet management system. Select the relevant worker types for your wo
Default worker type
G.2X
Clear selection
Allowed worker types
Choose one or more worker types
G.1X
G.2X
G.4X
G.8X
Timeout
The maximum time in minutes that a job run can consume resources before it is terminated and. Setup timeout values to avoid long running jobs.
Parmeter configurations for sessions Info
Configure usage restrictions for AWS Glue interactive sessions. Each parameter has a default value preconfigured for different types of interactive sessions.
Number of workers
The number of workers of a defined worker_type that are allocated. Customize number of workers to avoid excessive use of compute resources.
Worker type
The type of a unit capable of performing operational processes dictated by its fleet management system. Select the relevant worker types for your workloads.
Idle timeout
The number of minutes of inactivity after which an interactive session will timeout after a cell has been executed. Define idle-timeout for sessions to terminate after the work completed.
Default (minutes)
2880
Between minimum and maximum
Minimum (minutes)
1
Minimum allowed value: 1
Maximum (minutes)
4000
Timeout
The maximum time in minutes that an interactive session run can consume resources before it is terminated. Setup timeout values to avoid long running sessions.
Tags – optional
Tags are user-defined key-value pairs that provide metadata to organize and classify your AWS resources.
Cancel
Save edits
CloudShell
Feedback
Language
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

사용 프로필을 편집하려면(AWS CLI)

- 다음 명령을 입력합니다. 위의 create 명령에 나와 있는 것과 동일한 `--configuration` 파일 구문이 사용됩니다.

```
aws glue update-usage-profile --name profile-name --configuration file://
config.json
```

여기서 config.json은 대화형 세션(`SessionConfiguration`) 및 작업(`JobConfiguration`)에 대한 파라미터 값을 정의합니다.

사용 프로필 할당

사용 프로필 페이지의 사용률 상태 열에는 사용 프로필이 사용자에게 할당되었는지 여부가 표시됩니다. 상태를 마우스로 가리키면 할당된 IAM 엔터티가 표시됩니다.

관리자는 AWS Glue 리소스를 생성하는 사용자/역할에 AWS Glue 사용 프로필을 할당할 수 있습니다. 프로필 할당은 다음 두 가지 작업의 조합입니다.

- `glue:UsageProfile` 키로 IAM 사용자/역할 태그를 업데이트
- 사용자/역할의 IAM 정책 업데이트

AWS Glue Studio를 사용하여 작업/대화형 세션을 생성하는 사용자의 경우, 관리자가 다음 역할을 태깅합니다.

- 작업 제한의 경우 관리자는 로그인한 콘솔 역할을 태깅합니다.
- 대화형 세션에 대한 제한 사항의 경우 관리자는 사용자가 노트북을 만들 때 제공하는 역할을 태깅합니다.

다음은 관리자가 AWS Glue 리소스를 생성하는 IAM 사용자/역할에 대해 업데이트해야 하는 예시 정책입니다.

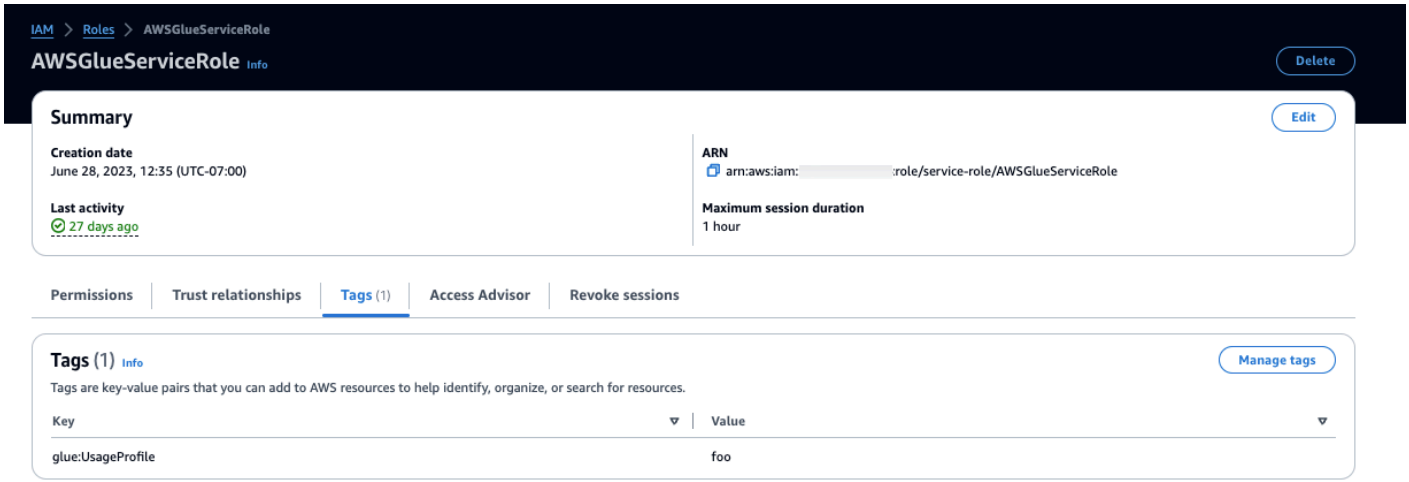
```
{
    "Effect": "Allow",
    "Action": [
        "glue:GetUsageProfile"
    ],
    "Resource": [
```

```
        "arn:aws:glue:us-east-1:123456789012:usageProfile/foo"
    ]
}
```

AWS Glue는 AWS Glue사용 프로필에 지정된 값을 기반으로 작업, 작업 실행 및 세션 요청을 검증하고 요청이 허용되지 않는 경우 예외를 발생시킵니다. 동기 API의 경우 사용자에게 오류가 발생합니다. 비동기 경로의 경우 입력 파라미터가 사용자/역할에 할당된 프로필에 허용되는 범위를 벗어났다는 오류 메시지와 함께 실패한 작업 실행이 생성됩니다.

사용자/역할에 사용 프로필을 할당하려면:

1. Identity and Access Management(IAM) 콘솔을 엽니다.
2. 탐색 창에서 사용자 또는 역할을 선택합니다.
3. 사용자 또는 역할을 선택합니다.
4. 태그 탭을 선택합니다.
5. 새 태그 추가를 선택합니다.
6. 키가 `glue:UsageProfile`이고 값이 사용자 프로필 이름인 태그를 추가합니다.
7. 변경 사항 저장(Save changes)을 선택합니다



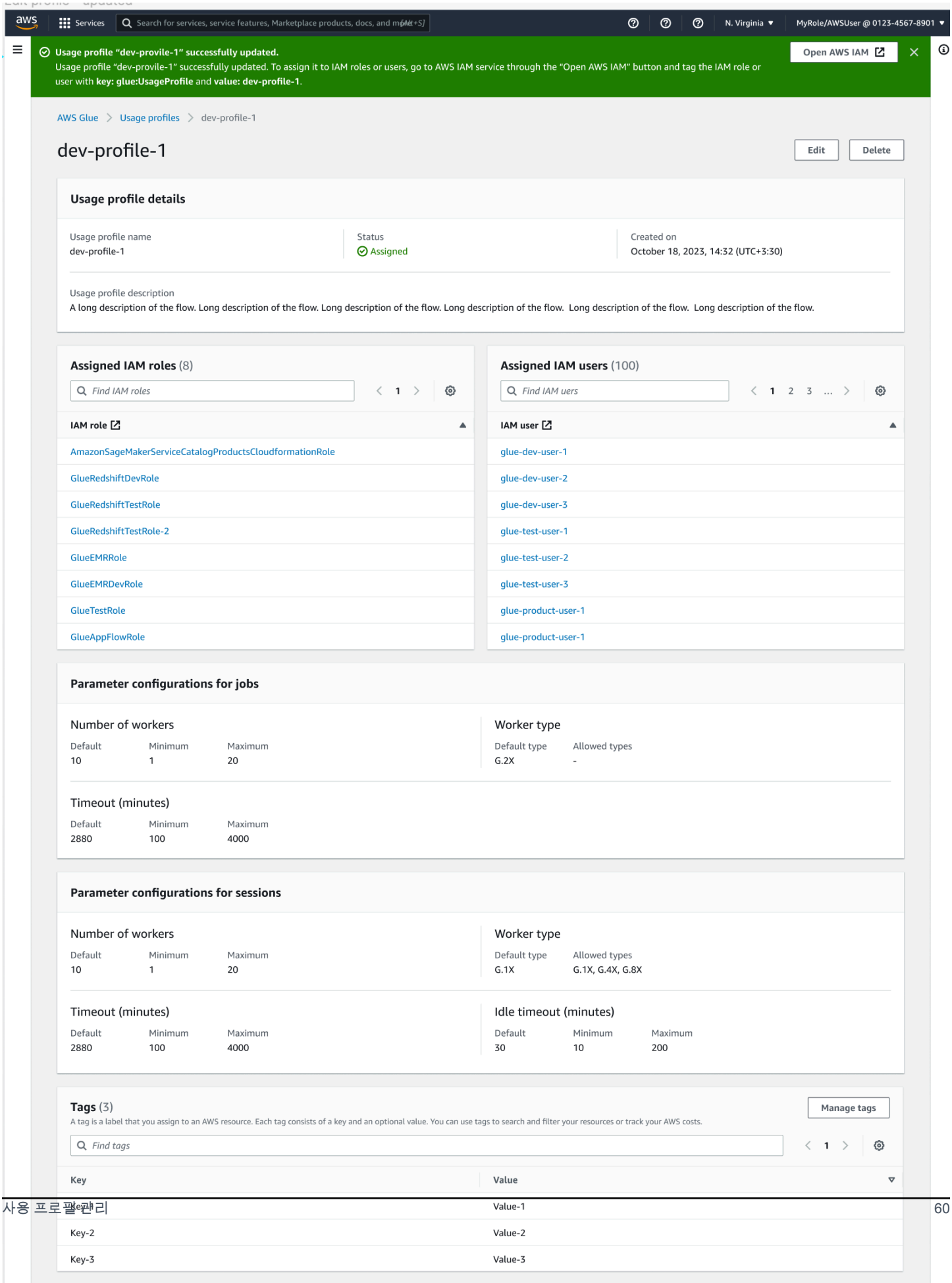
할당된 사용 프로필 보기

사용자는 할당된 사용 프로필을 보고 API 호출을 통해 AWS Glue 작업 및 세션 리소스를 생성하거나 작업을 시작할 때 사용할 수 있습니다.

프로필 권한은 IAM 정책에서 제공됩니다. 호출자 정책에 `glue:UsageProfile` 권한이 있으면 사용자는 프로필을 볼 수 있습니다. 그렇지 않으면 액세스 거부됨 오류가 발생합니다.

할당된 사용 프로필을 보려면:

1. 왼쪽 탐색 창에서 비용 관리를 선택합니다.
2. 볼 권한이 있는 사용자 프로필을 선택합니다.


Services
Search for services, service features, Marketplace products, docs, and more [Alt+S]
N. Virginia
MyRole/AWSUser @ 0123-4567-8901
Usage profile "dev-provile-1" successfully updated.
Usage profile "dev-provile-1" successfully updated. To assign it to IAM roles or users, go to AWS IAM service through the "Open AWS IAM" button and tag the IAM role or user with key: glue:UsageProfile and value: dev-profile-1.
Open AWS IAM
AWS Glue > Usage profiles > dev-profile-1
dev-profile-1
Edit
Delete
Usage profile details
Usage profile name
dev-profile-1
Status
Assigned
Created on
October 18, 2023, 14:32 (UTC+3:30)
Usage profile description
A long description of the flow. Long description of the flow. Long description of the flow. Long description of the flow. Long description of the flow. Long description of the flow.
Assigned IAM roles (8)
Find IAM roles
IAM role
AmazonSageMakerServiceCatalogProductsCloudformationRole
GlueRedshiftDevRole
GlueRedshiftTestRole
GlueRedshiftTestRole-2
GlueEMRRole
GlueEMRDevRole
GlueTestRole
GlueAppFlowRole
Assigned IAM users (100)
Find IAM uers
IAM user
glue-dev-user-1
glue-dev-user-2
glue-dev-user-3
glue-test-user-1
glue-test-user-2
glue-test-user-3
glue-product-user-1
glue-product-user-1
Parameter configurations for jobs
Number of workers
Default 10
Minimum 1
Maximum 20
Worker type
Default type G.2X
Allowed types -
Timeout (minutes)
Default 2880
Minimum 100
Maximum 4000
Parameter configurations for sessions
Number of workers
Default 10
Minimum 1
Maximum 20
Worker type
Default type G.1X
Allowed types G.1X, G.4X, G.8X
Timeout (minutes)
Default 2880
Minimum 100
Maximum 4000
Idle timeout (minutes)
Default 30
Minimum 10
Maximum 200
Tags (3)
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
Manage tags
Find tags
Key
Value
Key-1
Value-1
Key-2
Value-2
Key-3
Value-3

사용 프로필 및 작업

사용 프로필 관련 작업 작성

작업을 작성하는 동안에는 사용 프로필에 설정된 제한과 기본값이 적용됩니다. 저장 시 작업에 프로필이 할당됩니다.

사용 프로필 관련 작업 실행

작업 실행을 시작하면 AWS Glue가 호출자 프로필에 설정된 제한을 적용합니다. 직접 호출자가 없는 경우 Glue는 작성자가 작업에 할당한 프로필의 한도를 적용합니다.

> **Note**
>
> AWS Glue 워크플로 또는 AWS Glue 트리거에 의해 일정에 따라 작업이 실행되면 작성자가 작업에 할당된 프로필이 적용됩니다.
> 외부 서비스(Step Functions, MWAA) 또는 `StartJobRun` API에서 작업을 실행하는 경우 호출자의 프로필 한도가 적용됩니다.

AWS Glue 워크플로 또는 AWS Glue 트리거의 경우: 기존 작업을 업데이트하여 새 프로필 이름을 저장해야 합니다. 그래야 예약된 실행에 대해 런타임에 프로필의 한도(최소, 최대, 허용된 작업자 수)이 적용됩니다.

작업에 할당된 사용 프로필 보기

작업에 할당된 프로필(런타임 시 예약된 AWS Glue 워크플로 또는 AWS Glue 트리거와 함께 사용됨)을 보려면 작업 세부 정보 탭을 확인하면 됩니다. 작업 실행 세부 정보 탭에서 이전 실행에 사용된 프로필을 볼 수도 있습니다.

작업에 연결된 사용 프로필 업데이트 또는 삭제

작업에 할당된 프로필은 업데이트 시에 변경됩니다. 작성자에게 사용 프로필이 할당되지 않은 경우 이전에 작업에 연결된 모든 프로필이 사용자 프로필에서 제거됩니다.

AWS Glue Data Catalog 시작하기

AWS Glue Data Catalog는 영구적 기술 메타데이터 스토어입니다. AWS 클라우드에서 메타데이터를 저장 및 공유하고 주석을 다는 데 사용할 수 있는 관리형 서비스입니다. 자세한 내용은 AWS Glue Data Catalog 단원을 참조하십시오.

AWS Glue 콘솔 및 일부 사용자 인터페이스가 최근에 업데이트되었습니다.

개요

이 자습서를 사용하여 Amazon S3 버킷을 데이터 원본으로 사용하는 첫 번째 AWS Glue 데이터 카탈로그를 생성할 수 있습니다.

이 자습서에서는 AWS Glue 콘솔을 사용하여 다음을 수행합니다.

1. 데이터베이스 생성
2. 테이블 생성
3. Amazon S3 버킷을 데이터 소스로 사용

이 단계를 완료하면 Amazon S3 버킷을 데이터 원본으로 사용하여 AWS Glue 데이터 카탈로그를 채울 수 있습니다.

1단계: 데이터베이스 생성

시작하려면 AWS Management Console에 로그인한 후 AWS Glue 콘솔을 엽니다.

AWS Glue 콘솔을 사용하여 데이터베이스 생성

1. AWS Glue 콘솔에서 왼쪽 메뉴의 Data catalog(데이터 카탈로그)에서 Databases(데이터베이스)를 선택합니다.
2. 데이터베이스 추가(Add database)를 선택합니다.
3. 데이터베이스 생성 페이지에서 데이터베이스의 이름을 입력합니다. 위치 - 선택 사항 섹션에서 데이터 카탈로그의 클라이언트가 사용할 URI 위치를 설정합니다. 이 정보를 몰라도 데이터베이스 생성을 계속할 수 있습니다.
4. (선택 사항). 데이터베이스에 대한 설명을 입력합니다.
5. 데이터베이스 생성를 선택합니다.

축하합니다. AWS Glue 콘솔을 사용하여 첫 번째 데이터베이스를 설정했습니다. 새 데이터베이스가 사용 가능한 데이터베이스 목록에 나타납니다. 데이터베이스(Databases) 대시보드에서 데이터베이스 이름을 선택하여 데이터베이스를 편집할 수 있습니다.

다음 단계

데이터베이스를 생성하는 다른 방법:

방금 AWS Glue 콘솔을 사용하여 데이터베이스를 생성했지만 데이터베이스를 생성하는 다른 방법이 있습니다.

- 크롤러를 사용하여 자동으로 데이터베이스와 테이블을 생성할 수 있습니다. 크롤러를 사용하여 데이터베이스를 설정하려면 AWS Glue 콘솔에서 크롤러 작업을 참조하세요.
- AWS CloudFormation 템플릿을 사용할 수 있습니다. AWS Glue Data Catalog 템플릿을 사용하여 AWS Glue 리소스 생성을 참조하세요.
- AWS Glue 데이터베이스 API 작업을 사용하여 데이터베이스를 생성할 수도 있습니다.

  `create` 작업을 사용하여 데이터베이스를 생성하려면 `DatabaseInput`(필수) 파라미터를 포함하여 요청을 구조화합니다.

  예:

  다음은 CLI, Boto3 또는 DDL을 사용하여 자습서에서 사용한 S3 버킷의 동일한 flight_data.csv 파일을 기반으로 테이블을 정의하는 방법의 예입니다.

  CLI

  ```
  aws glue create-database --database-input "{\"Name\":\"clidb\"}"
  ```

  Boto3

  ```
  glueClient = boto3.client('glue')

  response = glueClient.create_database(
      DatabaseInput={
          'Name': 'boto3db'
      }
  )
  ```

데이터베이스 API 데이터 유형, 구조 및 작업에 대한 자세한 내용은 데이터베이스 API를 참조하세요.

다음 단계

다음 섹션에서는 테이블을 생성하고 데이터베이스에 추가합니다.

데이터 카탈로그에 대한 설정과 권한을 탐색할 수도 있습니다. AWS Glue 콘솔에서 데이터 카탈로그 설정 관련 작업을 참조하세요.

단계 2. 테이블 생성

이 단계에서는 AWS Glue 콘솔을 사용하여 테이블을 생성합니다.

1. AWS Glue 콘솔에서 왼쪽 메뉴에서 테이블(Tables)을 선택합니다.
2. 테이블 추가를 선택합니다.
3. Table details(테이블 세부 정보)에 테이블 이름을 입력하여 테이블 속성을 설정합니다.
4. Databases(데이터베이스) 섹션의 드롭다운 메뉴에서 1단계에서 생성한 데이터베이스를 선택합니다.
5. Add a data store(데이터 스토어 추가) 섹션에서 S3는 소스 유형으로 기본 선택됩니다.
6. Data is located in(데이터 위치)에서 Specified path in another account(다른 계정의 지정된 경로)를 선택합니다.
7. Include path(포함 경로) 입력 필드의 경로를 복사하여 붙여 넣습니다.

   ```
   s3://crawler-public-us-west-2/flight/2016/csv/
   ```

8. Data format(데이터 형식) 섹션의 Classification(분류)에서 CSV를 선택하고 Delimiter(구분 기호)에서 comma (,)(쉼표(,))를 선택합니다. Next(다음)를 선택합니다.
9. 스키마를 정의하라는 메시지가 나타납니다. 스키마는 데이터 레코드의 구조와 포맷을 정의합니다. 열 추가(Add column)를 선택합니다. 자세한 내용은 스키마 레지스트리를 참조하세요.
10. 열 속성을 지정합니다.
    a. 열 이름을 입력합니다.
    b. 열 유형(Column type)에 기본적으로 '문자열(string)'이 이미 선택되어 있습니다.
    c. 열 번호(Column number)에 기본적으로 '1'이 이미 선택되어 있습니다.
    d. 추가를 선택합니다.
11. 파티션 인덱스를 추가하라는 메시지가 나타납니다. 이는 선택 사항입니다. 이 단계를 건너뛰려면 다음(Next)을 선택합니다.
12. 테이블 속성의 요약이 표시됩니다. 모든 것이 예상대로 표시되면 생성을 선택합니다. 그렇지 않으면 뒤로(Back)를 선택하고 필요에 따라 편집합니다.

축하합니다. 성공적으로 테이블을 수동으로 생성하고 데이터베이스에 연결했습니다. 새로 생성된 테이블은 테이블(Tables) 대시보드에 나타납니다. 대시보드에서 모든 테이블을 수정하고 관리할 수 있습니다.

자세한 내용은 AWS Glue 콘솔에서 테이블 관련 작업을 참조하세요.

다음 단계

다음 단계

이제 데이터 카탈로그가 채워졌으므로 AWS Glue에서 작업 작성을 시작할 수 있습니다. AWS Glue Studio를 사용하여 시각적 ETL 작업 구축을 참조하세요.

콘솔을 사용하는 것 외에도 다음과 같은 다른 방법으로 데이터 카탈로그에서 테이블을 정의할 수 있습니다.

- 크롤러 생성 및 실행
- AWS Glue의 크롤러로 분류자 추가
- AWS Glue 테이블 API 사용
- AWS Glue Data Catalog 템플릿 사용
- Apache Hive 메타스토어 마이그레이션
- AWS CLI, Boto3 또는 데이터 정의 언어(DDL) 사용

  다음은 CLI, Boto3 또는 DDL을 사용하여 자습서에서 사용한 S3 버킷의 동일한 flight_data.csv 파일을 기반으로 테이블을 정의하는 방법의 예입니다.

  AWS CLI 명령을 구조화하는 방법은 관련 설명서를 참조하세요. CLI 예제에는 'aws glue create-table --table-input' 값에 대한 JSON 구문이 포함되어 있습니다.

  CLI

```
{
        "Name": "flights_data_cli",
        "StorageDescriptor": {
            "Columns": [
                {
                    "Name": "year",
                    "Type": "bigint"
                },
                {
                    "Name": "quarter",
```

```
                    "Type": "bigint"
                }
            ],
            "Location": "s3://crawler-public-us-west-2/flight/2016/csv",
            "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
            "OutputFormat":
 "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
            "Compressed": false,
            "NumberOfBuckets": -1,
            "SerdeInfo": {
                "SerializationLibrary":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
                "Parameters": {
                    "field.delim": ",",
                    "serialization.format": ","
                }
            }
        },
        "PartitionKeys": [
            {
                "Name": "mon",
                "Type": "string"
            }
        ],
        "TableType": "EXTERNAL_TABLE",
        "Parameters": {
            "EXTERNAL": "TRUE",
            "classification": "csv",
            "columnsOrdered": "true",
            "compressionType": "none",
            "delimiter": ",",
            "skip.header.line.count": "1",
            "typeOfData": "file"
        }
    }
```

Boto3

```
import boto3

glue_client = boto3.client("glue")
```

```
response = glue_client.create_table(
    DatabaseName='sampledb',
    TableInput={
        'Name': 'flights_data_manual',
    'StorageDescriptor': {
      'Columns': [{
        'Name': 'year',
        'Type': 'bigint'
      }, {
        'Name': 'quarter',
        'Type': 'bigint'
      }],
      'Location': 's3://crawler-public-us-west-2/flight/2016/csv',
      'InputFormat': 'org.apache.hadoop.mapred.TextInputFormat',
      'OutputFormat':
 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat',
      'Compressed': False,
      'NumberOfBuckets': -1,
      'SerdeInfo': {
        'SerializationLibrary':
 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe',
        'Parameters': {
          'field.delim': ',',
          'serialization.format': ','
        }
      },
    },
    'PartitionKeys': [{
      'Name': 'mon',
      'Type': 'string'
    }],
    'TableType': 'EXTERNAL_TABLE',
    'Parameters': {
      'EXTERNAL': 'TRUE',
      'classification': 'csv',
      'columnsOrdered': 'true',
      'compressionType': 'none',
      'delimiter': ',',
      'skip.header.line.count': '1',
      'typeOfData': 'file'
    }
    }
)
```

DDL

```
CREATE EXTERNAL TABLE `sampledb`.`flights_data` (
  `year` bigint,
  `quarter` bigint)
PARTITIONED BY (
  `mon` string)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://crawler-public-us-west-2/flight/2016/csv/'
TBLPROPERTIES (
  'classification'='csv',
  'columnsOrdered'='true',
  'compressionType'='none',
  'delimiter'=',',
  'skip.header.line.count'='1',
  'typeOfData'='file')
```

데이터 스토어에 대한 네트워크 액세스 설정

작업을 추출, 변환 및 로드(ETL)하려면 AWS Glue는 데이터 스토어에 액세스할 수 있어야 합니다. 작업이 Virtual Private Cloud(VPC) 서브넷에서 실행될 필요가 없으면(예를 들어, 데이터를 Amazon S3에서 Amazon S3로 변환하는 것) 추가 구성이 필요 없습니다.

VPC 서브넷에서 작업을 실행해야 하는 경우(예: 프라이빗 서브넷의 JDBC 데이터 스토어에서 데이터 변환) AWS Glue는 작업이 VPC 내의 다른 리소스에 안전하게 연결할 수 있도록 탄력적 네트워크 인터페이스를 설정합니다. 지정한 서브넷의 IP 주소 범위에 속하는 프라이빗 IP 주소가 이 탄력적 네트워크 인터페이스에 할당됩니다. 퍼블릭 IP 주소가 할당되지 않습니다. AWS Glue 연결에 지정된 보안 그룹은 탄력적 네트워크 인터페이스 각각에 적용됩니다. 자세한 내용은 AWS Glue에서 Amazon RDS 데이터 스토어에 대해 JDBC를 연결하도록 Amazon VPC 설정 단원을 참조하십시오.

작업이 액세스한 모든 JDBC 데이터 스토어는 VPC 서브넷에서 사용 가능해야 합니다. VPC 안에서 Amazon S3에 액세스하려면 VPC 엔드포인트가 필요합니다. 작업에서 VPC 리소스와 퍼블릭 인터넷에 모두 액세스해야 할 경우 VPC 내부 네트워크 주소 변환(NAT) 게이트웨이가 VPC에 있어야 합니다.

작업 또는 개발 엔드포인트는 한 번에 하나의 VPC(와 서브넷)에만 액세스할 수 있습니다. 다른 VPC의 데이터 스토어로 액세스하려면 다음과 같은 옵션이 있습니다.

- VPC 피어링을 사용하여 데이터 스토어에 액세스합니다. VPC 피어링에 대한 자세한 내용은 VPC 피어링 기초 단원을 참조하십시오.
- Amazon S3 버킷을 중개 스토리지 위치로 사용합니다. 작업 1 출력값인 Amazon S3를 작업 2의 입력값으로 하여 작업을 두 가지로 분할합니다.

Amazon VPC를 사용하여 Amazon Redshift 데이터 스토어에 연결하는 방법에 대한 자세한 내용은 the section called “Redshift 구성” 섹션을 참조하세요.

Amazon VPC를 사용하여 Amazon RDS 데이터 스토어에 연결하는 방법에 대한 자세한 내용은 the section called “Amazon RDS 데이터 스토어에 연결하도록 Amazon VPC 설정” 섹션을 참조하세요.

Amazon VPC에 필요한 규칙을 설정한 후 데이터 스토어에 연결하는 데 필요한 속성을 사용하여 AWS Glue에서 연결을 생성합니다. 연결에 대한 자세한 정보는 데이터에 연결 섹션을 참조하세요.

> **Note**
>
> AWS Glue DNS 환경을 설정하도록 합니다. 자세한 내용은 VPC에서 DNS 설정 단원을 참조하십시오.

주제

- AWS Glue에서 PyPI에 연결하도록 VPC 설정
- VPC에서 DNS 설정

AWS Glue에서 PyPI에 연결하도록 VPC 설정

Python Package Index(PyPI)는 Python 프로그래밍 언어용 소프트웨어 리포지토리입니다. 이 주제에서는 pip 설치 패키지(세션 생성자가 `--additional-python-modules` 플래그를 사용하여 지정함) 사용을 지원하는 데 필요한 세부 정보를 다룹니다.

커넥터와 함께 AWS Glue 대화형 세션을 사용하면 커넥터에 지정된 서브넷을 통해 VPC 네트워크를 사용하게 됩니다. 따라서 특별 구성을 설정하지 않으면 AWS 서비스 및 기타 네트워크 대상을 사용할 수 없습니다.

이 문제의 해결 방법은 다음과 같습니다.

- 세션에서 연결할 수 있는 인터넷 게이트웨이 사용.
- 패키지 세트의 종속성에서 전이 폐쇄 특성을 포함하는 PyPI/simple 리포지토리로 S3 버킷의 설정 및 사용.
- PyPI를 미러링하고 VPC에 연결하는 CodeArtifact 리포지토리 사용.

인터넷 게이트웨이 설정

기술적 측면은 NAT 게이트웨이 사용 사례에 자세히 설명되어 있지만 `--additional-python-modules` 사용에 관한 다음 요구 사항도 참고하세요. 특히, VPC의 구성에 따라 결정되는 pypi.org에 대한 액세스 권한이 `--additional-python-modules`에 필요합니다. 다음과 같은 요구 사항을 확인합니다.

1. 사용자 세션의 경우 pip install을 통해 추가 python 모듈을 설치해야 합니다. 세션에서 커넥터를 사용하는 경우 구성이 영향을 받을 수 있습니다.
2. `--additional-python-modules`에서 커넥터를 사용할 때 세션이 시작되면 커넥터의 `PhysicalConnectionRequirements`에 연결된 서브넷이 pypi.org에 도달하기 위한 네트워크 경로를 제공해야 합니다.
3. 구성이 올바른지 확인해야 합니다.

대상 PyPI/simple 리포지토리를 호스팅하도록 Amazon S3 버킷 설정

이 예제는 Amazon S3에서 패키지 세트 및 해당 종속성에 대한 PyPI 미러를 설정합니다.

패키지 세트에 대한 PyPI 미러를 설정하려면:

```
# pip download all the dependencies
pip download -d s3pypi --only-binary :all: plotly gglplot
pip download -d s3pypi --platform manylinux_2_17_x86_64 --only-binary :all: psycopg2-
binary
# create and upload the pypi/simple index and wheel files to the s3 bucket
s3pypi -b test-domain-name --put-root-index -v s3pypi/*
```

기존 아티팩트 리포지토리가 이미 있는 경우, 위와 같이 Amazon S3 버킷의 예제 URL 대신 제공할 수 있는 pip 사용을 위한 인덱스 URL이 있을 것입니다.

몇 가지 예제 패키지에서 사용자 지정 indexx-url을 사용하려면:

```
%%configure
{
    "--additional-python-modules": "psycopg2_binary==2.9.5",
    "python-modules-installer-option": "--no-cache-dir --verbose --index-url https://
test-domain-name.s3.amazonaws.com/ --trusted-host test-domain-name.s3.amazonaws.com"
}
```

VPC에 연결된 pypi의 CodeArtifact 미러 설정

미러를 설정하려면:

1. 커넥터에서 사용하는 서브넷과 동일한 리전에서 리포지토리를 생성합니다.

   `Public upstream repositories`를 선택하고 `pypi-store`를 선택합니다.
2. 서브넷에 VPC에서 리포지토리로의 액세스를 제공합니다.
3. `python-modules-installer-option`을 사용하여 올바른 `--index-url`을 지정합니다.

   ```
   %%configure
   {
       "--additional-python-modules": "psycopg2_binary==2.9.5",
       "python-modules-installer-option": "--no-cache-dir --verbose --index-url https://
   test-domain-name.s3.amazonaws.com/ --trusted-host test-domain-name.s3.amazonaws.com"
   }
   ```

자세한 내용은 VPC에서 CodeArtifact 사용을 참조하세요.

VPC에서 DNS 설정

도메인 이름 시스템(DNS)은 인터넷에서 사용되는 이름을 해당 IP 주소로 확인할 때 기준이 됩니다. DNS 호스트 이름은 컴퓨터를 고유적으로 지정하는 이름으로서, 호스트 이름과 도메인 이름으로 구성됩니다. DNS 서버는 DNS 호스트 이름을 해당 IP 주소로 확인합니다.

VPC에 DNS를 설정하려면 DNS 호스트 이름과 DNS 확인이 모두 VPC에서 활성화되었는지 확인합니다. VPC 네트워크 속성 `enableDnsHostnames` 및 `enableDnsSupport`는 `true`로 설정되어 있어야

합니다. 이러한 속성을 보고 수정하려면 https://console.aws.amazon.com/vpc/의 VPC 콘솔로 이동합니다.

자세한 내용은 VPC에서 DNS 사용하기 단원을 참조하세요. 또한 AWS CLI를 사용하고 modify-vpc-attribute 명령을 호출하여 VPC 네트워크 속성을 구성할 수 있습니다.

> **Note**
>
> Route 53를 사용한다면 구성이 DNS 네트워크 속성을 재정의하지 않도록 확인합니다.

AWS Glue에서 암호화 설정

다음 예제 작업 흐름에는 AWS Glue로 암호화를 사용할 때 구성하는 옵션이 강조되어 있습니다. 이 예제는 특정 AWS Key Management Service(AWS KMS) 키 사용을 설명하지만 사용자의 특정 요구에 따라 다른 설정을 선택할 수도 있습니다. 이 작업 흐름에는 AWS Glue 설정 시 암호화에 관한 옵션만 강조 표시되어 있습니다.

1. AWS Glue 콘솔 사용자가 모든 AWS Glue API 작업(예: `"glue:*"`)을 허용하는 권한 정책을 사용하지 않을 경우에는 다음 작업이 허용되는지 확인하십시오.
   - `"glue:GetDataCatalogEncryptionSettings"`
   - `"glue:PutDataCatalogEncryptionSettings"`
   - `"glue:CreateSecurityConfiguration"`
   - `"glue:GetSecurityConfiguration"`
   - `"glue:GetSecurityConfigurations"`
   - `"glue:DeleteSecurityConfiguration"`
2. 암호화된 카탈로그에 액세스하거나 이에 쓰는 모든 클라이언트(즉, 콘솔 사용자, 크롤러, 작업 또는 개발 엔드포인트)에는 다음 권한이 필요합니다.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt",
           "kms:Encrypt"
      ],
```

```
      "Resource": "<key-arns-used-for-data-catalog>"
    }
}
```

3. 암호화된 연결 암호에 액세스하는 사용자 또는 역할은 다음 권한이 필요합니다.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt"
         ],
     "Resource": "<key-arns-used-for-password-encryption>"
         }
}
```

4. 암호화된 데이터를 Amazon S3에 기록하는 추출, 변환, 로드 작업 역할은 다음 권한이 있어야 합니다.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt",
           "kms:Encrypt",
           "kms:GenerateDataKey"
       ],
     "Resource": "<key-arns-used-for-s3>"
    }
}
```

5. 암호화된 Amazon CloudWatch Logs를 기록하는 ETL 작업 또는 크롤러는 키 및 IAM 정책에 다음 권한이 있어야 합니다.

   키 정책(IAM 정책 아님)에서 다음을 수행합니다.

```
{
  "Effect": "Allow",
  "Principal": {
   "Service": "logs.region.amazonaws.com"
  },
```

```
  "Action": [
   "kms:Encrypt*",
   "kms:Decrypt*",
   "kms:ReEncrypt*",
   "kms:GenerateDataKey*",
   "kms:Describe*"
  ],
  "Resource": "<arn of key used for ETL/crawler cloudwatch encryption>"
 }
```

키 정책에 대한 자세한 내용은 AWS Key Management Service Developer Guide의 Using Key Policies in AWS KMS를 참조하세요.

IAM 정책에서 `logs:AssociateKmsKey` 권한을 첨부합니다.

```
{
  "Effect": "Allow",
  "Principal": {
   "Service": "logs.region.amazonaws.com"
  },
  "Action": [
   "logs:AssociateKmsKey"
  ],
  "Resource": "<arn of key used for ETL/crawler cloudwatch encryption>"
 }
```

6. 암호화된 작업 북마크를 사용하는 ETL 작업은 다음 권한이 있어야 합니다.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "<key-arns-used-for-job-bookmark-encryption>"
   }
}
```

7. AWS Glue 콘솔의 탐색 창에서 설정을 선택합니다.

a. [데이터 카탈로그 설정(Data catalog settings)] 페이지에서 [메타데이터 암호화(Metadata encryption)]를 선택하여 Data Catalog를 암호화합니다. 이 옵션은 사용자가 선택한 AWS KMS 키로 Data Catalog의 모든 객체를 암호화합니다.

b. AWS KMS 키의 경우 aws/glue를 선택합니다. 사용자가 생성한 AWS KMS 키를 선택할 수도 있습니다.

> **Important**
>
> AWS Glue에서는 대칭 고객 마스터 키(CMK)만 지원합니다. AWS KMS 키(key) 목록에는 대칭 키만 표시됩니다. 그러나 Choose a AWS KMS key ARN(KMS 키 ARN 선택)을 선택하면 콘솔에서 모든 키 유형의 ARN을 입력할 수 있습니다. 대칭 키에 대한 ARN만 입력해야 합니다.

암호화가 활성화된 경우 Data Catalog에 액세스하는 클라이언트는 AWS KMS 권한이 있어야 합니다.

8. 탐색 창에서 보안 구성을 선택합니다. 보안 구성은 AWS Glue 프로세스를 구성할 때 사용할 수 있는 보안 속성 세트입니다. 그런 다음 보안 구성 추가를 선택합니다. 구성에서 다음 옵션 중 하나를 선택합니다.

a. S3 암호화를 선택합니다. 암호화 모드로 SSE-KMS를 선택합니다. AWS KMS키는 aws/s3을 선택합니다(사용자에게 이 키를 사용할 권한이 있어야 합니다). 이제 작업이 Amazon S3에 기록한 데이터로 AWS 관리형 AWS Glue AWS KMS 키를 사용할 수 있습니다.

b. CloudWatch Logs 암호화를 선택하고 CMK를 선택합니다. 사용자에게 이 키를 사용할 권한이 있어야 합니다. 자세한 내용은 AWS Key Management Service Developer Guide의 Encrypt Log Data in CloudWatch Logs Using AWS KMS를 참조하세요.

> **Important**
>
> AWS Glue에서는 대칭 고객 마스터 키(CMK)만 지원합니다. AWS KMS 키(key) 목록에는 대칭 키만 표시됩니다. 그러나 Choose a AWS KMS key ARN(KMS 키 ARN 선택)을 선택하면 콘솔에서 모든 키 유형의 ARN을 입력할 수 있습니다. 대칭 키에 대한 ARN만 입력해야 합니다.

c. 고급 속성을 선택하고 작업 북마크 암호화를 선택합니다. AWS KMS키는 aws/glue를 선택합니다(사용자에게 이 키를 사용할 권한이 있어야 합니다). 이제 Amazon S3에 기록된 작업 북마크를 AWS Glue AWS KMS 키로 암호화할 수 있습니다.

9. 탐색 창에서 연결을 선택합니다.
   a. 연결 추가를 선택하여 ETL 작업의 대상인 Java Database Connectivity(JDBC) 데이터 스토어에 대한 연결을 생성합니다.
   b. Secure Sockets Layer(SSL) 암호화를 사용하도록 강제하려면 Require SSL connection(SSL 연결 요구)을 선택하고 연결을 테스트합니다.
10. 탐색 창에서, 작업을 선택합니다.
   a. 작업 추가를 선택하여 데이터를 변환하는 작업을 만듭니다.
   b. 작업 정의에서 생성한 보안 구성을 선택합니다.
11. AWS Glue 콘솔에서 온디맨드 작업을 실행합니다. 작업에서 작성한 Amazon S3 데이터, 작업에서 작성한 CloudWatch Logs 및 작업 북마크가 모두 암호화되어 있는지 확인합니다.

AWS Glue의 개발에 대한 네트워킹 설정

AWS Glue로 추출, 변환, 로드(ETL) 스크립트를 실행하기 위해서 개발 엔드포인트를 사용하여 스크립트를 개발하고 테스트할 수 있습니다. 개발 엔드포인트는 AWS Glue 버전 2.0 작업과 함께 사용하도록 지원되지 않습니다. 버전 2.0 이상의 경우 선호하는 개발 방법은 AWS Glue 커널 중 하나에서 Jupyter Notebook을 사용하는 것입니다. 자세한 내용은 the section called “AWS Glue 대화형 세션 시작하기” 단원을 참조하십시오.

개발 엔드포인트에 대한 네트워크 설정

개발 엔드포인트를 설정할 때 Virtual Private Cloud(VPC), 서브넷 및 보안 그룹을 지정합니다.

> **Note**
>
> AWS Glue DNS 환경을 설정하도록 합니다. 자세한 내용은 VPC에서 DNS 설정 단원을 참조하십시오.

AWS Glue를 가능하게 하여 필요한 리소스에 액세스하기 위해서는 행을 서브넷 라우팅 테이블에 추가하여 Amazon S3 접두사 목록과 VPC 엔드포인트를 연결합니다. 접두사 목록 ID는 VPC 트래픽을 허용하여 VPC 엔드포인트를 통해 AWS 서비스로 액세스할 수 있는 아웃바운드 보안 그룹을 생성하는 데 필요합니다. 로컬 시스템에서 이 개발 엔드포인트와 연결된 노트북 서버를 쉽게 연결하려면 행을 라우팅 테이블에 추가하여 인터넷 게이트웨이 ID를 추가합니다. 자세한 내용은 VPC 엔드포인트를 참조하세요. 서브넷 라우팅 테이블을 업데이트하여 다음 테이블과 비슷하게 만듭니다.

대상 주소	대상		
10.0.0.0/16	로컬		
Amazon S3용 pl-id	vpce-id		
0.0.0.0/0	igw-xxxx		

AWS Glue가 구성 요소 간에 통신하려면 보안 그룹이 모든 TCP 포트에 자기 참조 인바운드 규칙을 지정해야 합니다. 자기 참조 규칙을 생성하여 VPC에서 동일한 보안 그룹으로 소스를 제한하고 모든 네트워크로 공개되지 않도록 합니다. VPC의 기본 보안 그룹은 ALL Traffic의 자기 참조 인바운드 액세스 규칙이 있습니다.

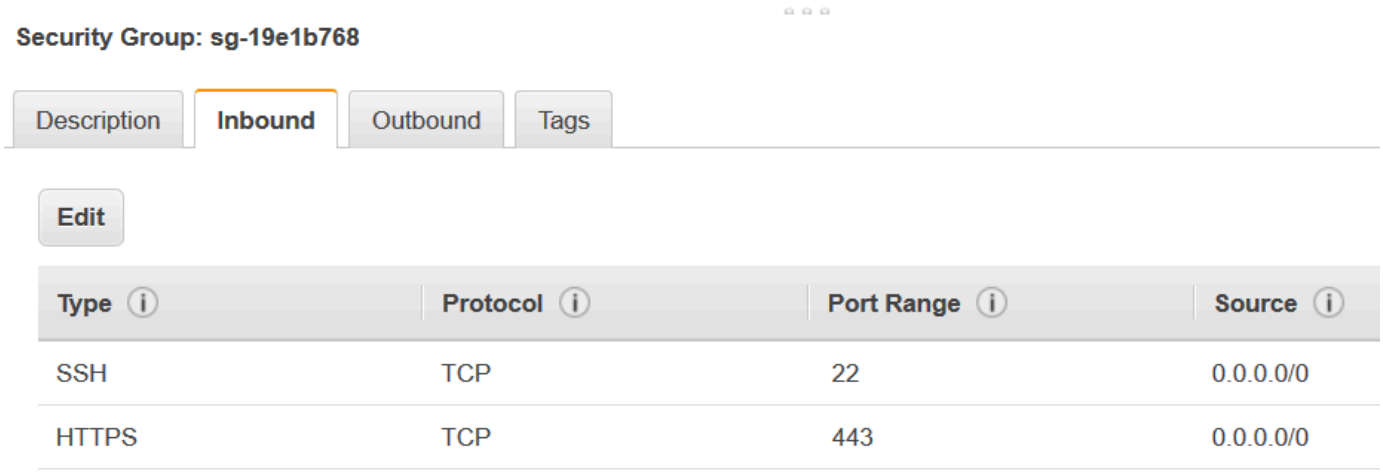
보안 그룹을 설정하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [Security Groups]를 선택합니다.
3. 목록에서 기존 보안 그룹을 선택하거나 [Create Security Group]를 선택하여 개발 엔드포인트와 함께 사용합니다.
4. 보안 그룹 화면에서 [Inbound] 탭으로 이동합니다.
5. 자기 참조 규칙을 추가하여 AWS Glue 구성 요소를 허용하여 통신합니다. 특히, [Type(유형)] `All TCP` 의 규칙을 추가하고 확인합니다. [Protocol(프로토콜)]는 `TCP`이고 [Port Range(포트 범위)]는 모든 포트를 포함하고 포트의 [Source(원본)]은 [Group ID(그룹 ID)]이라는 동일한 보안 그룹입니다.

   인바운드 규칙은 다음과 비슷하게 보입니다.

유형	프로토콜	포트 범위	소스
모든 TCP	TCP	0~65535	*security-group*

   다음은 자기 참조 인바운드 규칙 예제입니다.



6. 아웃바운드 트래픽 규칙도 추가합니다. 아웃바운드 트래픽을 모든 포트로 열거나 자기 [Type(유형)] `All TCP` 의 자기 참조 규칙을 생성합니다. [Protocol(프로토콜)]는 TCP이고 [Port Range(포트 범위)]는 모든 포트를 포함하고 포트의 [Source(원본)]은 [Group ID(그룹 ID)]이라는 동일한 보안 그룹입니다.

   아웃바운드 규칙은 다음 규칙 중 하나와 비슷합니다.

유형	프로토콜	포트 범위	대상
모든 TCP	TCP	0~65535	*security-group*
모든 트래픽	ALL	ALL	0.0.0.0/0

노트북 서버용 Amazon EC2 설정

개발 엔드포인트를 통해 노트북 서버를 생성하여 Jupyter 노트북에서 ETL 스크립트를 테스트합니다. 노트북 통신을 가능하게 하려면 HTTPS(port 443)와 SSH(port 22)의 인바운드 규칙으로 보안 그룹을 지정합니다. 규칙의 소스가 0.0.0.0/0 또는 노트북에 연결하는 시스템의 IP 주소인지 확인합니다.

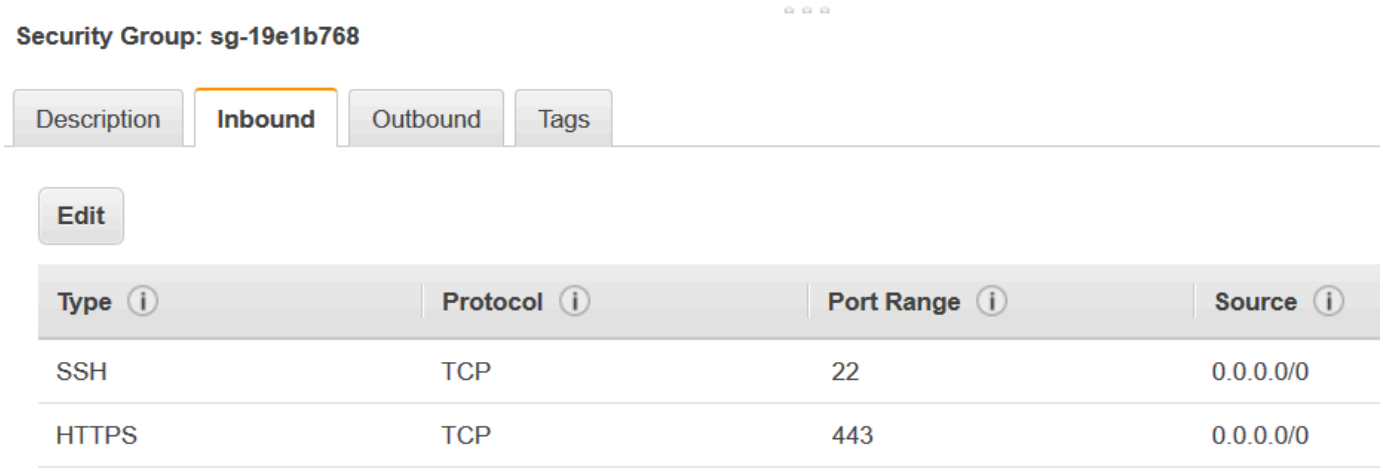
보안 그룹을 설정하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [Security Groups]를 선택합니다.

3. 목록에서 기존 보안 그룹을 선택하거나 [Create Security Group]를 선택하여 노트북 서버와 함께 사용합니다. 개발 엔드포인트와 연결된 보안 그룹은 노트북 서버를 생성하는 데도 사용할 수 있습니다.
4. 보안 그룹 화면에서 [Inbound] 탭으로 이동합니다.
5. 이와 비슷한 인바운드 규칙을 추가합니다.

유형	프로토콜	포트 범위	소스
SSH	TCP	22	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

다음은 보안 그룹에 대한 인바운드 규칙 예제를 소개합니다.



AWS Glue에서 데이터 검색 및 카탈로그 작성

AWS Glue Data Catalog는 조직의 데이터 세트에 대한 메타데이터를 저장하는 중앙 집중식 리포지토리입니다. 즉, 데이터 소스의 위치, 스키마 및 런타임 메트릭에 대한 인덱스로 동작합니다. 메타데이터는 메타데이터 테이블에 저장되며, 여기서 각 테이블은 단일 데이터 스토어를 나타냅니다.

데이터 소스를 자동으로 스캔하고 메타데이터를 추출하는 크롤러를 사용하여 데이터 카탈로그를 채울 수 있습니다. 크롤러는 AWS 내부(AWS 기반) 및 외부 데이터 소스에 연결할 수 있습니다.

지원되는 데이터 소스에 대한 자세한 내용은 크롤링에 지원되는 데이터 소스 섹션을 참조하세요.

특정 요구 사항에 따라 테이블 구조, 스키마 및 파티셔닝 구조를 정의하여 데이터 카탈로그에서 수동으로 테이블을 생성할 수도 있습니다.

수동으로 메타데이터 테이블을 생성하는 것에 대한 자세한 내용은 수동으로 메타데이터 정의 섹션을 참조하세요.

데이터 카탈로그에서 이 정보를 사용하여 ETL 작업을 생성하고 모니터링할 수 있습니다. 데이터 카탈로그는 다른 AWS 분석 서비스와 통합되어 데이터 소스에 대한 통합 보기를 제공하므로 데이터를 더 쉽게 관리하고 분석할 수 있습니다.

- Amazon Athena - SQL을 사용하여 Amazon S3 데이터에 대한 테이블 메타데이터를 데이터 카탈로그에 저장하고 쿼리합니다.
- AWS Lake Formation - 세분화된 데이터 액세스 정책을 중앙에서 정의하고 관리하며 데이터 액세스를 감사합니다.
- Amazon EMR - 빅 데이터 처리를 위해 데이터 카탈로그에 정의된 데이터 소스에 액세스합니다.
- Amazon SageMaker AI - 기계 학습 모델을 빠르고 확실하게 구축하고, 학습시키고, 배포합니다.

데이터 카탈로그의 주요 기능

다음은 데이터 카탈로그의 주요 기능입니다.

메타데이터 리포지토리

데이터 카탈로그는 중앙 메타데이터 리포지토리로 작동하며, 데이터 소스의 위치, 스키마 및 속성에 대한 정보를 저장합니다. 이 메타데이터는 기존 관계형 데이터베이스 카탈로그와 유사하게 데이터베이스 및 테이블로 구성됩니다.

자동 데이터 검색 기능

AWS Glue 크롤러는 새 데이터 소스나 업데이트된 데이터 소스를 자동으로 검색하고 카탈로그를 작성하여 수동 메타데이터 관리에 따른 오버헤드를 줄이고 데이터 카탈로그가 최신 상태로 유지되도록 보장합니다. 데이터 카탈로그는 데이터 소스의 카탈로그를 작성함으로써 사용자와 애플리케이션이 조직 내에서 사용 가능한 데이터 자산을 더 쉽게 발견하고 이해할 수 있도록 도와주며 데이터 재사용과 협업을 촉진시킵니다.

데이터 카탈로그는 Amazon S3, Amazon RDS, Amazon Redshift, Apache Hive 등을 비롯한 매우 다양한 데이터 소스를 지원합니다. 또한 AWS Glue 크롤러를 사용하여 이러한 소스에서 메타데이터를 자동으로 추론하고 저장할 수 있습니다.

자세한 내용은 크롤러를 사용하여 데이터 카탈로그 채우기 단원을 참조하십시오.

스키마 관리

데이터 카탈로그는 스키마 추론, 진화 및 버전 관리를 포함하여 데이터 소스의 스키마를 자동으로 캡처하고 관리합니다. 데이터 카탈로그에서 AWS Glue ETL 작업을 사용하여 스키마 및 파티션을 업데이트할 수 있습니다.

테이블 최적화

AWS 분석 서비스(예: Amazon Athena 및 Amazon EMR)와 AWS Glue ETL 작업에서 읽기 성능을 향상시키기 위해 데이터 카탈로그는 데이터 카탈로그의 Iceberg 테이블에 대해 관리형 압축(작은 Amazon S3 객체를 큰 객체로 압축하는 프로세스)을 제공합니다. AWS Glue 콘솔, AWS Lake Formation 콘솔, AWS CLI 또는 AWS API를 사용하여 데이터 카탈로그에 있는 개별 Iceberg 테이블에 대한 압축을 활성화하거나 비활성화할 수 있습니다.

자세한 내용은 Iceberg 테이블 최적화 섹션을 참조하세요.

열 통계값

추가 데이터 파이프라인을 설정하지 않고도 Parquet, ORC, JSON, ION, CSV 및 XML과 같은 데이터 형식의 데이터 카탈로그 테이블에 대한 열 수준 통계를 계산할 수 있습니다. 열 통계는 열 내 값에 대한 통찰력을 얻어 데이터 프로필을 이해하는 데 도움이 됩니다. 데이터 카탈로그는 최소값, 최대값, 총 null 값, 총 고유 값, 값의 평균 길이, 실제 값의 총 발생 횟수 등과 같은 열 값에 대한 통계 생성을 지원합니다.

자세한 내용은 열 통계를 사용한 쿼리 성능 최적화 섹션을 참조하세요.

데이터 계보

데이터 카탈로그는 데이터에 대해 수행된 변환 및 작업의 기록을 유지 관리하며, 데이터 계보 정보를 제공합니다. 이 계보 정보는 감사, 규정 준수 및 데이터 출처 이해에 유용합니다.

다른 AWS 서비스와의 통합

데이터 카탈로그는 AWS Lake Formation, Amazon Athena, Amazon Redshift Spectrum, Amazon EMR 등과 같은 다른 AWS 서비스와 원활하게 통합됩니다. 이 통합을 통해 일관된 단일 메타데이터 계층을 사용하여 다양한 데이터 스토어 간에서 데이터를 쿼리하고 분석할 수 있습니다.

보안 및 액세스 제어

AWS Glue는 AWS Lake Formation과 통합되어 Data Catalog 리소스에 대한 세분화된 액세스 제어를 지원하므로 조직의 정책 및 요구 사항에 따라 데이터 자산에 대한 사용 권한을 관리하고 액세스를 보호할 수 있습니다. AWS Glue는 AWS Key Management Service(AWS KMS)와 통합되어 데이터 카탈로그에 저장된 메타데이터를 암호화합니다.

주제

- AWS Glue 데이터 카탈로그 채우기
- 트랜잭션 테이블 채우기 및 관리
- 데이터 카탈로그 관리
- 데이터 카탈로그 액세스
- AWS Glue 데이터 카탈로그 모범 사례
- Amazon CloudWatch에서 Data Catalog 사용량 지표 모니터링
- AWS Glue 스키마 레지스트리

AWS Glue 데이터 카탈로그 채우기

다음 방법을 사용하여 AWS Glue Data Catalog를 채울 수 있습니다.

- AWS Glue 크롤러 - AWS Glue 크롤러를 통해 데이터베이스, 데이터 레이크 및 스트리밍 데이터와 같은 데이터 소스를 자동으로 검색하고 카탈로그화할 수 있습니다. 크롤러는 매우 다양한 데이터 소스의 메타데이터를 자동으로 검색하고 추론할 수 있으므로 데이터 카탈로그를 채울 때 가장 일반적이고 권장되는 방법입니다.
- 메타데이터 수동 추가 - AWS Glue 콘솔, Lake Formation 콘솔, AWS CLI 또는 AWS Glue API를 사용하여 데이터베이스, 테이블 및 연결 세부 정보를 수동으로 정의하고 데이터 카탈로그에 추가할 수 있습니다. 수동 입력은 크롤링할 수 없는 데이터 소스의 카탈로그를 작성하려는 경우에 유용합니다.
- 다른 AWS 서비스와의 통합 - AWS Lake Formation 및 Amazon Athena와 같은 서비스의 메타데이터로 데이터 카탈로그를 채울 수 있습니다. 이러한 서비스는 데이터 카탈로그에서 데이터 소스를 검색하고 등록할 수 있습니다.

- 기존 메타데이터 리포지토리에서 채우기 - Apache Hive Metastore와 같은 기존 메타데이터 저장소가 있는 경우 AWS Glue를 사용하여 해당 메타데이터를 데이터 카탈로그로 가져올 수 있습니다. 자세한 내용은 GitHub의 Hive 메타스토어와 AWS Glue Data Catalog 간의 마이그레이션을 참조하세요.

주제

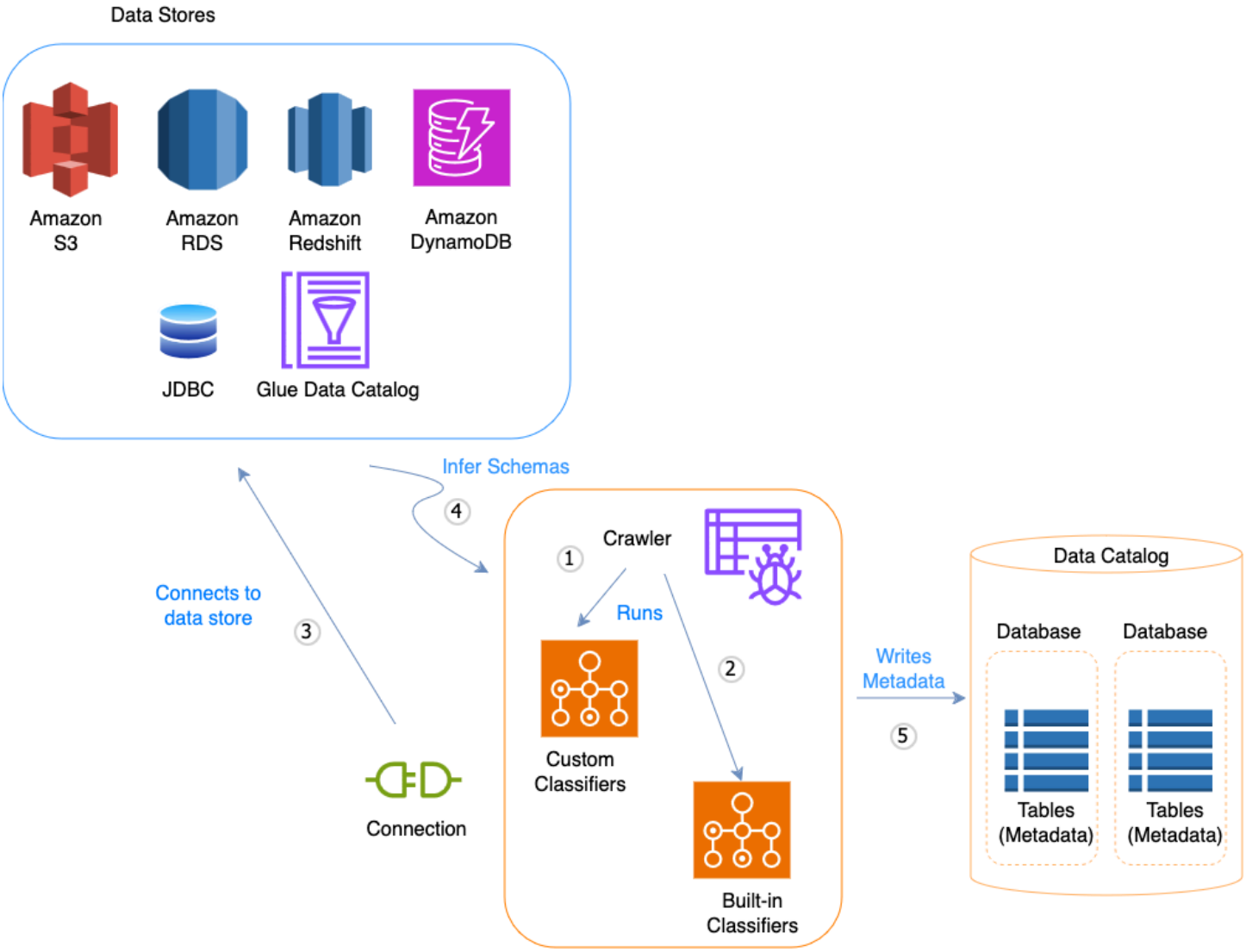
- 크롤러를 사용하여 데이터 카탈로그 채우기
- 수동으로 메타데이터 정의
- 다른 AWS 서비스와 통합
- 데이터 카탈로그 설정

크롤러를 사용하여 데이터 카탈로그 채우기

AWS Glue 크롤러를 사용하면 데이터베이스 및 테이블을 사용하여 AWS Glue Data Catalog를 채울 수 있습니다. 대부분의 AWS Glue 사용자가 사용하는 기본적인 방법입니다. 크롤러는 단일 실행으로 여러 데이터 스토어를 크롤할 수 있습니다. 완료 시 크롤러는 데이터 카탈로그에서 하나 이상의 테이블을 생성하거나 업데이트합니다. AWS Glue에서 정의한 추출, 변환, 로드 작업은 이러한 데이터 카탈로그 테이블을 원본 및 대상으로 사용합니다. ETL 작업은 원본 및 대상 데이터 카탈로그 테이블에 지정된 데이터 스토어에서 읽기와 쓰기를 수행합니다.

워크플로

다음 워크플로 다이어그램은 AWS Glue 크롤러가 데이터 스토어와 다른 요소와 상호 작용하여 Data Catalog를 채우는 방법을 보여줍니다.



다음은 크롤러가 AWS Glue Data Catalog을 채우는 방법에 대한 일반적인 워크플로입니다.

1. 크롤러는 선택한 사용자 지정 분류자를 실행하여 데이터의 형식 및 스키마를 추론합니다. 사용자 지정 분류자에 대한 코드를 제공하면 지정한 순서대로 실행됩니다.

   첫 번째 사용자 분류자가 데이터 구조를 성공적으로 인식하는 과정은 테이블의 스키마를 생성하는 데 사용됩니다. 하위 목록에 있는 사용자 분류자는 건너뜁니다.

2. 어떠한 사용자 지정 분류자도 데이터 스키마와 일치하지 않는다면 기본 설정 분류자는 데이터 스키마를 인식할 시도를 합니다. 기본 설정 분류자의 예는 JSON을 인식하는 분류자입니다.
3. 크롤러를 데이터 스토어로 연결합니다. 어떤 데이터 스토어는 크롤러 액세스 연결 속성을 요구합니다.
4. 추론된 스키마는 데이터 때문에 생성됩니다.

5. 크롤러는 메타데이터를 Data Catalog로 작성합니다. 테이블 정의는 데이터 스토어의 데이터에 대한 메타데이터를 포함합니다. 테이블은 Data Catalog에서 테이블 컨테이너인 데이터베이스에 작성됩니다. 테이블 속성은 테이블 스키마를 추론한 분류자에 의해 생성된 라벨인 분류자를 포함합니다.

주제

- 크롤러 작동 방식
- 크롤러는 파티션 생성 시기를 어떻게 결정하나요?
- 크롤링에 지원되는 데이터 소스
- 크롤러 사전 조건
- 분류자 정의 및 관리
- 크롤러 구성
- 크롤러 일정 관리
- 크롤러 결과 및 세부 정보 보기
- 크롤러 동작 사용자 지정
- 튜토리얼: AWS Glue 크롤러 추가

크롤러 작동 방식

크롤러가 실행되면 데이터 스토어에서 정보를 얻기 위한 다음 작업을 실행합니다.

- 데이터를 분류하여 원시 데이터의 포맷, 스키마 및 관련 속성 결정 - 분류 결과는 사용자 정의 분류자를 생성하여 구성할 수 있습니다.
- 데이터를 테이블 혹은 파티션으로 분류합니다. – 데이터는 크롤러 발견을 기반으로 분류합니다.
- 메타데이터를 데이터 카탈로그에 작성합니다 – 크롤러가 어떻게 테이블과 파티션을 추가하고 업데이트, 삭제하는지 구성합니다.

크롤러를 정의할 때 스키마를 추론할 수 있도록 데이터 포맷을 평가하는 하나 이상의 분류자를 선택합니다. 크롤러가 실행되면 목록의 첫 번째 분류자가 성공적으로 데이터 스토어를 인식하고 테이블의 스키마를 생성합니다. 기본 제공 분류자를 사용하거나 사용자가 직접 정의할 수 있습니다. 크롤러를 정의하기 전에 별도의 작업에서 사용자 지정 분류자를 정의합니다. AWS Glue는 기본 제공 분류자를 제공하여 일반 파일에서 JSON, CSV 및 Apache Avro를 포함하는 포맷으로 스키마를 추론합니다. AWS Glue의 기본 제공 분류자의 현재 목록은 기본 제공 분류자 섹션을 참조하세요.

크롤러가 생성하는 메타데이터는 크롤러를 정의할 때 데이터베이스에 포함됩니다. 크롤러가 데이터베이스를 지정하지 않으면 테이블은 기본 데이터베이스로 배치합니다. 또한, 각 테이블은 처음으로 데이터 스토어를 성공적으로 인식하는 분류자로 채워진 분류 열이 있습니다.

크롤된 파일이 압축되면 크롤러는 반드시 다운로드하고 실행해야 합니다. 크롤러가 실행되면 크롤러는 파일 정보를 얻어 파일 포맷 및 압축 유형을 결정하고 파일 속성을 데이터 카탈로그에 작성합니다. Apache Parquet과 같은 일부 파일 형식은 파일이 작성한 대로 파일 일부를 압축할 수 있습니다. 이런 파일의 압축 데이터는 파일의 내부 구성 요소이고, AWS Glue는 데이터 카탈로그에 테이블을 쓸 때 `compressionType` 속성을 채우지 않습니다. 반대로 전체 파일이 gzip처럼 압축 알고리즘을 통해 압축된 후 `compressionType` 속성은 테이블이 데이터 카탈로그에 작성될 때 채워집니다.

크롤러는 생성하는 테이블 이름을 만듭니다. AWS Glue Data Catalog에 저장된 테이블 이름은 다음 규칙을 따릅니다.

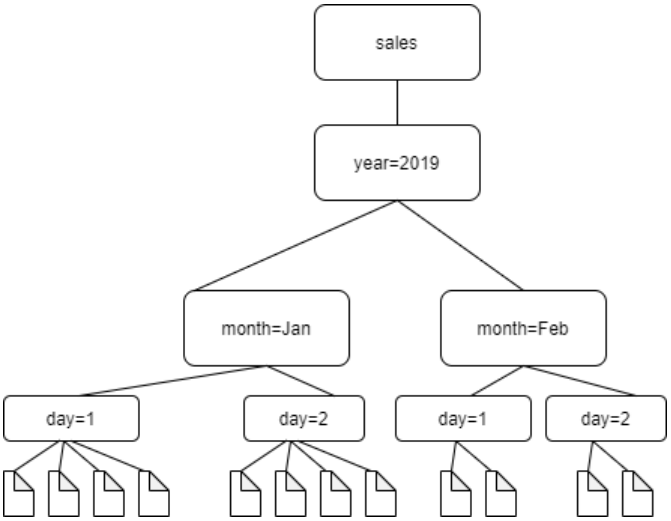
- 영숫자와 밑줄(_)만 허용됩니다.
- 사용자 지정 접두사는 64자보다 길 수 없습니다.
- 이름 최대 길이는 128자보다 길 수 없습니다. 크롤러는 이름이 제한 범위 내에 있도록 이름을 줄입니다.
- 테이블 이름이 복제된다면 크롤러는 이름에 해시 문자열 접미사를 추가합니다.

크롤러가 일정에 따라 한 번을 초과하여 실행된다면 데이터 스토어에서 새로운 또는 변화된 파일 혹은 테이블을 찾습니다. 크롤러 출력값은 과거 실행에서 찾은 새로운 테이블과 파티션을 포함합니다.

크롤러는 파티션 생성 시기를 어떻게 결정하나요?

AWS Glue 크롤러가 Amazon S3 데이터 stpre를 스캔하고 버킷에서 여러 폴더를 발견하면 폴더 구조의 테이블 루트 및 테이블의 파티션에 해당하는 폴더를 결정합니다. Amazon S3 접두사 또는 폴더 이름을 기반의 테이블 이름. 크롤할 폴더 수준을 가리키는 [추가 경로]를 제공합니다. 폴더 수준의 다수 스키마가 유사한 경우, 크롤러는 개별 테이블 대신 테이블 파티션을 생성합니다. 크롤러가 개별 테이블에 영향을 주는 방법은 크롤러를 정의할 때 각 테이블의 루트 폴더를 데이터 스토어로 추가하는 것입니다.

예를 들어 다음과 같은 Amazon S3 폴더 구조를 고려합니다.



4개의 최하위 수준 폴더에 대한 경로는 다음과 같습니다.

```
S3://sales/year=2019/month=Jan/day=1
S3://sales/year=2019/month=Jan/day=2
S3://sales/year=2019/month=Feb/day=1
S3://sales/year=2019/month=Feb/day=2
```

크롤러 대상이 Sales로 설정되고 day=*n* 폴더의 모든 파일이 동일한 포맷(예: JSON, 암호화되지 않음)이고 동일하거나 매우 유사한 스키마를 갖는다고 가정합니다. 크롤러는 파티션 키가 year, month 및 day인 4개의 파티션이 있는 단일 테이블을 생성합니다.

다음 예에서는 다음과 같은 Amazon S3 구조를 고려합니다.

```
s3://bucket01/folder1/table1/partition1/file.txt
s3://bucket01/folder1/table1/partition2/file.txt
s3://bucket01/folder1/table1/partition3/file.txt
s3://bucket01/folder1/table2/partition4/file.txt
s3://bucket01/folder1/table2/partition5/file.txt
```

table1과 table2 아래의 파일 스키마가 유사하고 [포함 경로(Include path)]가 s3://bucket01/folder1/인 크롤러에 데이터 스토어 1개가 정의된 경우, 크롤러는 파티션 키 열 2개로 테이블 하나를 생성합니다. 첫 번째 파티션 키 열에는 table1과 table2기 포함되고 두 번째 파티션 키 열에는 table1 파티션의 경우 partition1~partition3, table2 파티션의 경우 partition4 및 partition5가 포함됩니다. 두 데이터 스토어로 크롤러를 정의하여 두 개별 테이블을 생성합니다. 이 예제에서는 첫 번째 Include path(추가 경로)를 s3://bucket01/folder1/table1/로 두 번째는 s3://bucket01/folder1/table2로 정의합니다.

> **Note**
>
> Amazon Athena에서 각 테이블은 모든 객체가 들어 있는 Amazon S3 접두사에 해당합니다. 객체들이 다른 스키마를 가지고 있으면 Athena는 동일한 접두사 내 다른 객체를 다른 테이블로 인식하지 못합니다. 크롤러가 동일한 Amazon S3 접두사의 여러 테이블을 생성하면 이와 같은 현상이 발생할 수 있습니다. 이는 어떤 결과 없이 Athena의 쿼리로 이끕니다. Athena가 테이블을 올바르게 인식하고 쿼리할 수 있도록 Amazon S3 폴더 구조에서 서로 다른 테이블 스키마마다 별도의 [포함 경로(Include path)]를 사용하여 크롤러를 생성합니다. 자세한 내용은 AWS Glue와 함께 Athena를 사용할 때의 모범 사례와 이 AWS 지식 센터 문서를 참조하세요.

크롤링에 지원되는 데이터 소스

크롤러는 다음과 같은 파일 기반 및 테이블 기반 데이터 스토어를 크롤할 수 있습니다.

크롤러가 사용하는 액세스 유형	데이터 스토어
네이티브 클라이언트	• Amazon Simple Storage Service(S3) • Amazon DynamoDB • Delta Lake 2.0.x • Apache Iceberg 1.5 • Apache Hudi 0.14
JDBC	Amazon Redshift Snowflake Amazon Relational Database Service(Amazon RDS) 내 또는 Amazon RDS 외부: • Amazon Aurora • MariaDB • Microsoft SQL Server • MySQL • Oracle

크롤러가 사용하는 액세스 유형	데이터 스토어
	• PostgreSQL
MongoDB 클라이언트	• MongoDB • MongoDB Atlas • Amazon DocumentDB(MongoDB 호환)

> **Note**
>
> 현재 AWS Glue는 데이터 스트림에 대한 크롤러를 지원하지 않습니다.

JDBC, MongoDB, MongoDB Atlas 및 Amazon DocumentDB(MongoDB 호환) 데이터 스토어의 경우 크롤러가 데이터 스토어에 연결하는 데 사용할 수 있는 AWS Glue 연결을 지정해야 합니다. Amazon S3의 경우 선택적으로 네트워크 유형의 연결을 지정할 수 있습니다. 연결은 자격 증명, URL, Amazon Virtual Private Cloud 정보 등의 연결 정보를 저장하는 데이터 카탈로그 객체입니다. 자세한 내용은 데이터에 연결 단원을 참조하십시오.

크롤러에서 지원하는 드라이버 버전은 다음과 같습니다.

제품	크롤러 지원 드라이버
PostgreSQL	42.2.1
Amazon Aurora	네이티브 크롤러 드라이버와 동일
MariaDB	8.0.13
Microsoft SQL Server	6.1.0
MySQL	8.0.13
Oracle	11.2.2
Amazon Redshift	4.1
Snowflake	3.13.20

제품	크롤러 지원 드라이버
MongoDB	4.7.2
MongoDB Atlas	4.7.2

다음은 다양한 데이터 스토어에 대한 참고 사항입니다.

Amazon S3

사용자 계정 또는 다른 계정에서 경로를 크롤링하도록 선택할 수 있습니다. 폴더의 모든 Amazon S3 파일이 동일한 스키마를 보유하고 있으면 크롤러는 테이블 하나를 생성합니다. 또한 Amazon S3 객체가 분할된 경우 메타데이터 테이블이 하나만 생성되고 해당 테이블에 대한 데이터 카탈로그에 파티션 정보가 추가됩니다.

Amazon S3 및 Amazon DynamoDB

크롤러는 AWS Identity and Access Management(IAM) 역할을 사용하여 데이터 스토어에 대한 액세스 권한을 보유합니다. 크롤러에 전달하는 역할은 크롤된 Amazon S3 경로 및 Amazon DynamoDB 테이블에 대한 액세스 권한을 보유해야 합니다.

Amazon DynamoDB

AWS Glue 콘솔을 사용하여 크롤러를 정의하는 경우 DynamoDB 테이블을 지정할 수 있습니다. AWS Glue API를 사용하는 경우 테이블 목록을 지정할 수 있습니다. 크롤러 실행 시간을 줄이기 위해 데이터의 작은 샘플만 크롤링하도록 선택할 수 있습니다.

Delta Lake

각 Delta Lake 데이터 스토어에 대해 해당하는 Delta 테이블을 생성하는 방식을 지정합니다.

- 기본 테이블 생성: Delta 트랜잭션 로그의 쿼리를 직접 지원하는 쿼리 엔진과 통합할 수 있습니다. 자세한 내용은 Delta Lake 테이블 쿼리하기를 참조하세요.
- Symlink 테이블 생성: 지정된 구성 파라미터를 기반으로 파티션 키로 분할된 매니페스트 파일을 포함하는 `_symlink_manifest` 폴더를 생성합니다.

Iceberg

각 Iceberg 데이터 스토어에 대해 Iceberg 테이블의 메타데이터가 포함된 Amazon S3 경로를 지정합니다. 크롤러가 Iceberg 테이블 메타데이터를 검색하면 이를 데이터 카탈로그에 등록합니다. 크롤러가 테이블을 최신 상태로 유지하도록 예약을 설정할 수 있습니다.

데이터 스토어에 대해 다음 파라미터를 정의할 수 있습니다.

- 제외: 특정 폴더를 건너뛸 수 있습니다.
- 최대 이동 깊이: 크롤러가 Amazon S3 버킷에서 크롤링할 수 있는 깊이 제한을 설정합니다. 최대 이동 깊이의 기본값은 10이고 설정할 수 있는 최대 이동 깊이는 20입니다.

Hudi

각 Hudi 데이터 스토어에 대해 Hudi 테이블의 메타데이터가 포함된 Amazon S3 경로를 지정합니다. 크롤러가 Hudi 테이블 메타데이터를 검색하면 이를 데이터 카탈로그에 등록합니다. 크롤러가 테이블을 최신 상태로 유지하도록 예약을 설정할 수 있습니다.

데이터 스토어에 대해 다음 파라미터를 정의할 수 있습니다.

- 제외: 특정 폴더를 건너뛸 수 있습니다.
- 최대 이동 깊이: 크롤러가 Amazon S3 버킷에서 크롤링할 수 있는 깊이 제한을 설정합니다. 최대 이동 깊이의 기본값은 10이고 설정할 수 있는 최대 이동 깊이는 20입니다.

> **Note**
>
> 논리적 유형으로 `millis`를 사용하는 타임스탬프 열은 Hudi 0.13.1 및 타임스탬프 유형과의 비호환성으로 인해 `bigint`로 해석됩니다. 향후 Hudi 릴리스에서 이 문제의 해결 방법이 제공될 수 있습니다.

Hudi 테이블은 다음과 같이 분류되며 각각 특정한 의미를 함축합니다.

- 쓸 때 복사(CoW): 데이터가 열 기반 형식(Parquet)으로 저장되며, 업데이트마다 쓰기 중에 새 버전의 파일을 만듭니다.
- 읽을 때 병합(MoR): 데이터가 열 기반 형식(Parquet)과 행 기반(Avro) 형식을 조합하여 저장됩니다. 업데이트는 행 기반 delta 파일에 기록되며 새 버전의 열 형식 파일을 작성할 때 필요에 따라 압축됩니다.

CoW 데이터 세트를 사용하면 레코드에 대한 업데이트가 있을 때마다 레코드가 포함된 파일이 업데이트된 값으로 다시 작성됩니다. MoR 데이터 세트를 사용하면 Hudi는 업데이트가 있을 때마다 변경된 레코드에 대한 행만 씁니다. MoR은 읽기 수행이 적고 쓰기 또는 변경이 많은 워크로드에 더 적합합니다. CoW는 자주 변경되지 않는 데이터에서 읽기 수행이 많은 워크로드에 더 적합합니다.

Hudi는 데이터 액세스를 위해 세 가지 쿼리 유형을 제공합니다.

- 스냅샷 쿼리: 지정된 커밋 또는 압축 작업 시 테이블의 최신 스냅샷을 보는 쿼리입니다. MoR 테이블의 경우 스냅샷 쿼리는 쿼리 시의 최신 파일 슬라이스의 기본 파일과 델타 파일을 병합하여 테이블의 최신 상태를 나타냅니다.

- 증분의 쿼리: 이 쿼리는 지정된 커밋 및 압축 이후 테이블에 기록된 새 데이터만 볼 수 있습니다. 이는 변경 스트림을 효과적으로 제공하여 증분 데이터 파이프라인을 사용할 수 있도록 합니다.
- 읽기 최적화 쿼리: MoR 테이블의 경우 쿼리가 압축된 최신 데이터를 표시합니다. CoW 테이블의 경우 이 쿼리는 커밋된 최신 데이터를 보여줍니다.

쓸 때 복사(CoW) 테이블의 경우 크롤러는 ReadOptimized serde `org.apache.hudi.hadoop.HoodieParquetInputFormat`을 사용하는 단일 테이블을 데이터 카탈로그에서 생성합니다.

읽을 때 병합(MoR) 테이블의 경우 크롤러는 동일한 테이블 위치에 대한 두 개의 테이블을 데이터 카탈로그에서 생성합니다.

- ReadOptimized serde `org.apache.hudi.hadoop.HoodieParquetInputFormat`을 사용하는 `_ro` 접미사가 있는 테이블.
- 스냅샷 쿼리를 허용하는 RealTime Serde를 사용하는 `_rt` 접미사가 포함된 테이블 (`org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat`).

MongoDB 및 Amazon DocumentDB(MongoDB와 호환)

MongoDB 버전 3.2 이상이 지원됩니다. 크롤러 실행 시간을 줄이기 위해 데이터의 작은 샘플만 크롤링하도록 선택할 수 있습니다.

관계형 데이터베이스

인증에는 데이터베이스 사용자 이름과 암호가 사용됩니다. 데이터베이스 엔진의 유형에 따라 데이터베이스, 스키마 및 테이블처럼 어떤 객체를 크롤할 것인지 선택합니다.

Snowflake

Snowflake JDBC 크롤러는 테이블, 외부 테이블, 뷰 및 구체화된 뷰의 크롤링을 지원합니다. 구체화된 뷰 정의는 채워지지 않습니다.

Snowflake 외부 테이블의 경우 크롤러는 Amazon S3 위치를 가리키는 경우에만 크롤링합니다. 크롤러는 테이블 스키마 이외에 Amazon S3 위치, 파일 형식 및 출력을 Data Catalog 테이블의 테이블 파라미터로 크롤링합니다. 파티션을 나눈 외부 테이블의 파티션 정보는 채워지지 않습니다.

ETL은 현재 Snowflake 크롤러를 사용하여 생성한 데이터 카탈로그 테이블에서 지원되지 않습니다.

크롤러 사전 조건

크롤러는 정의할 때 지정한 AWS Identity and Access Management(IAM) 역할의 권한을 수임합니다. IAM 역할은 데이터 스토어에서 데이터를 추출하여 데이터 카탈로그에 작성할 수 있는 권한이 있어야

합니다. AWS Glue 콘솔은 AWS Glue 보안 주체 서비스를 위한 신뢰할 수 잇는 정책과 연관된 IAM 역할만 목록에 기록합니다. 콘솔에서 IAM 역할을 IAM 정책을 통해 생성하여 크롤러에 의해 액세스된 Amazon S3 데이터 스토어로 액세스할 수 있습니다. AWS Glue에 역할을 부여하는 것에 대한 자세한 내용은 AWS Glue에 대한 자격 증명 기반 정책 단원을 참조하십시오.

> **Note**
>
> Delta Lake 데이터 스토어를 크롤링할 때는 Amazon S3 위치에 대한 읽기/쓰기 권한을 가지고 있어야 합니다.

크롤러의 경우 역할을 생성하고 다음 정책을 연결할 수 있습니다.

- 데이터 카탈로그에 필요한 권한을 부여하는 `AWSGlueServiceRole` AWS 관리형 정책
- 데이터 원본에 대한 권한을 부여하는 인라인 정책입니다.
- 역할에 대한 `iam:PassRole` 권한을 부여하는 인라인 정책입니다.

더 빠른 접근 방식은 AWS Glue 콘솔 크롤러 마법사가 역할을 생성하도록 하는 것입니다. 생성하는 역할은 특히 크롤러를 위한 것이며 `AWSGlueServiceRole` AWS 관리형 정책과 지정된 데이터 원본에 대한 필수 인라인 정책을 포함합니다.

크롤러에 대한 기존 역할을 지정하는 경우 `AWSGlueServiceRole` 정책 또는 이에 상응하는 것(또는 이 정책의 범위가 축소된 버전)과 필수 인라인 정책이 포함되어 있는지 확인합니다. 예를 들어 Amazon S3 데이터 스토어의 경우 인라인 정책은 최소한 다음과 같습니다.

```
{
   "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
              "s3:GetObject"
          ],
          "Resource": [
              "arn:aws:s3:::bucket/object*"
          ]
        }
    ]
}
```

Amazon DynamoDB 데이터 스토어의 경우 정책은 최소한 다음과 같습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:Scan"
      ],
      "Resource": [
        "arn:aws:dynamodb:region:account-id:table/table-name*"
      ]
    }
  ]
}
```

크롤러가 AWS Key Management Service(AWS KMS) 암호화 Amazon S3 데이터를 읽는 경우 IAM 역할이 AWS KMS 키에 대한 권한을 복호화해야 합니다. 자세한 내용은 2단계: AWS Glue에 대한 IAM 역할 생성 단원을 참조하십시오.

분류자 정의 및 관리

분류자는 데이터 스토어의 데이터를 읽습니다. 데이터 포맷을 인식하면 스키마를 생성합니다. 분류자는 확실성 숫자를 반환하여 포맷 인식이 어땠는지 보여줍니다.

AWS Glue에서 기본 분류자 세트를 제공하지만 사용자 지정 분류자를 생성할 수도 있습니다. AWS Glue는 크롤러 정의에 지정된 순서로 사용자 정의 분류자를 먼저 호출합니다. 사용자 분류자에서 반환된 결과에 따라 AWS Glue는 기본 설정 분류자도 시작할 수 있습니다. 분류자가 진행 중에 `certainty=1.0`을 반환하면 정확한 스키마를 생성할 수 있다는 100% 확실성을 나타냅니다. 이후 AWS Glue가 해당 분류자의 출력을 사용합니다.

어떠한 분류자도 `certainty=1.0`을 반환하지 않으면, AWS Glue는 높은 확실성이 있는 분류자 출력을 사용합니다. 어떠한 분류자도 `0.0`을 초과한 확실성을 반환하지 않으면, AWS Glue는 `UNKNOWN`의 기본 분류 문자열을 반환합니다.

언제 분류자를 사용하나요?

데이터 스토어를 크롤하여 AWS Glue Data Catalog에 메타데이터 테이블을 정의할 때 분류자를 사용합니다. 분류자의 순서가 있는 세트를 통해 크롤러를 설치할 수 있습니다. 크롤러가 분류자를 시작할

때 분류자는 데이터가 인식되었는지 결정합니다. 분류자가 데이터를 인식하지 못하거나 100% 확실성이 없다면 크롤러는 목록에서 다음 분류자를 시작하여 데이터를 인식하는지 결정합니다.

AWS Glue 콘솔을 사용한 분류자 생성에 대한 자세한 내용은 AWS Glue 콘솔을 사용하여 분류자 생성을 참조하십시오.

사용자 지정 분류자

분류자의 출력은 파일 분류 혹은 포맷(예를 들어 `json`) 및 파일 스키마를 보여주는 문자열을 포함합니다. 사용자 지정 분류자의 경우, 분류자 유형을 기반으로 스키마를 생성하는 로직을 정의합니다. 분류자 유형은 grok 패턴, XML 태그 및 JSON에 따른 스키마 정의를 포함합니다.

분류자 정의를 변경하면 분류자를 사용하여 이전에 크롤링된 어떤 데이터도 재분류가 되지 않습니다. 크롤러는 이전에 크롤링된 데이터를 계속 추적합니다. 새로운 데이터는 업데이트된 스키마 결과를 낳을 수 있는 업데이트된 분류자로 분류됩니다. 데이터 스키마가 연관되면 크롤러를 실행할 때 분류자를 계정에 업데이트하여 스키마를 변경합니다. 부정확한 분류자를 바로 잡기 위해 데이터를 재분류하려면 업데이트된 분류자로 새로운 데이터를 생성합니다.

AWS Glue의 사용자 지정 분류자를 생성하는 방법에 대한 자세한 내용은 다양한 데이터 형식에 대한 사용자 지정 분류자 작성 섹션을 참조하십시오.

> **Note**
>
> 데이터 포맷이 기본 설정 분류자 중 하나로 인식되면 사용자 지정 분류자를 생성할 필요가 없습니다.

기본 제공 분류자

AWS Glue는 JSON, CSV, 웹 로그 및 다수 데이터베이스 시스템을 포함한 다양한 포맷의 기본 설정 분류자를 제공합니다.

AWS Glue가 100% 확실성의 입력 데이터 포맷에 맞는 사용자 지정 분류자를 찾을 수 없다면 다음 테이블에 보여진 것처럼 기본 설정 분류자를 시작합니다. 기본 설정 분류자는 포맷이 일치한지(`certainty=1.0`) 일치하지 않는지(`certainty=0.0`) 보여주는 결과를 반환합니다. `certainty=1.0`이 있는 첫 번째 분류자는 Data Catalog의 메타데이터 테이블의 분류 문자열 및 스키마를 제공합니다.

분류자 유형	분류 문자열	참고
Apache Avro	`avro`	파일 시작 부분의 스키마를 읽어 포맷을 결정합니다.
Apache ORC	`orc`	파일 메타데이터를 읽어 포맷을 결정합니다.
Apache Parquet	`parquet`	파일 끝 부분의 스키마를 읽어 포맷을 결정합니다.
JSON	`json`	파일 시작 부분을 읽어 포맷을 결정합니다.
이진수 JSON	`bson`	파일 시작 부분을 읽어 포맷을 결정합니다.
XML	`xml`	파일 시작부를 읽어 형식을 확인합니다. AWS Glue는 문서의 XML 태그에 기반하여 테이블 스키마를 확인합니다. 문서의 행을 지정하기 위한 사용자 지정 분류자를 생성하는 방법에 대한 자세한 내용은 XML 사용자 지정 분류자 작성 단원을 참조하십시오.
Amazon Ion	`ion`	파일 시작 부분을 읽어 포맷을 결정합니다.
통합된 Apache Log	`combined_apache`	grok 패턴을 통해 로그 포맷을 결정합니다.
Apache log	`apache`	grok 패턴을 통해 로그 포맷을 결정합니다.
Linux kernel log	`linux_kernel`	grok 패턴을 통해 로그 포맷을 결정합니다.
Microsoft log	`microsoft_log`	grok 패턴을 통해 로그 포맷을 결정합니다.
Ruby log	`ruby_logger`	파일 시작 부분을 읽어 포맷을 결정합니다.
Squid 3.x log	`squid`	파일 시작 부분을 읽어 포맷을 결정합니다.
Redis monitor log	`redismonlog`	파일 시작 부분을 읽어 포맷을 결정합니다.
Redis log	`redislog`	파일 시작 부분을 읽어 포맷을 결정합니다.

분류자 유형	분류 문자열	참고
CSV	`csv`	콤마(,) 혹은 파이프(|), 탭(\t), 세미콜론(;), Ctrl+A(\u0001)의 다음 구획 문자를 확인합니다. Ctrl-A는 `Start Of Heading`용 유니코드 관리 문자입니다.
Amazon Redshift	`redshift`	JDBC 연결을 사용하여 메타데이터를 가져옵니다.
MySQL	`mysql`	JDBC 연결을 사용하여 메타데이터를 가져옵니다.
PostgreSQL	`postgresql`	JDBC 연결을 사용하여 메타데이터를 가져옵니다.
Oracle 데이터베이스	`oracle`	JDBC 연결을 사용하여 메타데이터를 가져옵니다.
Microsoft SQL Server	`sqlserver`	JDBC 연결을 사용하여 메타데이터를 가져옵니다.
Amazon DynamoDB	`dynamodb`	DynamoDB 테이블에서 데이터를 읽습니다.

다음 형식의 파일은 분류될 수 있습니다.

- ZIP(단일 파일만 있는 아카이브에 지원됨). 다른 서비스에서는 Zip이 잘 지원되지 않습니다(아카이브 때문에).
- bzip
- GZIP
- LZ4
- Snappy(표준 및 Hadoop 네이티브 Snappy 형식 모두에서 지원됨)

기본 설정 CSV 분류자

기본 설정 CSV 분류자는 CSV 파일 내용을 파싱하여 AWS Glue 테이블 스키마를 결정합니다. 이 분류자는 다음 구분 기호를 확인합니다.

- 쉼표(,)
- 파이프(|)

- 탭(\t)
- 세미콜론(;)
- Ctrl-A (\u0001)

  Ctrl-A는 `Start Of Heading`용 유니코드 관리 문자입니다.

CSV로써 분류되려면 테이블 스키마는 적어도 데이터의 두 개의 열과 두 개의 행이 있어야 합니다. CSV 분류자는 휴리스틱 숫자를 사용하여 헤더가 주어진 파일에 존재하는지 결정합니다. 분류자가 데이터의 첫 번째 행에서 헤더를 결정하지 못하면 열 헤더는 `col1`, `col2`, `col3` 등으로 보여집니다. 기본 설정 CSV 분류자는 다음 파일의 특성을 평가하여 헤더를 추론할지 여부를 결정합니다.

- 잠재적 헤더의 모든 열은 STRING(문자열) 데이터 유형으로 파싱합니다.
- 마지막 열을 제외하면 잠재적 헤더의 모든 열은 150 문자보다 적습니다. 후행 구분 기호를 허용하기 위해서는 마지막 열이 파일 전체로 비어 있을 수 있습니다.
- 잠재적 헤더의 모든 열은 열 이름 때문에 AWS Glue `regex` 요구 사항과 일치해야 합니다.
- 헤더는 데이터 행과 많이 달라야 합니다. 이를 결정하기 위해서는 하나 이상의 행이 STRING(문자열) 유형 아닌 것으로써 파싱해야 합니다. 모든 열이 문자열 유형이면 데이터 첫 번째 행은 헤더로써 사용되는 후속 행과 많이 다르지 않습니다.

> **Note**
>
> 기본 설정 CSV 분류자는 원하는 AWS Glue 테이블을 생성하지 않으면 다음 대안 중 하나 이상을 사용할 수 있습니다.
>
> - Data Catalog의 열 이름을 변경하고 `SchemaChangePolicy`를 LOG에 설정하고 파티션 출력 구성을 `InheritFromTable`로 설정하여 추후에 실행할 크롤러에 대비합니다.
> - 사용자 지정 grok 분류자를 생성하여 데이터를 파싱하고 원하는 열을 지정합니다.
> - 기본 설정 CSV 분류자는 유형 추론의 좋은 선택인 직렬화 라이브러리로써 `LazySimpleSerDe`를 참조하는 테이블을 생성합니다. 하지만 CSV 데이터가 따옴표로 묶인 문자열을 포함하면 테이블 정의를 편집하고 SerDe 라이브러리를 `OpenCSVSerDe`로 변경합니다. 추론된 유형을 STRING(문자열)로 조절하고 `SchemaChangePolicy`를 로그에 설치하고 파티션 출력 구성을 `InheritFromTable`로 설치하여 추후에 실행할 크롤러에 대비합니다. SerDe 라이브러리에 대한 자세한 내용은 Amazon Athena User Guide의 SerDe Reference를 참조하세요.

다양한 데이터 형식에 대한 사용자 지정 분류자 작성

AWS Glue에서 데이터를 분류할 수 있도록 사용자 지정 분류자를 제공할 수 있습니다. You can create a custom classifier using a grok 패턴, XML 태그, JavaScript Object Notation(JSON) 또는 CSV(쉼표로 분리된 값)를 이용하여 사용자 지정 분류자를 생성할 수 있습니다. AWS Glue 크롤러가 사용자 지정 분류자를 호출합니다. 분류자가 데이터를 인식하면 크롤러에 데이터 분류 및 스키마를 반환합니다. 데이터가 내장된 분류자와 일치하지 않거나 크롤러가 생성한 테이블을 사용자 지정하기 원할 경우 사용자 지정 분류자를 정의해야 할 수도 있습니다.

AWS Glue 콘솔을 사용한 분류자 생성에 대한 자세한 내용은 AWS Glue 콘솔을 사용하여 분류자 생성을 참조하십시오.

AWS Glue는 사용자가 지정한 순서대로 내장된(기본) 분류자에 앞서 사용자 지정 분류자를 실행합니다. 크롤러가 데이터와 일치하는 분류자를 찾으면, 사용자의 AWS Glue Data Catalog에 작성되는 테이블 정의에 분류 문자열과 스키마를 사용합니다.

주제

- Grok 사용자 지정 분류자 작성
- XML 사용자 지정 분류자 작성
- JSON 사용자 지정 분류자 작성
- CSV 사용자 지정 분류자 작성

Grok 사용자 지정 분류자 작성

Grok는 텍스트 데이터의 패턴 일치 구문 분석에 사용하는 도구입니다. Grok 패턴은 한 번에 한 줄씩 데이터를 일치시킬 때 사용하는 명명된 정규식(regex) 세트입니다. AWS Glue는 grok 패턴을 사용하여 사용자 데이터의 스키마를 추정합니다. Grok 패턴과 사용자 데이터가 일치하는 경우 AWS Glue는 이 패턴을 사용해 데이터의 구조를 결정하고 이를 필드로 매핑합니다.

AWS Glue는 많은 기본 패턴을 제공하지만, 사용자가 직접 정의할 수도 있습니다. 기본 패턴을 사용하여 grok 패턴을 생성하거나 사용자 지정 분류자 정의에서 패턴을 사용자 지정할 수 있습니다. 사용자 지정 텍스트 파일 형식을 분류할 수 있도록 grok 패턴을 조정할 수 있습니다.

> **Note**
>
> AWS Glue grok 사용자 지정 분류자는 AWS Glue Data Catalog에서 생성된 테이블에 대해 `GrokSerDe` 직렬화 라이브러리를 사용합니다. AWS Glue Data Catalog를 Amazon Athena, Amazon EMR 또는 Redshift Spectrum에 사용하려는 경우 이러한 서비스에 대한 설명서에서

GrokSerDe 지원에 대한 정보를 확인하세요. 현재는 Amazon EMR 및 Redshift Spectrum에서 GrokSerDe로 생성된 테이블을 쿼리할 때 문제가 발생할 수 있습니다.

다음은 grok 패턴 구성 요소의 기본적인 구문입니다.

```
%{PATTERN:field-name}
```

명명된 PATTERN과 일치하는 데이터를 string이라는 기본 데이터 유형을 가진 스키마의 field-name 열로 매핑합니다. 선택적으로, 필드의 데이터 형식을 결과 스키마에서 byte, boolean, double, short, int, long 또는 float로 캐스트할 수 있습니다.

```
%{PATTERN:field-name:data-type}
```

예를 들면, num 필드를 int 데이터 유형으로 캐스트하기 위해 이 패턴을 사용할 수 있습니다.

```
%{NUMBER:num:int}
```

패턴을 다른 패턴으로 구성할 수 있습니다. 예를 들어 월, 해당 월의 날, 시간(예: Feb 1 06:25:43)의 패턴으로 정의되는 SYSLOG 타임스탬프의 패턴이 있을 수 있습니다. 이 데이터에서 다음 패턴을 정의할 수 있습니다.

```
SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
```

Note

Grok 패턴은 한 번에 한 줄씩만 처리할 수 있습니다. 여러 줄로 된 패턴을 지원하지 않습니다. 또 패턴 내부의 줄 바꿈도 지원하지 않습니다.

grok 분류자의 사용자 지정 값

grok 분류자를 정의할 때 사용자 지정 분류자를 생성하기 위해 다음 값을 제공합니다.

명칭

분류자의 이름.

분류

`special-logs` 같이 분류되는 데이터의 형식을 설명하기 위해 작성하는 텍스트 문자열입니다.

Grok 패턴

일치 여부를 결정하기 위해 데이터 스토어에 적용하는 패턴 세트입니다. 이 패턴은 AWS Glue 기본 패턴 및 사용자가 정의한 사용자 지정 패턴입니다.

다음은 grok 패턴의 예입니다.

```
%{TIMESTAMP_ISO8601:timestamp} \[%{MESSAGEPREFIX:message_prefix}\]
 %{CRAWLERLOGLEVEL:loglevel} : %{GREEDYDATA:message}
```

데이터가 `TIMESTAMP_ISO8601`와 일치하면 스키마 열인 `timestamp`가 생성됩니다. 동작은 예제의 다른 명명 패턴과 유사합니다.

사용자 지정 패턴

선택적으로 사용자가 정의한 사용자 지정 패턴입니다. 사용자 데이터를 분류하는 grok 패턴이 이 패턴을 참조합니다. 데이터에 적용되는 grok 패턴에서 이런 사용자 지정 패턴을 참조시킬 수 있습니다. 각 사용자 지정 구성 요소 패턴은 별개 줄로 구성되어 있어야 합니다. 정규식(regex) 구문을 사용하여 패턴을 정의합니다.

다음은 사용자 지정 패턴 사용에 대한 예입니다.

```
CRAWLERLOGLEVEL (BENCHMARK|ERROR|WARN|INFO|TRACE)
MESSAGEPREFIX .*-.*-.*-.*-.*
```

첫 번째 사용자 지정 명명 패턴인 `CRAWLERLOGLEVEL`은 데이터가 열거된 문자열 중 하나와 일치할 때 일치됩니다. 두 번째 사용자 지정 패턴인 `MESSAGEPREFIX`은 메시지 접두사 문자열과 일치를 시도합니다.

AWS Glue는 계속해서 생성 시간, 마지막 업데이트 시간, 분류자 버전을 추적합니다.

기본 제공 패턴

AWS Glue는 사용자 지정 분류자 구축에 사용할 수 있는 많은 범용 패턴을 제공합니다. 분류자 정의의 `grok pattern`에 명명 패턴을 추가합니다.

다음 목록은 각 패턴이 한 줄씩 구성되어 있습니다. 각 줄의 패턴 이름에는 정의가 적용됩니다. 정규식(regex) 구문은 패턴을 정의하는 데 사용됩니다.

```
#<noloc>&GLU;</noloc> Built-in patterns
 USERNAME [a-zA-Z0-9._-]+
 USER %{USERNAME:UNWANTED}
 INT (?:[+-]?(?:[0-9]+))
 BASE10NUM (?<![0-9.+-])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+)))
 NUMBER (?:%{BASE10NUM:UNWANTED})
 BASE16NUM (?<![0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
 BASE16FLOAT \b(?<![0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[0-9A-Fa-f]*)?)|
(?:\.[0-9A-Fa-f]+)))\b
 BOOLEAN (?i)(true|false)

 POSINT \b(?:[1-9][0-9]*)\b
 NONNEGINT \b(?:[0-9]+)\b
 WORD \b\w+\b
 NOTSPACE \S+
 SPACE \s*
 DATA .*?
 GREEDYDATA .*
 #QUOTEDSTRING (?:(?<!\\)(?:"(?:\\.|[^\\"])*"|(?:'(?:\\.|[^\\'])*')|(?:`(?:\\.|[^\
\`])*`)))
 QUOTEDSTRING (?>(?<!\\)(?>"(?>\\.|[^\\"]+)+"|""|(?>'(?>\\.|[^\\']+)+')|''|(?>`(?>\\.|
[^\\`]+)+`)|``))
 UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}

 # Networking
 MAC (?:%{CISCOMAC:UNWANTED}|%{WINDOWSMAC:UNWANTED}|%{COMMONMAC:UNWANTED})
 CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2}[A-Fa-f0-9]{4})
 WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
 COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
 IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-
Fa-f]{1,4}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))
{3})|:))|(([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d
\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}(((:
[0-9A-Fa-f]{1,4}){1,3})|((:[0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.
(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]
{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,2}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|
2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){2}(((:[0-9A-Fa-f]{1,4}){1,5})|
((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d
\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-Fa-f]{1,4}){1,6})|((:[0-9A-Fa-
f]{1,4}){0,4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))
```

```
{3}))|:))|(:(((:[0-9A-Fa-f]{1,4}){1,7})|((:[0-9A-Fa-f]{1,4}){0,5}:((25[0-5]|2[0-4]\d|
1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:)))(%.+)?
 IPV4 (?<![0-9])(?:(?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|
[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|
[0-1]?[0-9]{1,2}))(?![0-9])
 IP (?:%{IPV6:UNWANTED}|%{IPV4:UNWANTED})
 HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-_]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-_]
{0,62}))*(\.?|\b)
 HOST %{HOSTNAME:UNWANTED}
 IPORHOST (?:%{HOSTNAME:UNWANTED}|%{IP:UNWANTED})
 HOSTPORT (?:%{IPORHOST}:%{POSINT:PORT})

 # paths
 PATH (?:%{UNIXPATH}|%{WINPATH})
 UNIXPATH (?>/(?>[\w_%!$@:.,~-]+|\\.)*)+
 #UNIXPATH (?<![\w\/])(?:/[^\/\s?*]*)+
 TTY (?:/dev/(pts|tty([pq])?)(\w+)?/?(?:[0-9]+))
 WINPATH (?>[A-Za-z]+:|\\)(?:\\[^\\?*]*)+
 URIPROTO [A-Za-z]+(\+[A-Za-z+]+)?
 URIHOST %{IPORHOST}(?::%{POSINT:port})?
 # uripath comes loosely from RFC1738, but mostly from what Firefox
 # doesn't turn into %XX
 URIPATH (?:/[A-Za-z0-9$.+!*'(){},~:;=@#%_\-]*)+
 #URIPARAM \?(?:[A-Za-z0-9]+(?:=(?:[^&]*))?(?:&(?:[A-Za-z0-9]+(?:=(?:[^&]*))?)?)*)?
 URIPARAM \?[A-Za-z0-9$.+!*'|(){},~@#%&/=:;_?\-\[\]]*
 URIPATHPARAM %{URIPATH}(?:%{URIPARAM})?
 URI %{URIPROTO}://(?:%{USER}(?::[^@]*)?@)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?

 # Months: January, Feb, 3, 03, 12, December
 MONTH \b(?:Jan(?:uary)?|Feb(?:ruary)?|Mar(?:ch)?|Apr(?:il)?|May|Jun(?:e)?|Jul(?:y)?|
Aug(?:ust)?|Sep(?:tember)?|Oct(?:ober)?|Nov(?:ember)?|Dec(?:ember)?)\b
 MONTHNUM (?:0?[1-9]|1[0-2])
 MONTHNUM2 (?:0[1-9]|1[0-2])
 MONTHDAY (?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])

 # Days: Monday, Tue, Thu, etc...
 DAY (?:Mon(?:day)?|Tue(?:sday)?|Wed(?:nesday)?|Thu(?:rsday)?|Fri(?:day)?|
Sat(?:urday)?|Sun(?:day)?)

 # Years?
 YEAR (?>\d\d){1,2}
 # Time: HH:MM:SS
 #TIME \d{2}:\d{2}(?::\d{2}(?:\.\d+)?)?
 # TIME %{POSINT<24}:%{POSINT<60}(?::%{POSINT<60}(?:\.%{POSINT})?)?
```

```
 HOUR (?:2[0123]|[01]?[0-9])
 MINUTE (?:[0-5][0-9])
 # '60' is a leap second in most time standards and thus is valid.
 SECOND (?:(?:[0-5]?[0-9]|60)(?:[:.,][0-9]+)?)
 TIME (?!<[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![0-9])
 # datestamp is YYYY/MM/DD-HH:MM:SS.UUUU (or something like it)
 DATE_US %{MONTHNUM}[/-]%{MONTHDAY}[/-]%{YEAR}
 DATE_EU %{MONTHDAY}[./-]%{MONTHNUM}[./-]%{YEAR}
 DATESTAMP_US %{DATE_US}[- ]%{TIME}
 DATESTAMP_EU %{DATE_EU}[- ]%{TIME}
 ISO8601_TIMEZONE (?:Z|[+-]%{HOUR}(?::?%{MINUTE}))
 ISO8601_SECOND (?:%{SECOND}|60)
 TIMESTAMP_ISO8601 %{YEAR}-%{MONTHNUM}-%{MONTHDAY}[T ]%{HOUR}:?%{MINUTE}(?::?
%{SECOND})?%{ISO8601_TIMEZONE}?
 TZ (?:[PMCE][SD]T|UTC)
 DATESTAMP_RFC822 %{DAY} %{MONTH} %{MONTHDAY} %{YEAR} %{TIME} %{TZ}
 DATESTAMP_RFC2822 %{DAY}, %{MONTHDAY} %{MONTH} %{YEAR} %{TIME} %{ISO8601_TIMEZONE}
 DATESTAMP_OTHER %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{TZ} %{YEAR}
 DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}
 CISCOTIMESTAMP %{MONTH} %{MONTHDAY} %{TIME}

 # Syslog Dates: Month Day HH:MM:SS
 SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
 PROG (?:[\w._/%-]+)
 SYSLOGPROG %{PROG:program}(?:\[%{POSINT:pid}\])?
 SYSLOGHOST %{IPORHOST}
 SYSLOGFACILITY <%{NONNEGINT:facility}.%{NONNEGINT:priority}>
 HTTPDATE %{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT}

 # Shortcuts
 QS %{QUOTEDSTRING:UNWANTED}

 # Log formats
 SYSLOGBASE %{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY} )?%{SYSLOGHOST:logsource}
 %{SYSLOGPROG}:

 MESSAGESLOG %{SYSLOGBASE} %{DATA}

 COMMONAPACHELOG %{IPORHOST:clientip} %{USER:ident} %{USER:auth}
 \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/
%{NUMBER:httpversion})?|%{DATA:rawrequest})" %{NUMBER:response} (?:%{Bytes:bytes=
%{NUMBER}|-})
 COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}
```

```
 COMMONAPACHELOG_DATATYPED %{IPORHOST:clientip} %{USER:ident;boolean} %{USER:auth}
 \[%{HTTPDATE:timestamp;date;dd/MMM/yyyy:HH:mm:ss Z}\] "(?:%{WORD:verb;string}
 %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion;float})?|%{DATA:rawrequest})"
 %{NUMBER:response;int} (?:%{NUMBER:bytes;long}|-)


 # Log Levels
 LOGLEVEL ([A|a]lert|ALERT|[T|t]race|TRACE|[D|d]ebug|DEBUG|[N|n]otice|NOTICE|[I|i]nfo|
INFO|[W|w]arn?(?:ing)?|WARN?(?:ING)?|[E|e]rr?(?:or)?|ERR?(?:OR)?|[C|c]rit?(?:ical)?|
CRIT?(?:ICAL)?|[F|f]atal|FATAL|[S|s]evere|SEVERE|EMERG(?:ENCY)?|[Ee]merg(?:ency)?)
```

XML 사용자 지정 분류자 작성

XML는 파일에서 태그를 사용해 문서의 구조를 정의합니다. XML 사용자 지정 분류자의 경우 사용자가 행을 정의할 때 사용하는 태그 이름을 지정할 수 있습니다.

XML 분류자의 사용자 지정 분류자 값

XML 분류자를 정의할 때 사용자 지정 분류자를 생성하기 위해 AWS Glue에 다음 값을 제공합니다. 이 분류자의 분류 필드를 `xml`으로 설정합니다.

명칭

분류자의 이름.

행 태그

XML 문서의 테이블 행을 정의하는 XML 태그 이름입니다. 꺾쇠 괄호 `< >`는 사용하지 않습니다. 이름은 태그에 대한 XML 규칙을 따라야 합니다.

> **Note**
>
> 행 데이터가 포함된 요소는 자기 복제를 하는 비어 있는 요소가 될 수 없습니다. 예를 들어 이 빈 요소는 AWS Glue에서 구문 분석되지 않습니다.
>
> ```
> <row att1=”xx” att2=”yy” />
> ```
>
> 비어 있는 요소를 다음과 같이 작성할 수 없습니다.

```
<row att1=”xx” att2=”yy”> </row>
```

AWS Glue는 계속해서 생성 시간, 마지막 업데이트 시간, 분류자 버전을 추적합니다.

예를 들어 다음과 같은 XML 파일이 있다고 가정하겠습니다. 작성자와 제목 열만 포함된 AWS Glue 테이블을 생성하려면 AWS Glue 콘솔에서 Row tag(행 태그)가 AnyCompany인 분류자를 생성합니다. 그런 다음 이 사용자 지정 분류자를 사용하는 크롤러를 추가해 실행합니다.

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
     <AnyCompany>
       <author>Rivera, Martha</author>
       <title>AnyCompany Developer Guide</title>
     </AnyCompany>
   </book>
   <book id="bk102">
     <AnyCompany>
       <author>Stiles, John</author>
       <title>Style Guide for AnyCompany</title>
     </AnyCompany>
   </book>
</catalog>
```

JSON 사용자 지정 분류자 작성

JSON은 데이터 교환 형식입니다. 이름 값 쌍이나 순서가 지정된 값 목록을 가진 데이터 구조를 정의합니다. JSON 사용자 지정 분류자의 경우 데이터 구조에 JSON 경로를 지정해 사용자 테이블의 스키마를 정의할 수 있습니다.

AWS Glue의 사용자 지정 분류자 값

JSON 분류자를 정의할 때 사용자 지정 분류자를 생성하기 위해 AWS Glue에 다음 값을 제공합니다. 이 분류자의 분류 필드를 `json`으로 설정합니다.

명칭

분류자의 이름.

JSON 경로

테이블 스키마 정의에 사용하는 객체를 가리키는 JSON 경로입니다. JSON 경로 작성 시 점이나 대괄호를 사용할 수 없습니다. 다음의 연산자가 지원됩니다:

설명
JSON 객체의 루트 요소입니다. 모든 경로 표현식은
와일드카드 문자로 시작됩니다. JSON 경로의 경우 위치와 무관하게 이름이나 숫자를 사용해야 합니다.
점으로 표기된 하위 필드. JSON 객체의 하위 필드를 지정합니다.
괄호로 표기된 하위 필드. JSON 객체의 하위 필드를 지정합니다. 하위 필드 1개만 지정할 수 있습니다.
어레이 인덱스. 인덱스 별로 어레이 값을 지정합니다.

AWS Glue는 계속해서 생성 시간, 마지막 업데이트 시간, 분류자 버전을 추적합니다.

Example 배열에서 레코드를 가져오는 데 JSON 분류자 사용

JSON 데이터가 레코드 배열이라고 가정하겠습니다. 예를 들어, 파일의 처음 몇 줄은 다음과 비슷할 수 있습니다.

```
[
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ak",
    "name": "Alaska"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:1",
    "name": "Alabama's 1st congressional district"
  },
```

```
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:2",
    "name": "Alabama's 2nd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:3",
    "name": "Alabama's 3rd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:4",
    "name": "Alabama's 4th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:5",
    "name": "Alabama's 5th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:6",
    "name": "Alabama's 6th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:7",
    "name": "Alabama's 7th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:1",
    "name": "Arkansas's 1st congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:2",
    "name": "Arkansas's 2nd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:3",
    "name": "Arkansas's 3rd congressional district"
```

```
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:4",
    "name": "Arkansas's 4th congressional district"
  }
]
```

기본 JSON 분류자를 사용하는 크롤러를 실행할 때는 스키마 정의에 전체 파일이 사용됩니다. JSON 경로를 지정하지 않기 때문에 크롤러는 데이터를 하나의 객체로 처리하며, 이것이 배열입니다. 예를 들어, 스키마의 모양은 다음과 같을 수 있습니다.

```
root
|-- record: array
```

하지만 JSON 어레이의 각 레코드를 토대로 스키마를 생성하려면 사용자 지정 JSON 분류자를 생성하고 JSON 경로를 `$[*]`로 지정합니다. 이 JSON 경로를 지정하는 경우, 분류자는 어레이의 12개 레코드를 모두 조사해 스키마를 결정합니다. 그 결과로 생성된 스키마에는 다음 예제와 유사하게 각 객체의 필드가 분리되어 포함됩니다.

```
root
|-- type: string
|-- id: string
|-- name: string
```

Example 파일의 일부만을 검사하는 데 JSON 분류자 사용

JSON 데이터가 http://everypolitician.org/에서 가져온 예제 JSON 파일 `s3://awsglue-datasets/examples/us-legislators/all/areas.json`의 패턴을 따른다고 가정합니다. JSON 파일의 예제 객체는 다음과 같습니다.

```
{
  "type": "constituency",
  "id": "ocd-division\/country:us\/state:ak",
  "name": "Alaska"
}
{
  "type": "constituency",
```

```
  "identifiers": [
    {
      "scheme": "dmoz",
      "identifier": "Regional\/North_America\/United_States\/Alaska\/"
    },
    {
      "scheme": "freebase",
      "identifier": "\/m\/0hjy"
    },
    {
      "scheme": "fips",
      "identifier": "US02"
    },
    {
      "scheme": "quora",
      "identifier": "Alaska-state"
    },
    {
      "scheme": "britannica",
      "identifier": "place\/Alaska"
    },
    {
      "scheme": "wikidata",
      "identifier": "Q797"
    }
  ],
  "other_names": [
    {
      "lang": "en",
      "note": "multilingual",
      "name": "Alaska"
    },
    {
      "lang": "fr",
      "note": "multilingual",
      "name": "Alaska"
    },
    {
      "lang": "nov",
      "note": "multilingual",
      "name": "Alaska"
    }
  ],
  "id": "ocd-division\/country:us\/state:ak",
```

```
  "name": "Alaska"
}
```

기본 JSON 분류자를 사용하는 크롤러를 실행할 때는 스키마를 생성하는 데 전체 파일이 사용됩니다. 그러면 다음과 같은 스키마가 생성될 수 있습니다.

```
root
|-- type: string
|-- id: string
|-- name: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
```

하지만 "id" 객체만을 사용하는 스키마를 생성하려면 사용자 지정 JSON 분류자를 생성하고 JSON 경로를 $.id로 지정합니다. 그러면 스키마는 "id" 필드에만 토대를 둡니다.

```
root
|-- record: string
```

이 스키마에서 추출한 데이터의 첫 몇 줄은 다음과 같습니다.

```
{"record": "ocd-division/country:us/state:ak"}
{"record": "ocd-division/country:us/state:al/cd:1"}
{"record": "ocd-division/country:us/state:al/cd:2"}
{"record": "ocd-division/country:us/state:al/cd:3"}
{"record": "ocd-division/country:us/state:al/cd:4"}
{"record": "ocd-division/country:us/state:al/cd:5"}
{"record": "ocd-division/country:us/state:al/cd:6"}
{"record": "ocd-division/country:us/state:al/cd:7"}
{"record": "ocd-division/country:us/state:ar/cd:1"}
{"record": "ocd-division/country:us/state:ar/cd:2"}
```

```
{"record": "ocd-division/country:us/state:ar/cd:3"}
{"record": "ocd-division/country:us/state:ar/cd:4"}
{"record": "ocd-division/country:us/state:as"}
{"record": "ocd-division/country:us/state:az/cd:1"}
{"record": "ocd-division/country:us/state:az/cd:2"}
{"record": "ocd-division/country:us/state:az/cd:3"}
{"record": "ocd-division/country:us/state:az/cd:4"}
{"record": "ocd-division/country:us/state:az/cd:5"}
{"record": "ocd-division/country:us/state:az/cd:6"}
{"record": "ocd-division/country:us/state:az/cd:7"}
```

JSON 파일의 "identifier" 같이 중첩된 객체를 토대로 스키마를 생성하려면 사용자 지정 JSON 분류자를 생성하고 JSON 경로를 $.identifiers[*].identifier로 지정합니다. 이 스키마는 이전 예제와 유사하지만, JSON 파일에서 전혀 다른 객체에 토대를 두고 있습니다.

이런 스키마는 다음과 같습니다.

```
root
|-- record: string
```

테이블에서 데이터의 첫 몇 줄은 스키마가 "identifier" 객체에 토대를 두고 있다는 점을 보여줍니다.

```
{"record": "Regional/North_America/United_States/Alaska/"}
{"record": "/m/0hjy"}
{"record": "US02"}
{"record": "5879092"}
{"record": "4001016-8"}
{"record": "destination/alaska"}
{"record": "1116270"}
{"record": "139487266"}
{"record": "n79018447"}
{"record": "01490999-8dec-4129-8254-eef6e80fadc3"}
{"record": "Alaska-state"}
{"record": "place/Alaska"}
{"record": "Q797"}
{"record": "Regional/North_America/United_States/Alabama/"}
{"record": "/m/0gyh"}
{"record": "US01"}
{"record": "4829764"}
```

```
{"record": "4084839-5"}
{"record": "161950"}
{"record": "131885589"}
```

JSON 파일의 "name" 배열의 "other_names" 필드 같이 다른 중첩된 객체를 토대로 테이블을 생성하려면 사용자 지정 JSON 분류자를 생성하고 JSON 경로를 $.other_names[*].name로 지정합니다. 이 스키마는 이전 예제와 유사하지만, JSON 파일에서 전혀 다른 객체에 토대를 두고 있습니다. 이런 스키마는 다음과 같습니다.

```
root
|-- record: string
```

이 테이블의 처음 몇 줄 데이터는 "name" 배열 내 "other_names" 객체의 데이터에 근거함을 보여줍니다.

```
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Аляска"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "######"}
{"record": "######"}
{"record": "######"}
{"record": "Alaska"}
{"record": "Alyaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Штат Аляска"}
{"record": "Аляска"}
{"record": "Alaska"}
{"record": "#######"}
```

CSV 사용자 지정 분류자 작성

사용자 지정 CSV 분류자를 사용하면 사용자 지정 csv 분류자 필드의 각 열에 대한 데이터 형식을 지정할 수 있습니다. 콤마로 분리된 각 열의 데이터 유형을 지정할 수 있습니다. 데이터 유형을 지정하면 크롤러가 추론한 데이터 유형을 재정의하고 데이터가 적절하게 분류되도록 할 수 있습니다.

분류자에서 CSV를 처리하기 위한 Serde를 설정할 수 있습니다. 이는 데이터 카탈로그에서 적용됩니다.

사용자 지정 분류기를 만들면 분류기를 다른 크롤러에 다시 사용할 수도 있습니다.

- 헤더만 있는(데이터 없음) csv 파일의 경우, 정보가 충분하지 않기 때문에 이러한 파일은 UNKNOWN으로 분류됩니다. 열 제목 옵션에서 CSV에 '제목 포함'을 지정하고 데이터 유형을 제공하면 이러한 파일을 올바르게 분류할 수 있습니다.

사용자 지정 CSV 분류자를 사용하여 다양한 유형의 CSV 데이터 스키마를 추론할 수 있습니다. 분류자에 제공할 수 있는 사용자 지정 속성에는 구분 기호, CSV SerDe 옵션, 헤더에 대한 옵션, 데이터에서 특정 검증을 수행할 수 있는지 여부가 포함되어 있습니다.

AWS Glue의 사용자 지정 분류자 값

CSV 분류자를 정의할 때 분류자를 생성하기 위해 AWS Glue에 다음 값을 제공합니다. 이 분류자의 분류 필드를 `csv`으로 설정합니다.

분류자 이름

분류자의 이름.

CSV Serde

분류자에서 CSV를 처리하기 위한 Serde를 설정합니다. 이는 데이터 카탈로그에서 적용됩니다. 옵션은 Open CSV SerDe, Lazy Simple SerDe, 없음과 같습니다. 크롤러에서 감지하려는 경우 없음 값을 지정할 수 있습니다.

열 구분 기호

행의 열 입력 항목 각각을 구분하는 것을 나타내기 위한 사용자 지정 기호입니다. 유니코드 문자를 제공합니다. 구분 기호를 입력할 수 없는 경우 구분 기호를 복사하여 붙여넣을 수 있습니다. 이 방법은 시스템에서 지원하지 않는 문자(일반적으로 □로 표시)를 포함하여 인쇄 가능한 문자에도 사용할 수 있습니다.

인용 기호

단일 열 값에 내용을 결합하는 것을 나타내기 위한 사용자 지정 기호입니다. 열 구분 기호와 달라야 합니다. 유니코드 문자를 제공합니다. 구분 기호를 입력할 수 없는 경우 구분 기호를 복사하여 붙여 넣을 수 있습니다. 이 방법은 시스템에서 지원하지 않는 문자(일반적으로 □로 표시)를 포함하여 인쇄 가능한 문자에도 사용할 수 있습니다.

열 제목

열 제목을 CSV 파일에서 어떻게 탐지해야 하는지에 대한 행동을 표시합니다. 사용자 지정 CSV 파일에 열 제목이 포함되어 있는 경우에는 쉼표로 구분된 열 제목 목록을 입력합니다.

처리 옵션: 단일 열에서 파일 허용

오직 하나의 열만 포함하는 파일을 처리할 수 있도록 합니다.

처리 옵션: 열 값을 식별하기 전에 공백 트리밍

열 값의 유형을 식별하기 전에 값의 트리밍 여부를 지정합니다.

사용자 지정 데이터 유형 - 선택 사항

쉼표로 분리된 사용자 지정 데이터 유형을 입력합니다. CSV 파일의 사용자 지정 데이터 유형을 지정합니다. 사용자 지정 데이터 유형은 지원되는 데이터 유형이어야 합니다. 지원되는 데이터 유형은 “바이너리”, “부울”, “날짜”, “십진수”, “더블”, “플로트”, “정수”, “롱”, “쇼트”, “문자열”, “타임스탬프”입니다. 지원되지 않는 데이터 유형은 오류를 표시합니다.

AWS Glue 콘솔을 사용하여 분류자 생성

분류자는 데이터의 스키마를 결정합니다. 사용자 지정 분류자를 작성하고 AWS Glue에서 분류자를 가리킬 수 있습니다.

분류자 생성

AWS Glue 콘솔에서 분류자를 추가하려면 [분류자 추가(Add classifier)]를 선택합니다. 분류자를 정의할 때 다음에 대한 값을 제공합니다.

- [분류자 이름(Classifier name)] – 분류자에 대한 고유 이름을 입력합니다.
- [분류자 유형(Classifier type)] – 분류자가 유추한 테이블의 분류자 유형입니다.
- [마지막 업데이트(Last updated) – 이 분류자가 업데이트된 마지막 시간입니다.

분류자 이름

분류자 고유 이름을 입력합니다.

분류자 유형

생성할 분류자의 유형을 선택합니다.

선택한 분류기 유형에 따라 분류기에 대해 다음 속성을 구성합니다.

Grok

- 분류

  분류된 데이터의 포맷 또는 유형을 설명하거나 사용자 정의 레이블을 제공합니다.

- Grok 패턴

  데이터를 구조화 스키마로 구분 분석하는 데 사용됩니다. Grok 패션은 데이터 스토어의 포맷을 보여주는 이름이 붙여진 패턴으로 구성됩니다. AWS Glue에서 제공하는 명명된 기본 제공 패턴과 [사용자 정의 패턴(Custom patterns)] 필드에 작성하고 포함하는 사용자 정의 패턴을 사용하여 이 grok 패턴을 작성합니다. Grok 디버거 결과가 AWS Glue 결과와 정확하게 일치하지 않더라도 grok 디버거를 통한 몇 가지 샘플 데이터를 사용하여 패턴을 시도해 볼 수 있습니다. Grok 디버거는 웹에서 볼 수 있습니다. AWS Glue가 제공한 이름이 붙여진 기본 설정 패턴은 웹에서 사용 가능한 grok 패턴과 일반적으로 호환됩니다.

  반복적으로 이름이 붙여진 패턴을 추가하여 Grok 패턴을 만들고 디버거에서 결과를 확인합니다. 이 활동은 AWS Glue 크롤러가 grok 패턴을 실행할 때 신뢰를 줘 데이터가 구문 분석될 수 있습니다.

- 사용자 지정 패턴

  Grok 분류자의 경우, 이것은 여러분이 작성하는 Grok pattern(Grok 패턴)의 조건부 빌딩 블록입니다. 기본 설정 패턴이 데이터를 구분 분석하지 못할 경우, 사용자 지정 패턴을 작성해야 할 수도 있습니다. 이런 사용자 지정 패턴은 이 필드에서 정의되고 Grok pattern(Grok 패턴) 필드에서 참조됩니다. 각 사용자 지정 패턴은 개별 라인에서 정의됩니다. 기본 설정 패턴과 같이 regular expression(정규식) (regex) 구문을 사용하는 이름이 붙여진 패턴 정의로 구성됩니다.

  예를 들어, 다음은 뒤이어 정규식 정의가 있는 `MESSAGEPREFIX` 이름을 붙여 데이터를 적용하고 데이터가 패턴을 따르는지 결정합니다.

  ```
  MESSAGEPREFIX .*-.*-.*-.*-.*
  ```

XML

- 행 태그

  XML 분류자를 위해, 이것은 XML 문서의 테이블 행을 정의하는 XML 태그 이름입니다. < > 꺽쇠 괄호없이 이름을 입력합니다. 이름은 태그에 대한 XML 규칙을 따라야 합니다.

  자세한 내용은 XML 사용자 지정 분류자 작성 단원을 참조하십시오.

JSON

- JSON 경로

  JSON 분류자를 위해, 이것은 생성된 테이블 행을 정의하는 객체, 배열 및 값까지의 JSON 경로입니다. AWS Glue가 지원한 연산자를 사용하여 점 혹은 괄호 JSON 구문 이름을 입력합니다.

  자세한 내용은 JSON 사용자 지정 분류자 작성의 연산자 목록을 참조하십시오.

CSV

- 열 구분 기호

  행의 열 입력 항목 각각을 구분하는 것을 나타내기 위한 단일 문자 또는 기호입니다. 목록에서 구분 기호를 선택하거나 `Other`를 선택하여 사용자 정의 구분 기호를 입력합니다.

- 인용 기호

  단일 열 값에 내용을 결합하는 것을 나타내기 위한 단일 문자 또는 기호입니다. 열 구분 기호와 달라야 합니다. 목록에서 따옴표 기호를 선택하거나 `Other`를 선택하여 사용자 정의 따옴표 문자를 입력합니다.

- 열 제목

  열 제목을 CSV 파일에서 어떻게 탐지해야 하는지에 대한 행동을 표시합니다. `Has headings`, `No headings` 또는 `Detect headings`를 선택할 수 있습니다. 사용자 지정 CSV 파일에 열 제목이 포함되어 있는 경우에는 쉼표로 구분된 열 제목 목록을 입력합니다.

- 단일 열이 있는 파일 허용

CSV로 분류되려면 데이터는 적어도 데이터의 두 개의 열과 두 개의 행이 있어야 합니다. 하나의 열만 포함하는 파일의 처리를 허용하려면 이 옵션을 사용합니다.

- 열 값을 식별하기 전에 공백 트리밍

  이 옵션은 열 값의 유형을 식별하기 전에 값의 트리밍 여부를 지정합니다.

- 사용자 지정 데이터 유형

  (선택 사항) - 쉼표로 구분된 목록에 사용자 지정 데이터 유형을 입력합니다. 지원되는 데이터 유형은 “바이너리”, “부울”, “날짜”, “십진수”, “더블”, “플로트”, “정수”, “롱”, “쇼트”, “문자열”, “타임스탬프”입니다.

- CSV Serde

  (선택 사항) - 분류자에서 CSV를 처리하기 위한 Serde입니다. 이는 데이터 카탈로그에서 적용됩니다. `Open CSV SerDe`, `Lazy Simple SerDe` 또는 `None` 중에서 선택합니다. 크롤러에서 감지하려는 경우 `None` 값을 지정할 수 있습니다.

자세한 내용은 다양한 데이터 형식에 대한 사용자 지정 분류자 작성 단원을 참조하십시오.

분류자 보기

AWS Glue 콘솔을 https://console.aws.amazon.com/glue/에서 열고 [분류자(Classifiers)] 탭을 선택하여 생성한 모든 분류자에 대한 목록을 봅니다.

분류자 목록은 각 분류자에 대한 다음 속성을 표시합니다.

- 분류자 - 분류자 이름입니다. 분류자를 생성할 때는 분류자 이름을 제공해야 합니다.
- [분류(Classification)] – 분류자가 유추한 테이블의 분류자 유형입니다.
- [마지막 업데이트(Last updated) – 이 분류자가 업데이트된 마지막 시간입니다.

분류자 관리

AWS Glue 콘솔의 [분류자(Classifiers)] 목록에서 분류자를 추가, 편집 및 삭제할 수 있습니다. 목록에서 분류자 이름을 선택하여 분류자에 대한 더 자세한 정보를 알아봅니다. 세부 정보는 분류자를 생성할 때 정의한 정보를 포함합니다.

크롤러 구성

크롤러가 데이터 스토어로 액세스하고 메타데이터를 식별하며 AWS Glue Data Catalog의 테이블 정의를 생성합니다. AWS Glue 콘솔의 [크롤러(Crawlers)] 창에는 생성한 모든 크롤러가 나열됩니다. 목록은 크롤러 마지막 실행의 상태와 지표를 보여줍니다.

이 항목에는 크롤러의 파라미터 설정, 크롤링할 데이터 소스 정의, 보안 설정, 크롤링된 데이터 관리와 같은 필수 요소를 다루는 크롤러 구성의 단계별 프로세스가 포함되어 있습니다.

주제

- 1단계: 크롤러 속성 설정
- 2단계: 데이터 소스 및 분류자 선택
- 3단계: 보안 설정 구성
- 4단계: 출력 및 일정 설정
- 5단계: 검토 및 생성

1단계: 크롤러 속성 설정

크롤러를 구성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다. 탐색 창에서 크롤러를 선택합니다.
2. 크롤러 생성을 선택하고 크롤러 추가 마법사의 지침을 따릅니다. 마법사는 크롤러 생성에 필요한 단계를 안내합니다. 사용자 지정 분류기를 추가하여 스키마를 정의하려면 분류자 정의 및 관리 섹션을 참조하세요.
3. 크롤러에 대한 이름과 설명(선택 사항)을 입력합니다. 선택에 따라 태그 키 와 선택 사항인 태그 값으로 크롤러에 태그를 지정할 수 있습니다. 생성된 태그 키는 읽기 전용입니다. 일부 리소스에서 태그를 이용하면 리소스를 정리하고 식별하는 데 도움이 됩니다. 자세한 내용은 AWS Glue의 AWS 태그를 참조하세요.

   명칭

   이름은 문자(A-Z), 숫자(0-9), 하이픈(-) 또는 밑줄(_) 포함할 수 있으며 최대 255자로 지정할 수 있습니다.

   설명

   설명은 최대 2048자까지 입력이 가능합니다.

Tags

태그를 사용하여 리소스를 정리하고 식별할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- AWS Glue의 AWS 태그

2단계: 데이터 소스 및 분류자 선택

다음으로 크롤러의 데이터 소스 및 분류자를 구성합니다.

지원되는 데이터 소스에 대한 자세한 내용은 크롤링에 지원되는 데이터 소스 섹션을 참조하세요.

데이터 소스 구성

데이터가 이미 매핑되어 있는지에 대한 적절한 옵션을 선택합니다.AWS Glue 테이블? '아직 없음' 또는 '예'를 선택합니다. 기본적으로 '아직'이 선택됩니다.

크롤러는 크롤의 원본으로 데이터 스토어에 직접 액세스하거나 데이터 카탈로그의 기존 테이블을 원본으로 사용할 수 있습니다. 크롤러가 기존 카탈로그 테이블을 사용하면 해당 카탈로그 테이블에 지정된 데이터 스토어가 크롤됩니다.

- 아직 안 됨: 크롤링할 데이터 소스를 하나 이상 선택합니다. 크롤러는 다양한 유형(Amazon S3, JDBC등)의 여러 데이터 스토어를 크롤링할 수 있습니다.

  한 번에 하나의 데이터 스토어만 구성할 수 있습니다. 연결 정보를 제공하고 경로를 포함하고 패턴을 제외하면 다른 데이터 스토어를 추가할 수 있습니다.

- 예: 에서 기존 테이블 선택 AWS Glue 데이터 카탈로그. 카탈로그 테이블은 크롤할 데이터 스토어를 지정합니다. 크롤러는 단일 실행으로 카탈로그 테이블만 크롤할 수 있습니다. 다른 원본 유형에서는 혼합할 수 없습니다.

  카탈로그 테이블을 원본으로 지정하는 일반적인 이유는 (데이터 스토어의 구조를 이미 알고 있었으므로) 테이블을 수동으로 생성했으며 크롤러가 새 파티션을 추가하는 등 테이블을 업데이트된 상태로 유지하기 때문입니다. 다른 이유에 대한 설명은 크롤러를 사용하여 수동으로 생성된 Data Catalog 테이블 업데이트 단원을 참조하십시오.

  기존 테이블을 크롤러 원본 유형으로 지정하면 다음 조건이 적용됩니다.

  - 데이터베이스 이름은 선택 사항입니다.
  - Amazon S3 또는 Amazon DynamoDB 또는 Delta Lake 데이터 스토어를 지정하는 카탈로그 테이블만 허용됩니다.

- 크롤러가 실행되면 새 카탈로그 테이블이 생성되지 않습니다. 기존 테이블은 새 파티션을 추가하는 등 필요에 따라 업데이트됩니다.
- 데이터 스토어에서 찾은 삭제된 객체는 무시됩니다. 카탈로그 테이블은 삭제되지 않습니다. 대신 크롤러는 로그 메시지를 작성합니다. (`SchemaChangePolicy.DeleteBehavior=LOG`)
- 각 Amazon S3 경로에 대해 단일 스키마를 생성하는 크롤러 구성 옵션은 기본적으로 사용되며 사용 중지할 수 없습니다. (`TableGroupingPolicy=CombineCompatibleSchemas`) 자세한 내용은 각 Amazon S3 포함 경로에 대해 단일 스키마 생성 섹션을 참조하십시오.
- 다른 원본 유형(예: Amazon S3 또는 Amazon DynamoDB)과 원본으로 카탈로그 테이블을 혼합할 수 없습니다.

Delta 테이블을 사용하려면 먼저 Athena DDL 또는 를 사용하여 Delta 테이블을 생성합니다 AWS Glue API.

Athena를 사용하여 위치를 Amazon S3 폴더로 설정하고 테이블 유형을 'DELTA'로 설정합니다.

```
CREATE EXTERNAL TABLE database_name.table_name
LOCATION 's3://bucket/folder/'
TBLPROPERTIES ('table_type' = 'DELTA')
```

를 사용하여 테이블 파라미터 맵 내에서 테이블 유형을 AWS Glue API지정합니다. 테이블 매개변수에는 다음과 같은 키/값 쌍이 포함되어야 합니다. 테이블을 생성하는 방법에 대한 자세한 내용은 create_table에 대한 Boto3 설명서를 참조하세요.

```
{
    "table_type":"delta"
}
```

데이터 소스

크롤러가 스캔할 데이터 소스 목록을 선택하거나 추가합니다.

(선택 사항) 데이터 소스JDBC로 를 선택하는 경우 JDBC 드라이버 정보가 저장되는 연결 액세스를 지정할 때 자체 드라이버를 사용할 수 있습니다.

포함 경로

크롤러에 무엇을 포함하거나 제외할 것인지 평가할 때 크롤러는 필요한 포함 경로를 평가하는 것으로 시작합니다. Amazon S3, MongoDB, MongoDB Atlas, Amazon DocumentDB(MongoDB 호환) 및 관계형 데이터 스토어의 경우 포함 경로를 지정해야 합니다.

Amazon S3 데이터 스토어의 경우

이 계정에서 경로를 지정할지 혹은 다른 계정에서 경로를 지정할지 선택한 다음 Amazon S3 경로 선택을 위해 찾아봅니다.

Amazon S3 데이터 스토어의 경우 경로 포함 구문은 `bucket-name/folder-name/file-name.ext`입니다. 버킷에 있는 모든 객체를 크롤하려면 추가 경로에서 버킷 이름을 지정합니다. 제외 패턴은 포함 경로와 상대적입니다.

Delta Lake 데이터 스토어의 경우

델타 테이블에 대한 하나 이상의 Amazon S3 경로를 s3://로 지정*bucket*/*prefix*/*object*.

Iceberg 또는 Hudi 데이터 스토어의 경우

Iceberg 또는 Hudi 테이블 메타데이터가 있는 폴더가 포함된 하나 이상의 Amazon S3 경로를 s3://로 지정합니다.*bucket*/*prefix*.

Iceberg 및 Hudi 데이터 스토어의 경우 Iceberg/Hudi 폴더는 루트 폴더의 하위 폴더에 있을 수 있습니다. 크롤러는 경로 아래에 있는 모든 폴더에서 Hudi 폴더를 스캔합니다.

JDBC 데이터 스토어의 경우

Enter *<database>*/*<schema>*/*<table>* 또는 *<database>*/*<table>*데이터베이스 제품에 따라 . Oracle Database와 MySQL는 경로에서 스키마를 지원하지 않습니다. 백분율(%) 문자를 로 대체할 수 있습니다.*<schema>* 또는 *<table>*. 예를 들어 시스템 식별자(SID)가 인 Oracle 데이터베이스의 경우 `orclorcl/%`를 입력하여 연결에서 명명된 사용자가 액세스할 수 있는 모든 테이블을 가져옵니다.

> **Important**
>
> 이 필드는 대/소문자를 구분합니다.

> **Note**
>
> 자체 JDBC 드라이버 버전을 가져오도록 선택한 경우 AWS Glue 크롤러는 에서 리소스를 사용합니다.AWS Glue 제공된 드라이버가 환경에서 실행되도록 하기 위한 작업 및 Amazon S3 버킷. 리소스의 추가 사용량은 계정에 반영됩니다. 드라이버는 추가에 설명된 속성으로 제한됩니다.AWS Glue 연결 .

MongoDB, MongoDB Atlas 또는 Amazon DocumentDB 데이터 스토어의 경우

MongoDB, MongoDB Atlas 및 Amazon DocumentDB(MongoDB와 호환)의 경우 구문은 `database/collection`입니다.

JDBC 데이터 스토어의 경우 구문은 `database-name/schema-name/table-name` 또는 입니다`database-name/table-name`. 구문은 데이터베이스 엔진이 데이터베이스 내에서 스키마를 지원할지 여부에 따라 다릅니다. 예를 들어 MySQL 또는 Oracle과 같은 데이터베이스 엔진의 경우 포함 경로`schema-name`에 를 지정하지 마세요. 퍼센트 표시(%)를 추가 경로 내 스키마 또는 테이블로 바꿔 데이터베이스 내 모든 스키마 혹은 모든 테이블을 나타냅니다. 퍼센트 표시(%)를 추가 경로 내 데이터베이스로 바꿀 수 없습니다.

최대 이동 깊이(Iceberg 또는 Hudi 데이터 스토어만 해당)

크롤러가 Amazon S3 경로에서 Iceberg 또는 Hudi 메타데이터 폴더를 검색하기 위해 이동할 수 있는 Amazon S3 경로의 최대 깊이를 정의합니다. 이 파라미터의 목적은 크롤러 실행 시간을 제한하는 것입니다. 기본값은 10이고, 최댓값은 20입니다.

제외 패턴

이렇게 하면 크롤에서 특정 파일이나 테이블을 제외할 수 있습니다. 제외 경로는 포함 경로와 상대적입니다. 예를 들어 JDBC 데이터 스토어에서 테이블을 제외하려면 제외 경로에 테이블 이름을 입력합니다.

크롤러는 를 사용하여 JDBC 데이터 스토어에 연결합니다.AWS Glue 연결 문자열이 포함된 JDBC URI 연결입니다. 크롤러는 의 JDBC 사용자 이름과 암호를 사용하여 데이터베이스 엔진의 객체에만 액세스할 수 있습니다.AWS Glue 연결. 크롤러는 JDBC 연결을 통해 액세스할 수 있는 테이블만 생성할 수 있습니다. 크롤러가 JDBC 를 사용하여 데이터베이스 엔진에 액세스하면 URI포함 경로가 Data Catalog에 데이터베이스 엔진의 어떤 테이블이 생성되는지 결정하는 데 사용됩니다. 예를 들어 내 에서 의 포함 경로를 지정SQL하면 내의 `MyDatabase/%`모든 테이블`MyDatabase`이 데이터 카탈로그에 생성됩니다. Amazon Redshift에 액세스할 경우, `MyDatabase/%`의 추가 경로를 지정하면 데이터베이스 `MyDatabase`의 모든 스키마 내 모든 테이블은 데이터 카탈로그에 생성됩니다. `MyDatabase/MySchema/%`의 추가 경로를 지정하면 `MyDatabase` 데이터베이스의 모든 테이블과 `MySchema` 스키마가 생성됩니다.

추가 경로를 지정한 후, 크롤러에서 객체를 제외합니다. 그렇지 않으면 추가 경로는 하나 이상의 Unix 스타일 `glob`이 패턴을 제외하는 것을 지정하여 객체를 추가합니다. 이런 패턴은 추가 경로에 적용되어 어떤 객체가 제외되어야 하는지 결정합니다. 또한 이런 패턴은 크롤러가 생성한 테이블의 속성으로 저장됩니다.AWS Glue PySpark 와 같은 확장은 테이블 속성을 `create_dynamic_frame.from_catalog`읽고 제외 패턴으로 정의된 객체를 제외합니다.

AWS Glue 는 제외 glob 패턴에서 다음 패턴을 지원합니다.

제외 패턴	설명
*.csv	.csv로 끝나는 현재 폴더의 객체 이름을 나타내는 Amazon S3 경로에 해당합니다.
.	점을 포함한 모든 객체 이름과 해당합니다.
*.{csv,avro}	.csv 또는 .avro로 끝나는 객체 이름과 해당합니다.
foo.?	하나의 문자 확장에 따라 foo.으로 시작하는 객체 이름과 해당합니다.
myfolder/*	/myfolder/mysource 와 같은 myfolder에서 하위 폴더의 한 수준의 객체와 해당합니다.
myfolder/*/*	/myfolder/mysource/data 와 같은 myfolder에서 하위 폴더의 두 가지 수준의 객체와 해당합니다.
myfolder/**	/myfolder/mysource/mydata 및 / myfolder/mysource/data 와 같은 myfolder의 모든 하위 폴더의 객체와 해당합니다.
myfolder**	myfolder 아래 파일뿐만 아니라 하위 폴더 myfolder도 일치시킵니다(예: /myfolder 와 /myfolder/mydata.txt).
Market*	JDBC 데이터베이스의 테이블을 Market_us 및 와 Market같이 로 시작하는 이름과 일치시킵니다. Market_fr

AWS Glue 는 다음과 같이 glob 제외 패턴을 해석합니다.

- Amazon S3 키를 폴더 계층으로 구분하기 위해 슬래시(/) 문자가 사용됩니다.

- 별표(*) 문자는 폴더의 경계를 넘지 않고 0개 이상의 이름 구성 요소 문자와 해당합니다.
- 이중 별표(**) 문자는 폴더 또는 스키마의 경계를 넘지 않고 0개 이상의 이름 구성 요소 문자와 해당합니다.
- 물음표(?) 문자는 이름 구성 요소의 정확히 1문자에 해당합니다.
- 역 슬래시(\) 문자는 특수 문자로 해석될 수 있는 이스케이프 문자로 사용됩니다. 표현식 \\는 하나의 백 슬래시에 해당하고 \{는 왼쪽 중괄호에 해당합니다.
- 대괄호([ ])는 문자 모음 중 한 문자의 이름 구성 요소에 해당하는 괄호 표현식을 생성합니다. 예를 들어, [abc]는 a, b 혹은 c에 해당합니다. 하이픈(-)은 범위를 지정하는 데 사용되기 때문에 [a-z]는 a에서 z까지의 범위를 지정합니다 (포괄적). 이런 형식은 혼합될 수 있어 [abce-g]는 a, b, c, e, f, 또는 g에 해당됩니다. 대괄호([) 다음 문자가 느낌표(!)라면, 괄호 표현식이 취소됩니다. 예를 들어, [!a-c]는 a, b 또는 c를 제외한 모든 문자와 대응합니다.

  괄호 표현식 내에 *, ? 및 \는 자체로 대응됩니다. 취소할 때 하이픈이 괄호 내 첫 번째 문자이거나 ! 다음 첫 번째 문자이면 하이픈(-) 문자는 자체로 대응합니다.
- 부 패턴이 그룹 내에서 대등하면 중괄호({ })는 그룹과 대응하는 부 패턴 그룹을 포함합니다. 쉼표(,) 문자는 부 패턴을 구분하는 기호입니다. 그룹은 중첩될 수 없습니다.
- 파일명의 첫 번째 마침표 혹은 점은 매칭 작업에서 일반 문자로 취급합니다. 예를 들어 *가 파일명 .hidden에 해당하는 패턴을 제외합니다.

Example Amazon S3 제외 패턴

각 제외 패턴은 추가 경로에 대응하여 평가됩니다. 예를 들어 다음과 같은 Amazon S3 디렉터리 구조가 있다고 가정하겠습니다.

```
/mybucket/myfolder/
   departments/
      finance.json
      market-us.json
      market-emea.json
      market-ap.json
   employees/
      hr.json
      john.csv
      jane.csv
      juan.txt
```

추가 경로 s3://mybucket/myfolder/에 따르면 다음 예는 패턴을 제외한 결과입니다.

제외 패턴	결과
`departments/**`	`departments` 아래에 있는 모든 파일 및 폴더를 제외하고 `employees` 폴더 및 하위 폴더를 포함합니다.
`departments/market*`	`market-us.json` , `market-emea.json` 및 `market-ap.json` 을 제외합니다.
`**.csv`	`.csv`로 끝나는 이름을 갖는 `myfolder`의 모든 하위 객체를 제외합니다.
`employees/*.csv`	`employees` 폴더 내 모든 `.csv` 파일을 제외합니다

Example Amazon S3 파티션의 하위 집합 제외

일별로 데이터가 분할되었다면 개별 Amazon S3 파티션에 1년의 각 일자가 존재합니다. 따라서 2015년 1월이면 31 파티션이 있습니다. 이제, 1월 첫째 주 데이터만 크롤하고자 한다면 1일부터 7일까지만 남겨놓고 모든 파티션을 제외합니다.

```
2015/01/{[!0],0[8-9]}**, 2015/0[2-9]/**, 2015/1[0-2]/**
```

이 glob 형식을 살펴보겠습니다. 첫 번째 부분인 `2015/01/{[!0],0[8-9]}**`는 2015년 01월 중 08일과 09뿐만 아니라 0으로 시작하지 않는 모든 일자를 제외합니다. "**"는 일수 형식의 접미사로 사용된다는 것을 기억하고 폴더 경계를 더 낮은 수준의 폴더로 지나갑니다. "*"가 사용되면 하위 폴더 수준은 제외되지 않습니다.

두 번째 부분인 `2015/0[2-9]/**`는 2015년 중 02월에서 09월까지의 일자를 제외합니다.

세 번째 부분인 `2015/1[0-2]/**`는 2015년 중 10월, 11월 및 12월의 일자를 제외합니다.

Example JDBC 패턴 제외

다음 스키마 구조를 사용하여 JDBC 데이터베이스를 크롤링한다고 가정해 보겠습니다.

```
MyDatabase/MySchema/
```

```
HR_us
HR_fr
Employees_Table
Finance
Market_US_Table
Market_EMEA_Table
Market_AP_Table
```

추가 경로 `MyDatabase/MySchema/%`에 따르면 다음 예는 패턴을 제외한 결과입니다.

제외 패턴	결과
`HR*`	HR로 이름이 시작되는 테이블을 제외합니다.
`Market_*`	Market_로 이름이 시작되는 테이블을 제외합니다.
`**_Table`	_Table로 이름이 끝나는 모든 테이블을 제외합니다.

추가 크롤러 소스 파라미터

소스 유형마다 다른 추가 파라미터 집합이 필요합니다.

연결

선택 또는 추가 AWS Glue 연결. 연결에 관한 자세한 내용은 데이터에 연결 단원을 참조하십시오.

추가 메타데이터 - 선택 사항(JDBC데이터 스토어의 경우)

크롤러가 크롤링할 추가 메타데이터 속성을 선택합니다.

- 댓글: 관련 테이블 수준 및 열 수준 설명을 크롤링합니다.
- 원시 유형: 테이블 열의 원시 데이터 유형을 추가 메타데이터에 유지합니다. 기본 동작으로 크롤러는 원시 데이터 유형을 Hive 호환 유형으로 변환합니다.

JDBC 드라이버 클래스 이름 - 선택 사항(JDBC데이터 스토어의 경우)

데이터 소스에 연결할 크롤러의 사용자 지정 JDBC 드라이버 클래스 이름을 입력합니다.

- Postgres: org.postgresql.Driver
- 내 SQL: com.mysql.jdbc.Driver, com.mysql.cj.jdbc.Driver

- Redshift: com.amazon.redshift.jdbc.Driver, com.amazon.redshift.jdbc42.Driver
- Oracle: oracle.jdbc.driver.OracleDriver
- SQL 서버: com.microsoft.sqlserver.jdbc.SQLServerDriver

JDBC 드라이버 S3 경로 - 선택 사항(JDBC데이터 스토어용)

`.jar` 파일에 대한 기존 Amazon S3 경로를 선택합니다. 여기에서 크롤러가 데이터 소스에 연결하기 위해 사용자 지정 JDBC 드라이버를 사용할 때 `.jar` 파일이 저장됩니다.

데이터 샘플링 사용(Amazon DynamoDB, MongoDB, MongoDB Atlas Amazon DocumentDB 데이터 스토어만 해당)

데이터 샘플만 크롤링할지 여부를 선택합니다. 선택하지 않으면 전체 테이블이 크롤링됩니다. 테이블이 높은 처리량 테이블이 아닌 경우 모든 레코드를 스캔하는 데 시간이 오래 걸릴 수 있습니다.

쿼리를 위한 테이블 생성(Delta Lake 데이터 스토어에만 해당)

Delta Lake 테이블 생성 방법을 선택합니다.

- 기본 테이블 생성: Delta 트랜잭션 로그의 쿼리를 직접 지원하는 쿼리 엔진과 통합할 수 있습니다.
- Symlink 테이블 생성: 지정된 구성 파라미터를 기반으로 파티션 키로 분할된 매니페스트 파일을 포함하는 symlink 매니페스트 폴더를 생성합니다.

스캔 속도 - 선택 사항(DynamoDB 데이터 스토어에만 해당)

크롤러에서 사용할 DynamoDB 테이블 읽기 용량 단위의 비율을 지정합니다. 읽기 용량 단위는 DynamoDB에서 정의한 용어이며, 초당 해당 테이블에서 수행할 수 있는 읽기 수에 대한 속도 제한기 역할을 하는 숫자 값입니다. 0.1과 1.5 사이의 값을 입력합니다. 값을 지정하지 않으면 기본값은 프로비저닝된 테이블의 경우 0.5이고 온디맨드 테이블의 경우 구성된 최대 용량의 1/4입니다. 프로비저닝된 용량 모드만 AWS Glue 크롤러와 함께 사용해야 합니다.

> **Note**
>
> DynamoDB 데이터 스토어의 경우, 테이블의 읽기 및 쓰기 처리를 위한 프로비저닝된 용량 모드를 설정합니다. AWS Glue 크롤러를 온디맨드 용량 모드와 함께 사용해서는 안 됩니다.

네트워크 연결 - 선택 사항(Amazon S3, Delta, Iceberg, Hudi 및 Catalog 대상 데이터 스토어의 경우)

필요에 따라 이 Amazon S3 대상과 함께 사용할 네트워크 연결을 포함합니다. 각 크롤러는 하나의 네트워크 연결로 제한되므로 다른 Amazon S3 대상도 동일한 연결을 사용하게 됩니다(비워 두면 연결 없음).

연결에 관한 자세한 내용은 데이터에 연결 단원을 참조하십시오.

파일의 하위 집합과 샘플 크기만 샘플링(Amazon S3 데이터 스토어에만 해당)

데이터 집합의 샘플 파일을 크롤링할 때 크롤링할 각 리프 폴더의 파일 수를 지정합니다. 이 기능이 설정되어 있으면 이 데이터 집합의 모든 파일을 크롤링하는 대신 크롤러가 크롤링할 각 리프 폴더의 일부 파일을 무작위로 선택합니다.

샘플링 크롤러는 데이터 포맷에 대한 사전 지식이 있고 폴더의 스키마가 변경되지 않는다는 것을 알고 있는 고객에게 가장 적합합니다. 이 기능을 설정하면 크롤러 런타임이 크게 감소합니다.

유효한 값은 1~249의 정수입니다. 지정하지 않으면 모든 파일이 크롤링됩니다.

후속 크롤러 실행

이 필드는 모든 Amazon S3 데이터 소스에 영향을 주는 글로벌 필드입니다.

- 모든 하위 폴더 크롤링: 이후에 크롤링할 때마다 모든 폴더를 다시 크롤링합니다.
- 새 하위 폴더만 크롤링: 마지막 크롤링된 이후에 추가된 Amazon S3 폴더만 크롤링됩니다. 스키마가 호환되는 경우 새 파티션이 기존 테이블에 추가됩니다. 자세한 내용은 the section called “증분 크롤링 예약” 단원을 참조하십시오.
- 이벤트 기반 크롤링: Amazon S3 이벤트를 사용하여 크롤링할 폴더를 제어합니다. 자세한 내용은 Amazon S3 이벤트 알림을 사용하여 크롤링 가속화 단원을 참조하십시오.

사용자 지정 분류자 - 선택 사항

크롤러를 정의하기 전에 사용자 지정 분류자를 정의합니다. 분류자는 지정된 파일이 크롤러가 처리할 수 있는 형식인지 여부를 확인합니다. 처리할 수 있는 형식인 경우, 분류자는 해당 데이터 형식과 일치하는 `StructType` 객체의 형태로 스키마를 생성합니다.

자세한 내용은 분류자 정의 및 관리 단원을 참조하십시오.

3단계: 보안 설정 구성

IAM 역할

크롤러는 이 역할을 수임합니다. AWS 관리형 정책 `AWSGlueServiceRole`과 비슷한 권한이 있어야 합니다. Amazon S3 및 DynamoDB 소스의 경우 데이터 스토어에 액세스할 수 있는 권한도 있어야 합니다. 크롤러가 AWS Key Management Service(AWS KMS)로 암호화된 Amazon S3 데이터를 읽는 경우 역할에 AWS KMS 키에 대한 복호화 권한이 있어야 합니다.

Amazon S3 데이터 스토어의 경우 역할에 연결된 추가 권한은 다음과 유사합니다.

```
{
   "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
              "s3:GetObject",
              "s3:PutObject"
          ],
          "Resource": [
              "arn:aws:s3:::bucket/object*"
          ]
        }
    ]
}
```

Amazon DynamoDB 데이터 스토어의 경우 역할에 연결된 추가 권한은 다음과 유사합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:Scan"
      ],
      "Resource": [
        "arn:aws:dynamodb:region:account-id:table/table-name*"
      ]
    }
  ]
}
```

자체 JDBC 드라이버를 추가하려면 추가 권한을 추가해야 합니다.

- `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun` 작업에 대한 권한을 부여합니다.
- Amazon S3 작업, `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`에 대한 권한을 부여합니다.

> **Note**
>
> Amazon S3 버킷 정책이 비활성화된 경우에는 `s3:ListBucket`이 필요하지 않습니다.

- 서비스 보안 주체에 Amazon S3 정책의 버킷 및 폴더에 대한 액세스 권한을 부여합니다.

Amazon S3 정책 예제:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/driver-parent-folder/driver.jar",
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

AWS Glue에서는 Amazon S3 버킷에서 JDBC 드라이버와 동일한 수준에 있는 `_crawler` 및 `_glue_job_crawler` 폴더를 생성합니다. 예를 들어 드라이버 경로가 `<s3-path/driver_folder/driver.jar>`인 경우 다음 폴더가 아직 없으면 폴더가 생성됩니다.

- <s3-path/driver_folder/_crawler>
- <s3-path/driver_folder/_glue_job_crawler>

선택적으로 크롤러에 보안 구성을 추가하여 저장 데이터 암호화 옵션을 지정할 수 있습니다.

자세한 내용은 2단계: AWS Glue에 대한 IAM 역할 생성 및 AWS Glue의 Identity and Access Management 단원을 참조하세요.

Lake Formation 구성 - 선택 사항

크롤러가 Lake Formation 보안 인증을 사용하여 데이터 소스를 크롤링하도록 허용합니다.

Use Lake Formation credentials for crawling S3 data source(Lake Formation 보안 인증을 사용하여 S3 데이터 소스 크롤링)을 선택하면 크롤러가 Lake Formation 보안 인증을 사용하여 데이터 소스를 크롤링할 수 있습니다. 데이터 소스가 다른 계정에 속하는 경우 등록된 계정 ID를 제공해야 합니다. 그렇지 않으면 크롤러는 계정과 연결된 데이터 소스만 크롤링합니다. Amazon S3 및 데이터 카탈로그 데이터 소스에만 해당됩니다.

보안 구성 - 선택 사항

설정에는 보안 구성이 포함됩니다. 자세한 내용은 다음 자료를 참조하세요.

- AWS Glue에서 작성한 데이터 암호화

> **Note**
>
> 크롤러에 보안 구성이 설정된 후에는 이를 변경할 수는 있지만 제거할 수는 없습니다. 크롤러의 보안 수준을 낮추려면 구성 내에서 보안 기능을 `DISABLED`로 명시적으로 설정하거나 새 크롤러를 생성하세요.

4단계: 출력 및 일정 설정

출력 구성

옵션에는 크롤러가 감지된 스키마 변경 사항, 데이터 스토어에서 삭제된 객체 등을 처리하는 방법이 포함됩니다. 자세한 내용은 크롤러 동작 사용자 지정 단원을 참조하세요.

크롤러 예약

필요에 따라 크롤러를 실행하거나 AWS Glue에서 크롤러 및 작업을 위한 시간 기반 일정을 정의합니다. 일정 정의는 Unix식 cron 구문을 사용합니다. 자세한 내용은 크롤러 일정 관리 단원을 참조하십시오.

5단계: 검토 및 생성

구성한 크롤러 설정을 검토하고 크롤러를 생성합니다.

크롤러 일정 관리

요구에 따라 정기 일정에 따라 AWS Glue 크롤러를 실행합니다. 일정에 따라 크롤러를 설정할 경우 크롤러의 실행 빈도, 실행할 요일, 실행 시간과 같은 특정 제약 조건을 지정할 수 있습니다. 이러한 사용자 지정 일정을 cron 형식으로 생성할 수 있습니다. 자세한 내용은 Wikipedia의 cron을 참조하십시오.

크롤러 일정을 설정하고자 한다면 CRON의 기능 및 제약점을 고려해야 합니다. 예를 들어, 매월 31일에 크롤러를 실행하고자 한다면 매월 31일이 없다는 점을 유의하기 바랍니다.

주제

- 크롤러 일정 생성
- 기존 크롤러에 대한 일정 생성

크롤러 일정 생성

AWS Glue 콘솔 또는 AWS CLI를 사용하여 크롤러의 일정을 생성할 수 있습니다.

AWS Management Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 크롤러를 선택합니다.
3. 아래 크롤러 구성 섹션의 1~3 단계를 따릅니다.
4. 4단계: 출력 및 일정 설정에서 크롤러 예약을 선택하여 실행 빈도를 설정합니다. 크롤러를 시간별, 일별, 주별, 월별로 실행하도록 선택하거나 cron 표현식을 사용하여 사용자 지정 일정을 정의할 수 있습니다.

   cron 표현식은 일정 패턴을 나타내는 문자열로, * * * <minute><hour><day of month><month><day of week><year>처럼 공백으로 구분된 6개 필드로 구성됩니다.

   예를 들어 매일 자정에 작업을 실행하는 경우 cron 표현식은 0 0 * *? *입니다.

   자세한 내용은 cron 표현식 섹션을 참조하세요.
5. 구성한 크롤러 설정을 검토하고 일정에 따라 실행되는 크롤러를 생성합니다.

AWS CLI

```
aws glue create-crawler
```

```
--name myCrawler \
--role AWSGlueServiceRole-myCrawler  \
--targets '{"S3Targets":[{Path="s3://amzn-s3-demo-bucket/"}]}' \
--schedule cron(15 12 * * ? *)
```

작업 및 크롤러의 일정을 관리하기 위해 cron을 사용하는 방법에 대한 자세한 내용은 작업 및 크롤러를 위한 시간 기반 일정 섹션을 참조하세요.

기존 크롤러에 대한 일정 생성

기존 크롤러에 대한 반복 일정을 설정하려면 다음 단계를 따릅니다.

AWS Management Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 크롤러를 선택합니다.
3. 사용 가능 목록에서 예약하려는 크롤러를 선택합니다.
4. 실행 메뉴에서 편집을 선택합니다.
5. 4단계: 출력 및 예약 설정까지 아래로 스크롤한 다음 편집을 선택합니다.
6. 크롤러 예약에서 크롤러 일정을 업데이트합니다.
7. 업데이트를 선택합니다.

AWS CLI

다음 CLI 명령을 사용하여 기존 크롤러 구성을 업데이트합니다.

```
aws glue update-crawler-schedule
   --crawler-name myCrawler
   --schedule cron(15 12 * * ? *)
```

크롤러 결과 및 세부 정보 보기

크롤러가 성공적으로 실행되면 데이터 카탈로그에 테이블 정의가 생성됩니다. 탐색 창에서 [테이블(Tables)] 탭을 선택하여 지정한 데이터베이스의 크롤러가 생성한 테이블을 살펴봅니다.

다음과 같이 크롤러 자체와 관련된 정보를 볼 수 있습니다.

- AWS Glue 콘솔의 [크롤러(Crawlers)] 페이지에는 크롤러에 대한 다음 속성이 표시됩니다.

속성	설명
이름	크롤러를 생성하는 경우, 고유 이름을 붙여야 합니다.
상태	크롤러는 준비, 시작, 중지, 예약 및 예약 중지 상태일 수 있습니다. 실행 중인 크롤러는 시작에서 중지까지 진행합니다. 크롤러 예약을 재개하거나 중단할 수 있습니다.
Schedule	필요에 따라 크롤러를 실행하거나 일정 빈도수를 선택합니다. 크롤러 예약에 대한 자세한 내용은 크롤러 일정 관리를 참조하십시오.
마지막 실행(Last run)	크롤러가 마지막으로 실행된 날짜 및 시간입니다.
로그(Log)	크롤러의 마지막 실행에 대한 사용 가능한 로그에 대한 링크.
마지막 실행 이후의 테이블 변경 사항(Tables changes from last run)	크롤러의 마지막 실행에 따라 업데이트된 AWS Glue Data Catalog의 테이블 수입니다.

- 크롤러의 기록을 보려면 탐색 창에서 Crawlers(크롤러)를 선택하여 생성한 크롤러를 확인합니다. 사용 가능한 크롤러 목록에서 크롤러를 선택합니다. Crawler runs(크롤러 실행) 탭에서 크롤러 속성과 크롤러 기록을 볼 수 있습니다.

  Crawler runs(크롤러 실행) 탭에는 Start time (UTC)(시작 시간(UTC)), End time (UTC)(종료 시간(UTC)), Duration(기간), Status(상태), DPU hours(DPU 시간) 및 Table changes(테이블 변경)를 비롯하여 크롤러가 실행된 각 시간에 대한 정보가 표시됩니다.

  크롤러 실행 탭은 크롤러 기록 기능의 시작 날짜 이후 발생한 크롤만 표시하고 최대 12개월의 크롤만 유지합니다. 이전의 크롤은 반환되지 않습니다.
- 추가 정보를 보려면 크롤러 세부 정보 페이지에서 탭을 선택합니다. 각 탭에는 크롤러 관련 정보가 표시됩니다.

- Schedule(일정): 크롤러에 대해 생성된 모든 일정이 여기에 표시됩니다.
- Data sources(데이터 소스): 크롤러에서 스캔한 모든 데이터 소스가 여기에 표시됩니다.
- Classifiers(분류자): 크롤러에 할당된 모든 분류자가 여기에 표시됩니다.
- Tags(태그): 생성되고 AWS 리소스에 할당된 모든 태그가 여기에 표시됩니다.

크롤러가 데이터 카탈로그 테이블에 설정한 파라미터

이러한 테이블 속성은 AWS Glue 크롤러가 설정합니다. 사용자가`classification` 및 `compressionType` 속성을 소비할 것으로 기대합니다. 테이블 크기 추정치를 비롯한 기타 속성은 내부 계산에 사용되며, 당사는 이러한 속성의 정확성이나 고객 사용 사례에 대한 적용 가능성을 보장하지 않습니다. 이러한 파라미터를 변경하면 크롤러의 동작이 변경될 수 있지만 이 워크플로는 지원되지 않습니다.

속성 키	속성 값
`UPDATED_BY_CRAWLER`	업데이트를 수행하는 크롤러의 이름입니다.
`connectionName`	크롤러가 데이터 스토어에 연결하는 데 사용되는 데이터 카탈로그의 연결 이름입니다.
`recordCount`	파일 크기 및 헤더를 기반으로 테이블의 레코드 수를 추정합니다.
`skip.header.line.count`	헤더를 건너뛰기 위해 행을 건너뛰었습니다. CSV로 분류된 테이블에 설정합니다.
`CrawlerSchemaSerializerVersion`	내부용
`classification`	크롤러에서 추론한 데이터 형식입니다. AWS Glue에 의해 지원되는 형식에 대한 자세한 내용은 the section called “기본 제공 분류자” 단원을 참조하세요.
`CrawlerSchemaDeserializerVersion`	내부용

속성 키	속성 값
sizeKey	테이블의 파일 크기를 합산하여 크롤링했습니다.
averageRecordSize	테이블 열의 평균 크기(바이트).
compressionType	테이블의 데이터에 사용되는 압축 유형입니다. AWS Glue 크롤러가 지원하는 압축 유형에 대한 자세한 내용은 the section called “기본 제공 분류자”을 참조하세요.
typeOfData	file, table, 또는 view.
objectCount	테이블의 Amazon S3 경로 아래에 있는 객체 수입니다.

이러한 추가 테이블 속성은 Snowflake 데이터 스토어의 AWS Glue 크롤러에 의해 설정됩니다.

속성 키	속성 값
aws:RawTableLastAltered	Snowflake 테이블의 마지막으로 변경된 타임스탬프를 기록합니다.
ViewOriginalText	SQL 문을 봅니다.
ViewExpandedText	Base64 형식으로 인코딩된 SQL 문을 볼 수 있습니다.
ExternalTable:S3Location	Snowflake 외부 테이블의 Amazon S3 위치입니다.
ExternalTable:FileFormat	Snowflake 외부 테이블의 Amazon S3 파일 형식입니다.

이러한 추가 테이블 속성은 Amazon Redshift, Microsoft SQL Server, MySQL, PostgreSQL 및 Oracle과 같은 JDBC 유형 데이터 스토어의 AWS Glue 크롤러에 의해 설정됩니다.

속성 키	속성 값
`aws:RawType`	크롤러가 데이터 카탈로그에 데이터를 저장하면 데이터 유형이 Hive 호환 유형으로 변환되므로 기본 데이터 유형에 대한 정보가 손실되는 경우가 많습니다. 크롤러는 `aws:RawType` 파라미터를 출력하여 기본 수준 데이터 유형을 제공합니다.
`aws:RawColumnComment`	설명이 데이터베이스의 열과 연결되는 경우 크롤러는 카탈로그 테이블에 해당 설명을 출력합니다. 주석 문자열은 255바이트로 잘립니다. Microsoft SQL Server에서는 설명이 지원되지 않습니다.
`aws:RawTableComment`	설명이 데이터베이스의 테이블과 연결되는 경우 크롤러는 카탈로그 테이블에 해당 설명을 출력합니다. 설명 문자열은 255바이트로 잘립니다. Microsoft SQL Server에서는 설명이 지원되지 않습니다.

크롤러 동작 사용자 지정

AWS Glue 크롤러을 구성하면 크롤러의 행태를 정의할 수 있는 몇 가지 옵션이 있습니다.

- 증분 크롤링 - 테이블 스키마에 새 파티션만 추가하도록 증분 크롤링을 실행하는 크롤러를 구성할 수 있습니다.
- 파티션 인덱스 - 크롤러는 기본적으로 Amazon S3 및 Delta Lake 대상에 대한 파티션 인덱스를 생성하여 특정 파티션을 효율적으로 조회합니다.
- Amazon S3 이벤트를 사용하여 크롤링 가속 - 전체 Amazon S3 또는 데이터 카탈로그 대상을 나열하는 대신 이벤트를 트리거한 하위 폴더의 모든 파일을 나열하여 두 크롤링 간의 변경 사항을 식별하기 위해 Amazon S3 이벤트를 사용하도록 크롤러를 구성할 수 있습니다.
- 스키마 변경 처리 - 크롤러가 기존 스키마에 스키마를 변경하지 못하게 할 수 있습니다. AWS Management Console 또는 AWS Glue API를 사용하여 크롤러가 특정 변화에 따라 어떤 절차를 밟는지 알아봅니다.
- 여러 Amazon S3 경로에 대한 단일 스키마 - 데이터가 호환되는 경우 각 S3 경로에 대해 단일 스키마를 생성하도록 크롤러를 구성할 수 있습니다.
- 테이블 위치 및 파티셔닝 수준 - 테이블 수준 크롤러 옵션을 사용하면 크롤러에 테이블의 위치와 파티션 생성 방법을 유연하게 알릴 수 있습니다.

- 테이블 임계값 - 테이블 임계값을 지정하여 크롤러가 생성할 수 있는 최대 테이블 수를 지정할 수 있습니다.
- AWS Lake Formation 자격 증명 - Lake Formation 자격 증명을 사용하여 동일한 AWS 계정 또는 다른 AWS 계정 내의 기본 Amazon S3 위치가 있는 데이터 카탈로그 테이블이나 Amazon S3 데이터 스토어에 액세스하도록 크롤러를 구성할 수 있습니다.

AWS Glue 콘솔을 사용하여 크롤러를 추가하는 방법에 대한 자세한 내용은 크롤러 구성 단원을 참조하십시오.

주제

- 새 파티션을 추가하기 위한 증분 크롤링 예약
- 파티션 인덱스 생성
- 크롤러가 기존 스키마를 변경하지 않도록 방지
- 각 Amazon S3 포함 경로에 대해 단일 스키마 생성
- 테이블 위치와 파티션 수준 지정
- 크롤러가 생성할 수 있는 최대 테이블 수 지정
- Lake Formation 자격 증명을 사용하도록 크롤러 구성
- Amazon S3 이벤트 알림을 사용하여 크롤링 가속화

새 파티션을 추가하기 위한 증분 크롤링 예약

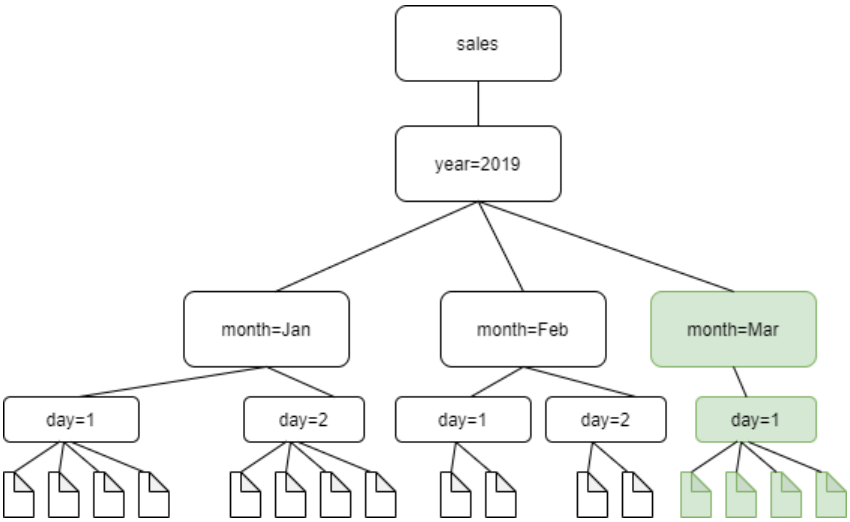
테이블 스키마에 새 파티션만 추가하도록 증분 크롤링 AWS Glue 크롤러 실행을 구성할 수 있습니다. 크롤러가 처음 실행되면 전체 데이터 소스를 처리에 전체 크롤링을 수행하여 전체 스키마와 모든 기존 파티션을 AWS Glue Data Catalog에 기록합니다.

초기 전체 크롤링 이후의 후속 크롤링은 증분 방식으로 진행되며, 이때 크롤러는 이전 크롤링 이후에 새롭게 추가된 파티션만 식별하여 추가합니다. 이 접근 방식을 사용하면 크롤러가 더 이상 각 실행에 대해 전체 데이터 소스를 처리할 필요가 없고 대신 새 파티션에만 집중하므로 크롤링 시간이 단축됩니다.

> **Note**
>
> 증분 크롤링에서는 기존 파티션의 수정 또는 삭제를 감지하지 않습니다. 이 구성은 안정적인 스키마를 사용하는 데이터 소스에 가장 적합합니다. 한 번의 주요 스키마 변경이 발생한 경우

새 스키마를 정확하게 캡처하기 위해 전체 크롤링을 수행하도록 크롤러를 일시적으로 설정한 다음 증분 크롤링 모드로 다시 전환하는 것이 좋습니다.

다음 다이어그램은 증분 크롤링 설정을 사용하도록 설정한 경우 크롤러가 새로 추가된 Month=March 폴더만 탐지하여 카탈로그에 추가한다는 것을 보여줍니다.



다음 단계에 따라 크롤러가 증분 크롤링을 수행하도록 업데이트하세요.

AWS Management Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 데이터 카탈로그에서 크롤러를 선택합니다.
3. 증분 크롤링 설정을 원하는 크롤러를 선택합니다.
4. 편집을 선택합니다.
5. 2단계를 선택합니다. 데이터 소스 및 분류자를 선택합니다.
6. 증분 크롤링 하려는 데이터 소스를 선택합니다.
7. 편집을 선택합니다.
8. 후속 크롤러 실행에서 새 하위 폴더만 크롤링을 선택합니다.
9. 업데이트를 선택합니다.

크롤러 일정을 만들려면 the section called “크롤러 일정 관리” 섹션을 참조하세요.

AWS CLI

```
aws glue update-crawler \
 --name myCrawler \
 --recrawl-policy RecrawlBehavior=CRAWL_NEW_FOLDERS_ONLY \
 --schema-change-policy UpdateBehavior=LOG,DeleteBehavior=LOG
```

규칙 및 제한

이 옵션이 설정되어 있으면 크롤러를 편집할 때 Amazon S3 대상 데이터 스토어를 변경할 수 없습니다. 이 옵션은 특정 크롤러 구성 설정에 영향을 줍니다. 설정하면 크롤러의 업데이트 동작 및 삭제 동작이 LOG에 기록됩니다. 이는 다음을 의미합니다.

- 스키마가 호환되지 않는 객체를 발견하면 크롤러는 데이터 카탈로그에 객체를 추가하지 않고 이 세부 정보를 CloudWatch Logs에 로그로 추가합니다.
- 데이터 카탈로그의 삭제된 개체는 업데이트되지 않습니다.

파티션 인덱스 생성

데이터 카탈로그는 특정 파티션을 효율적으로 조회할 수 있도록 파티션 인덱스 생성을 지원합니다. 자세한 내용은 파티션 인덱스 생성을 참조하세요. AWS Glue 크롤러는 기본적으로 Amazon S3 및 Delta Lake 대상에 대한 파티션 인덱스를 생성합니다.

AWS Management Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 데이터 카탈로그에서 크롤러를 선택합니다.
3. 크롤러를 정의하면 자동으로 파티션 인덱스 생성 옵션이 출력 및 예약 설정 페이지의 고급 옵션 아래에서 기본적으로 활성화됩니다.

   이 옵션을 비활성화하려면 콘솔에서 파티션 인덱스 자동 생성 확인란의 선택을 취소하면 됩니다.
4. 크롤러 구성을 완료하고 크롤러 생성을 선택합니다.

AWS CLI

AWS CLI를 사용하여 configuration 파라미터에서 CreatePartitionIndex 를 설정하여 이 옵션을 비활성화할 수도 있습니다. 기본값은 true입니다.

```
aws glue update-crawler \
    --name myCrawler \
    --configuration '{"Version": 1.0, "CreatePartitionIndex": false }'
```

파티션 인덱스에 대한 사용 참고 사항

- 크롤러에서 생성한 테이블에는 기본적으로 partition_filtering.enabled 변수가 없습니다. 자세한 내용은 AWS Glue 파티션 인덱싱 및 필터링을 참조하세요.
- 암호화된 파티션에 대한 파티션 인덱스 생성은 지원되지 않습니다.

크롤러가 기존 스키마를 변경하지 않도록 방지

AWS Glue 크롤러가 실행될 때 데이터 카탈로그에서 스키마를 변경하지 않도록 할 수 있습니다. 기본적으로 크롤러는 크롤링되는 데이터 소스와 일치하도록 데이터 카탈로그의 스키마를 업데이트합니다. 하지만 경우에 따라 크롤러가 기존 스키마를 수정하지 않도록 하는 것이 좋습니다. 특히 데이터를 변환하거나 정리한 후 원래 스키마가 변경 내용을 덮어쓰지 않도록 하려는 경우에는 더욱 그렇습니다.

테이블 정의의 기존 스키마를 덮어쓰지 않도록 크롤러를 구성하려면 다음 단계를 따릅니다.

AWS Management Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 데이터 카탈로그에서 크롤러를 선택합니다.
3. 목록에서 크롤러를 선택하고 편집을 선택합니다.
4. 4단계, 출력 및 예약 설정을 선택합니다.
5. 고급 옵션에서 새 열만 추가 또는 변경 내용 무시를 선택하고 데이터 카탈로그의 테이블을 업데이트하지 않음을 선택합니다.
6. 또한 구성 옵션을 테이블의 메타데이터로 모든 신규 및 기존 파티션 업데이트로 설정할 수 있습니다. 이렇게 하면 파티션 스키마가 테이블에서 상속되도록 설정됩니다.
7. 업데이트를 선택합니다.

AWS CLI

다음 예제는 기존 스키마를 변경하지 않고 새 열만 추가하도록 크롤러를 구성하는 방법을 보여줍니다.

```
aws glue update-crawler \
  --name myCrawler \
  --configuration '{"Version": 1.0, "CrawlerOutput": {"Tables":
 {"AddOrUpdateBehavior": "MergeNewColumns"}}}'
```

다음 예제는 기존 스키마를 변경하지 않고 새 열을 추가하지 않도록 크롤러를 구성하는 방법을 보여줍니다.

```
aws glue update-crawler \
  --name myCrawler \
  --schema-change-policy UpdateBehavior=LOG \
  --configuration '{"Version": 1.0, "CrawlerOutput": {"Partitions":
 { "AddOrUpdateBehavior": "InheritFromTable" }}}'
```

API

크롤러가 실행될 때 테이블 스키마가 전혀 변하지 않도록 하려면 LOG에 스키마 변경 정책을 설정합니다.

API를 사용하여 크롤러를 구성하면 다음 파라미터를 설정합니다.

- SchemaChangePolicy 구조의 UpdateBehavior 필드를 LOG로 설정합니다.
- 예를 들어, 크롤러 API의 다음 JSON 객체를 나타내는 문자열로 Configuration 필드를 설정합니다.

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" }
   }
}
```

각 Amazon S3 포함 경로에 대해 단일 스키마 생성

기본적으로 크롤러가 Amazon S3에 저장된 데이터용 테이블을 정의하면 데이터 호환성과 스키마 유사성을 모두 고려합니다. 고려되는 데이터 호환성 요인에는 데이터가 동일 포맷(예: JSON), 동일 압축 포맷(예: GZIP), Amazon S3 경로의 구조 및 기타 데이터 속성인지 여부가 포함됩니다. 스키마 유사성은 개별 Amazon S3 객체의 스키마가 얼마나 근접하게 유사한지를 측정합니다.

이 옵션의 이해를 돕기 위해 포함 경로가 `s3://amzn-s3-demo-bucket/table1/`인 크롤러를 정의한다고 가정해 보겠습니다. 이 크롤러는 실행 시 다음과 같은 특성을 가진 JSON 파일 두 개를 찾습니다.

- 파일 1 – `S3://amzn-s3-demo-bucket/table1/year=2017/data1.json`
- 파일 내용 – `{“A”: 1, “B”: 2}`
- 스키마 – `A:int, B:int`

- 파일 2 – `S3://amzn-s3-demo-bucket/table1/year=2018/data2.json`
- 파일 내용 – `{“C”: 3, “D”: 4}`
- 스키마 – `C: int, D: int`

스키마가 충분히 비슷하지 않으므로 기본적으로 크롤러가 `year_2017` 및 `year_2018`이라는 두 테이블을 생성합니다. 하지만 Create a single schema for each S3 path(각 S3 경로에 대해 단일 스키마 생성) 옵션을 선택했으며 데이터가 호환되는 경우, 크롤러가 한 개 테이블을 생성합니다. 이 테이블에는 `A:int,B:int,C:int,D:int` 및 `partitionKey year:string`이라는 스키마가 있습니다.

AWS Management Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 데이터 카탈로그에서 크롤러를 선택합니다.
3. 새 크롤러를 구성하는 경우 출력 및 예약의 고급 옵션에서 각 S3 경로에 대해 단일 스키마 생성 옵션을 선택합니다.

AWS CLI

가능하면 공통 테이블 정의에 호환 스키마를 결합(`CombineCompatibleSchemas`)하도록 크롤러를 구성할 수 있습니다. 이 옵션을 사용하는 경우 크롤러가 데이터 호환성을 고려하지만, 지정된 포함 경로에서 Amazon S3 객체를 평가할 때 특정 스키마의 유사성을 무시합니다.

AWS CLI를 사용하여 크롤러를 구성할 때는 다음 구성 옵션을 설정합니다.

```
aws glue update-crawler \
   --name myCrawler \
   --configuration '{"Version": 1.0, "Grouping": {"TableGroupingPolicy":
 "CombineCompatibleSchemas" }}'
```

API

API를 사용하여 크롤러를 구성할 때는 다음 구성 옵션을 설정합니다.

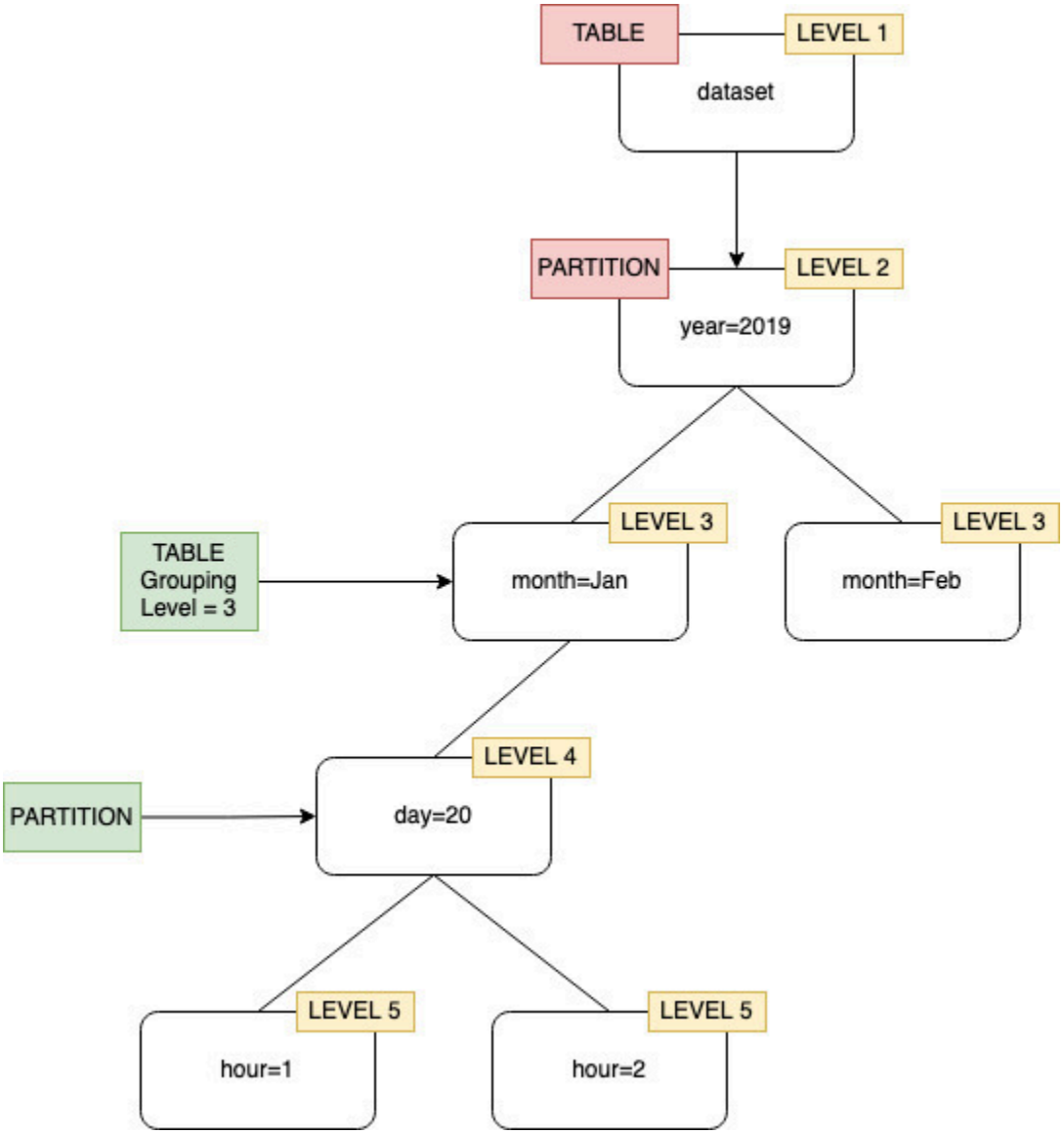
예를 들어, 크롤러 API의 다음 JSON 객체를 나타내는 문자열로 `Configuration` 필드를 설정합니다.

```
{
   "Version": 1.0,
   "Grouping": {
      "TableGroupingPolicy": "CombineCompatibleSchemas" }
}
```

테이블 위치와 파티션 수준 지정

기본적으로 크롤러가 Amazon S3에 저장된 데이터에 대한 테이블을 정의할 때 크롤러는 스키마를 병합하고 최상위 테이블(`year=2019`)을 생성하려고 시도합니다. 경우에 따라 크롤러가 폴더 `month=Jan`에 대한 테이블을 생성할 것으로 예상할 수 있지만 대신 형제 폴더(`month=Mar`)가 동일한 테이블에 병합되었기 때문에 크롤러가 파티션을 생성합니다.

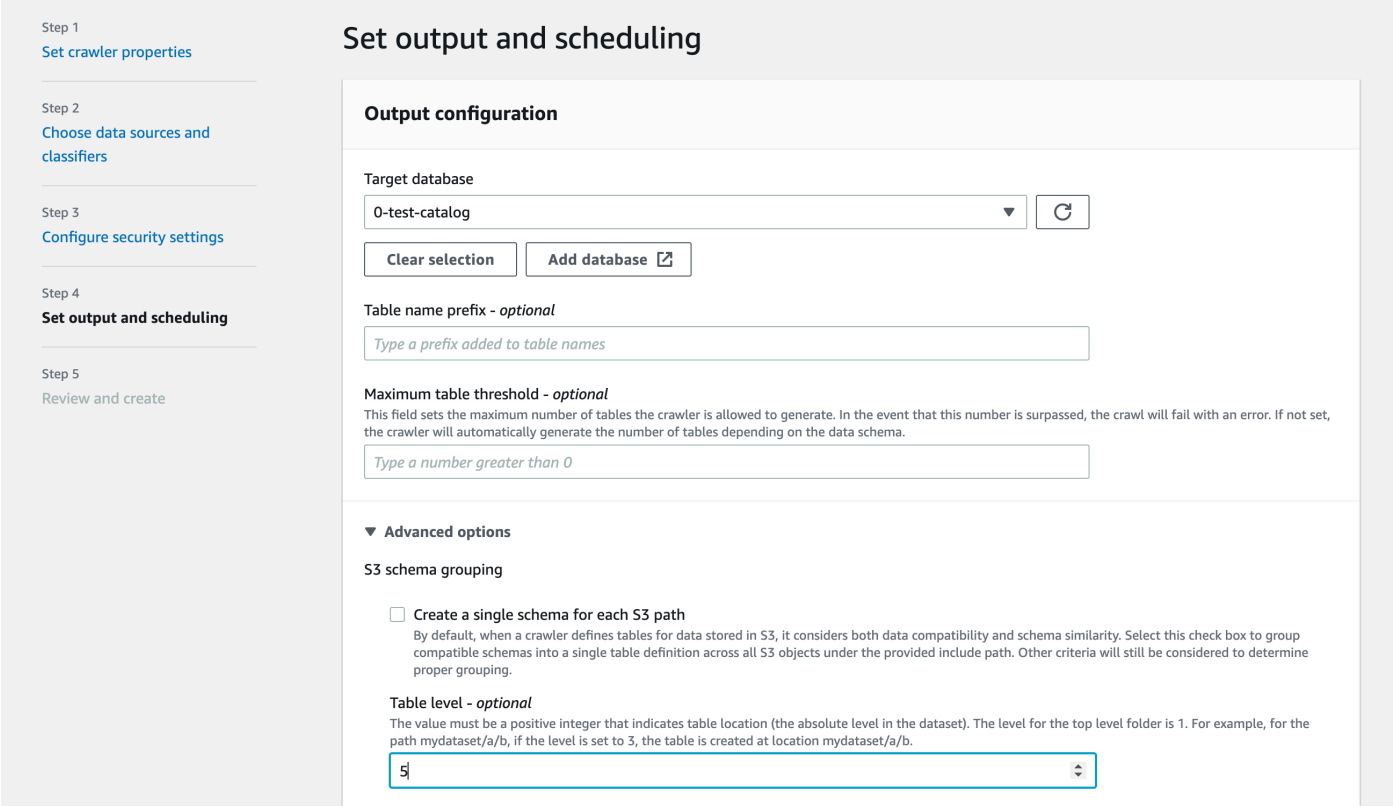
테이블 수준 크롤러 옵션을 사용하면 크롤러에 테이블의 위치와 파티션 생성 방법을 유연하게 알릴 수 있습니다. [테이블 수준(Table level)]을 지정하면 Amazon S3 버킷에서 해당 절대 수준으로 테이블이 생성됩니다.



콘솔에서 크롤러를 구성할 때 [테이블 수준(Table level)] 크롤러 옵션에 대한 값을 지정할 수 있습니다. 값은 테이블 위치(데이터 집합의 절대 수준)를 나타내는 양의 정수여야 합니다. 최상위 폴더의 수준은 1입니다. 예를 들어 경로 `mydataset/year/month/day/hour`의 경우 수준을 3으로 설정하면 `mydataset/year/month` 위치에 테이블이 생성됩니다.

AWS Management Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 데이터 카탈로그에서 크롤러를 선택합니다.
3. 크롤러를 구성할 때 출력 및 예약 설정의 고급 옵션에서 테이블 수준을 선택합니다.



AWS CLI

AWS CLI를 사용하여 크롤러를 구성하는 경우 예제 코드에 표시된 대로 `configuration` 파라미터를 설정합니다.

```
aws glue update-crawler \
  --name myCrawler \
  --configuration '{"Version": 1.0, "Grouping": { "TableLevelConfiguration": 2 }}'
```

API

API를 사용하여 크롤러를 구성하는 경우 다음 JSON 객체의 문자열 표현으로 `Configuration` 필드를 설정합니다. 예를 들면 다음과 같습니다.

```
configuration = jsonencode(
{
   "Version": 1.0,
   "Grouping": {
            TableLevelConfiguration = 2
        }
})
```

CloudFormation

이 예제에서는 콘솔에서 사용 가능한 테이블 레벨(Table level) 옵션을 CloudFormation 템플릿 내에서 설정합니다.

```
"Configuration": "{
    \"Version\":1.0,
    \"Grouping\":{\"TableLevelConfiguration\":2}
}"
```

크롤러가 생성할 수 있는 최대 테이블 수 지정

선택적으로 AWS Glue 콘솔 또는 AWS CLI를 통해 `TableThreshold`를 지정하여 크롤러가 생성할 수 있는 최대 테이블 수를 지정할 수 있습니다. 크롤링 중 크롤러가 탐지한 테이블이 이 입력 값보다 크면 크롤링이 실패하고 데이터 카탈로그에 데이터가 기록되지 않습니다.

이 파라미터는 크롤러에 의해 탐지되고 생성되는 테이블이 예상보다 훨씬 큰 경우에 유용합니다. 다음과 같은 여러 가지 이유가 있을 수 있습니다.

- AWS Glue 작업을 사용하여 Amazon S3 위치를 채우면 폴더와 같은 수준에 빈 파일이 생길 수 있습니다. 이러한 경우 이 Amazon S3 위치에서 크롤러를 실행하면 파일 및 폴더가 동일한 수준에 있기 때문에 크롤러가 여러 테이블을 생성합니다.
- `"TableGroupingPolicy": "CombineCompatibleSchemas"`를 설정하지 않으면 예상보다 많은 테이블이 생성될 수 있습니다.

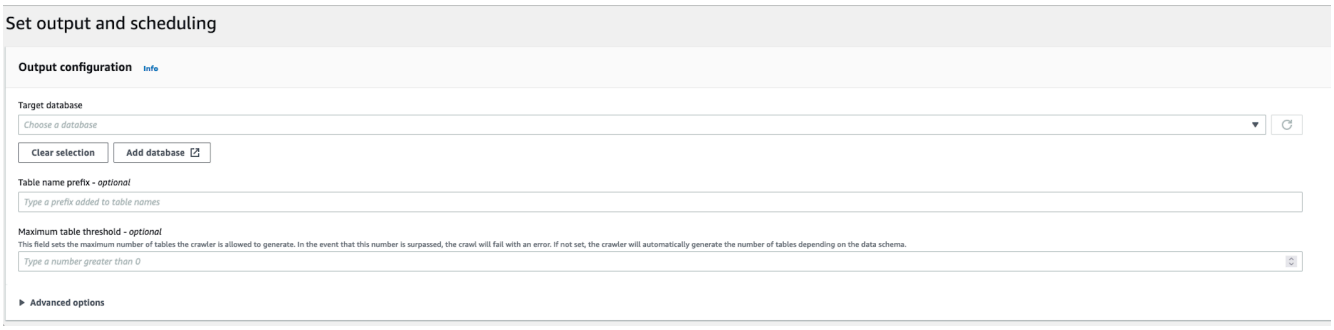
`TableThreshold`를 0보다 큰 정수 값으로 지정합니다. 이 값은 크롤러별로 구성됩니다. 즉, 모든 크롤링에 대해 이 값이 고려됩니다. 예: 크롤러의 `TableThreshold` 값이 5로 설정되어 있습니다. 각 크롤링에서 AWS Glue는 탐지된 테이블 수를 이 테이블 임곗값(5)과 비교하여 탐지된 테이블 수가 5보다 작으면 테이블을 데이터 카탈로그에 쓰고 AWS Glue 그렇지 않으면 데이터 카탈로그에 쓰지 않고 크롤링에 실패합니다.

AWS Management Console

AWS Management Console을 사용하여 **`TableThreshold`**를 설정하려면:

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

2. 크롤러를 구성할 때 출력 및 예약에서 최대 테이블 임계값을 크롤러가 생성할 수 있는 테이블 수로 설정합니다.



AWS CLI

AWS CLI를 사용하여 TableThreshold를 설정하려면:

```
aws glue update-crawler \
    --name myCrawler \
    --configuration '{"Version": 1.0, "CrawlerOutput": {"Tables":
 { "TableThreshold": 5 }}}'
```

API

API를 사용하여 TableThreshold를 설정하려면:

```
"{"Version":1.0,
"CrawlerOutput":
{"Tables":{"AddOrUpdateBehavior":"MergeNewColumns",
"TableThreshold":5}}}";
```

테이블 경로를 식별하고 데이터를 정리하는 데 도움이 되도록 오류 메시지가 기록됩니다. 다음은 테이블 수가 제공된 테이블 임곗값보다 커서 크롤러가 실패한 경우 계정의 예제 로그입니다.

```
Table Threshold value = 28, Tables detected - 29
```

CloudWatch에서는 탐지된 모든 테이블 위치를 INFO 메시지로 기록합니다. 오류가 실패 원인으로 기록됩니다.

```
ERROR com.amazonaws.services.glue.customerLogs.CustomerLogService - CustomerLogService
 received CustomerFacingException with message
```

```
The number of tables detected by crawler: 29 is greater than the table threshold value
 provided: 28. Failing crawler without writing to Data Catalog.
com.amazonaws.services.glue.exceptions.CustomerFacingInternalException: The number of
 tables detected by crawler: 29 is greater than the table threshold value provided:
 28.
Failing crawler without writing to Data Catalog.
```

Lake Formation 자격 증명을 사용하도록 크롤러 구성

AWS Lake Formation 자격 증명을 사용하여 동일한 AWS 계정 또는 다른 AWS 계정 내의 기본 Amazon S3 위치가 있는 데이터 카탈로그 테이블 또는 Amazon S3 데이터 스토어에 액세스하도록 크롤러를 구성할 수 있습니다. 크롤러와 데이터 카탈로그 테이블이 동일한 계정에 있는 경우 기존 데이터 카탈로그 테이블을 크롤러의 대상으로 구성할 수 있습니다. 현재 데이터 카탈로그 테이블을 크롤러의 대상으로 사용할 때 단일 카탈로그 테이블이 있는 단일 카탈로그 대상만 허용됩니다.

> **Note**
>
> 데이터 카탈로그 테이블을 크롤러 대상으로 정의할 때 데이터 카탈로그 테이블의 기본 위치가 Amazon S3 위치인지 확인합니다. Lake Formation 자격 증명을 사용하는 크롤러는 기본 Amazon S3 위치가 있는 데이터 카탈로그 대상만 지원합니다.

크롤러와 등록된 Amazon S3 위치 또는 데이터 카탈로그 테이블이 동일한 계정에 있는 경우 필요한 설정(계정 내 크롤링)

크롤러가 Lake Formation 자격 증명을 사용하여 데이터 스토어 또는 데이터 카탈로그 테이블에 액세스할 수 있도록 하려면 Lake Formation에 데이터 위치를 등록해야 합니다. 또한 크롤러의 IAM 역할에는 Amazon S3 버킷이 등록된 대상에서 데이터를 읽을 수 있는 권한이 있어야 합니다.

AWS Management Console 또는 AWS Command Line Interface(AWS CLI)를 사용하여 다음 구성 단계를 완료할 수 있습니다.

AWS Management Console

1. 크롤러 소스에 액세스하도록 크롤러를 구성하기 전에 데이터 스토어 또는 데이터 카탈로그의 데이터 위치를 Lake Formation에 등록합니다. Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)에서 크롤러가 정의된 AWS 계정에서 Amazon S3 위치를 데이터 레이크의 루트 위치로 등록합니다. 자세한 내용을 알아보려면 Registering an Amazon S3 location(Amazon S3 위치 등록)을 참조하세요.

2. 크롤러가 Lake Formation의 대상에서 데이터를 읽을 수 있도록 크롤러 실행에 사용되는 IAM 역할에 Data location(데이터 위치) 권한을 부여합니다. 자세한 내용을 알아보려면 Granting data location permissions (same account)(데이터 위치 권한 부여(동일한 계정))를 참조하세요.
3. 출력 데이터베이스로 지정된 데이터베이스에 크롤러 역할 액세스 권한(`Create`)을 부여합니다. 자세한 내용을 알아보려면 Granting database permissions using the Lake Formation console and the named resource method(Lake Formation 콘솔 및 명명된 리소스 메서드를 사용하여 데이터베이스 권한 부여)를 참조하세요.
4. IAM 콘솔(https://console.aws.amazon.com/iam/)에서 크롤러에 대한 IAM 역할을 생성합니다. 역할에 `lakeformation:GetDataAccess` 정책을 추가합니다.
5. AWS Glue 콘솔(https://console.aws.amazon.com/glue/)에서 크롤러를 구성하는 동안 Use Lake Formation credentials for crawling Amazon S3 data source(Amazon S3 데이터 소스 크롤링에 Lake Formation 자격 증명 사용) 옵션을 선택합니다.

> **Note**
>
> accountId 필드는 계정 내 크롤링의 선택 사항입니다.

AWS CLI

```
aws glue --profile demo create-crawler --debug --cli-input-json '{
    "Name": "prod-test-crawler",
    "Role": "arn:aws:iam::111122223333:role/service-role/AWSGlueServiceRole-prod-
test-run-role",
    "DatabaseName": "prod-run-db",
    "Description": "",
    "Targets": {
    "S3Targets":[
                {
                 "Path": "s3://amzn-s3-demo-bucket"
                }
                ]
                },
   "SchemaChangePolicy": {
      "UpdateBehavior": "LOG",
      "DeleteBehavior": "LOG"
  },
  "RecrawlPolicy": {
    "RecrawlBehavior": "CRAWL_EVERYTHING"
  },
```

```
  "LineageConfiguration": {
    "CrawlerLineageSettings": "DISABLE"
  },
  "LakeFormationConfiguration": {
    "UseLakeFormationCredentials": true,
    "AccountId": "111122223333"
  },
  "Configuration": {
           "Version": 1.0,
           "CrawlerOutput": {
             "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
             "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
           },
           "Grouping": { "TableGroupingPolicy": "CombineCompatibleSchemas" }
         },
  "CrawlerSecurityConfiguration": "",
  "Tags": {
    "KeyName": ""
  }
}'
```

크롤러와 등록된 Amazon S3 위치가 다른 계정에 있는 경우 필요한 설정(크로스 계정 크롤링)

크롤러가 Lake Formation 자격 증명을 사용하여 다른 계정의 데이터 스토어에 액세스할 수 있도록 하려면 먼저 Lake Formation에 Amazon S3 데이터 위치를 등록해야 합니다. 그리고 다음 단계에 따라 크롤러의 계정에 데이터 위치 권한을 부여합니다.

AWS Management Console 또는 AWS CLI를 사용하여 다음 단계를 완료할 수 있습니다.

AWS Management Console

1. Amazon S3 위치가 등록된 계정(계정 B)에서
   a. Lake Formation에 Amazon S3 경로를 등록합니다. 자세한 내용을 알아보려면 Registering an Amazon S3 location(Amazon S3 위치 등록)을 참조하세요.
   b. 크롤러가 실행될 계정(계정 A)에 Data location(데이터 위치) 권한을 부여합니다. 자세한 내용을 알아보려면 Granting data location permissions(데이터 위치 권한 부여)를 참조하세요.
   c. 기본 위치를 대상 Amazon S3 위치로 사용하여 Lake Formation에 빈 데이터베이스를 생성합니다. 자세한 내용을 알아보려면 Creating a database(데이터베이스 생성)를 참조하세요.

d. 계정 A(크롤러가 실행될 계정)에 이전 단계에서 생성한 데이터베이스에 대한 액세스 권한을 부여합니다. 자세한 내용을 알아보려면 Granting database permissions(데이터베이스 권한 부여)를 참조하세요.

2. 크롤러가 생성되어 실행될 계정(계정 A)에서

a. AWS RAM 콘솔을 사용하여 외부 계정(계정 B)에서 공유된 데이터베이스를 수락합니다. 자세한 내용을 알아보려면 AWS Resource Access Manager에서 리소스 공유 초대 수락을 참조하세요.

b. 크롤러에 대한 IAM 역할을 생성합니다. 역할에 `lakeformation:GetDataAccess` 정책을 추가합니다.

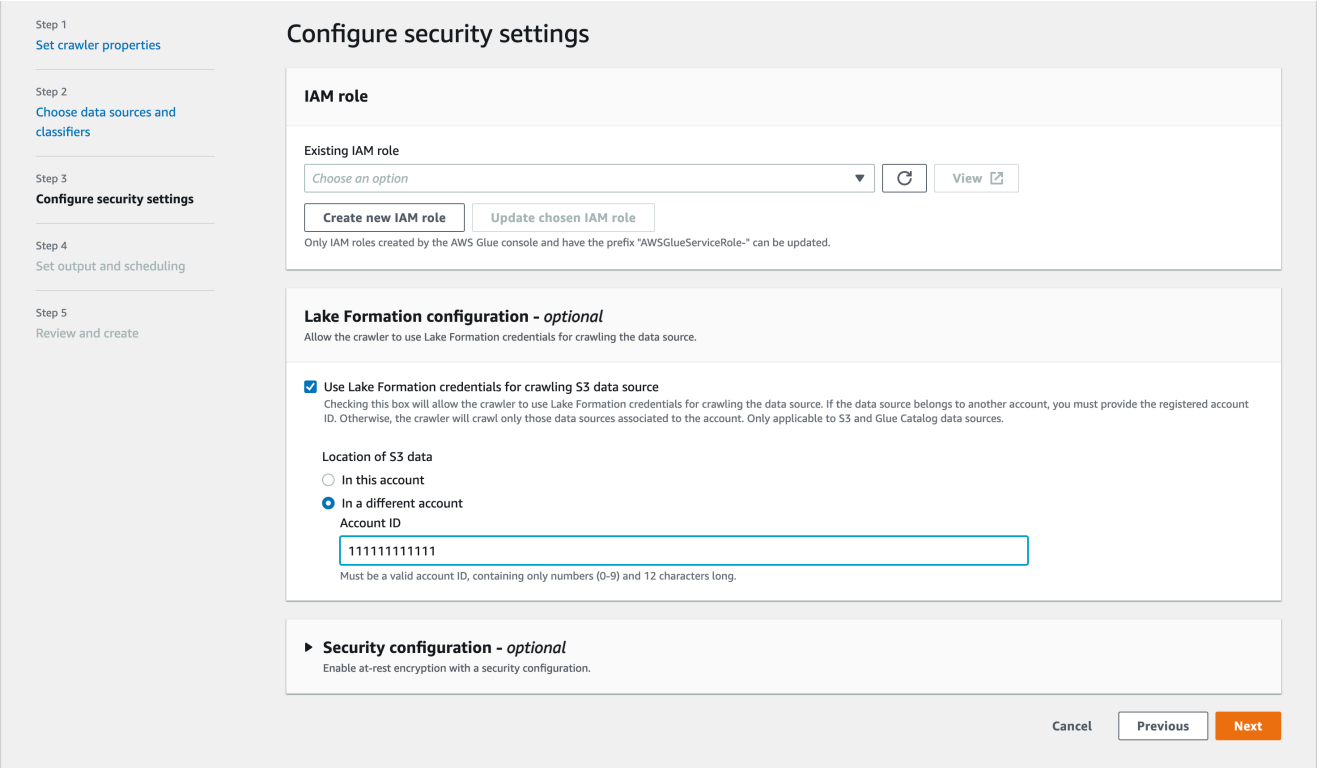
c. Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)에서 크롤러가 Lake Formation의 대상에서 데이터를 읽을 수 있도록 크롤러 실행에 사용되는 IAM 역할에 대상 Amazon S3 위치에 대한 Data location(데이터 위치) 권한을 부여합니다. 자세한 내용을 알아보려면 Granting data location permissions(데이터 위치 권한 부여)를 참조하세요.

d. 공유 데이터베이스에 리소스 링크를 생성합니다. 자세한 내용을 알아보려면 Create a resource link(리소스 링크 만들기)를 참조하세요.

e. 공유 데이터베이스 및 (`Describe`) 리소스 링크에 대한 크롤러 역할 액세스 권한(`Create`)을 부여합니다. 리소스 링크는 크롤러의 출력에 지정됩니다.

f. AWS Glue 콘솔(https://console.aws.amazon.com/glue/)에서 크롤러를 구성하는 동안 Use Lake Formation credentials for crawling Amazon S3 data source(Amazon S3 데이터 소스 크롤링에 Lake Formation 자격 증명 사용) 옵션을 선택합니다.

   크로스 계정 크롤링의 경우 대상 Amazon S3 위치가 Lake Formation에 등록된 AWS 계정 ID를 지정합니다. 계정 내 크롤링의 경우 accountId 필드는 선택 사항입니다.



AWS CLI

```
aws glue --profile demo create-crawler --debug --cli-input-json '{
    "Name": "prod-test-crawler",
    "Role": "arn:aws:iam::111122223333:role/service-role/AWSGlueServiceRole-prod-
test-run-role",
    "DatabaseName": "prod-run-db",
    "Description": "",
    "Targets": {
    "S3Targets":[
                {
                 "Path": "s3://amzn-s3-demo-bucket"
                }
                ]
                },
   "SchemaChangePolicy": {
      "UpdateBehavior": "LOG",
      "DeleteBehavior": "LOG"
  },
  "RecrawlPolicy": {
    "RecrawlBehavior": "CRAWL_EVERYTHING"
  },
```

```
  "LineageConfiguration": {
    "CrawlerLineageSettings": "DISABLE"
  },
  "LakeFormationConfiguration": {
    "UseLakeFormationCredentials": true,
    "AccountId": "111111111111"
  },
  "Configuration": {
           "Version": 1.0,
           "CrawlerOutput": {
             "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
             "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
           },
           "Grouping": { "TableGroupingPolicy": "CombineCompatibleSchemas" }
         },
  "CrawlerSecurityConfiguration": "",
  "Tags": {
    "KeyName": ""
  }
}'
```

> **Note**
>
> - Lake Formation 자격 증명을 사용하는 크롤러는 Amazon S3 및 데이터 카탈로그 대상에 대해서만 지원됩니다.
> - Lake Formation 자격 증명 벤딩을 사용하는 대상의 경우 기본 Amazon S3 위치가 동일한 버킷에 속해야 합니다. 예를 들어, 고객은 모든 대상 위치가 동일한 버킷(amzn-s3-demo-bucket1) 아래에 있는 한 여러 대상(s3://amzn-s3-demo-bucket1/folder1, s3://amzn-s3-demo-bucket1/folder2)을 사용할 수 있습니다. 서로 다른 버킷(s3://amzn-s3-demo-bucket1/folder1, s3://amzn-s3-demo-bucket2/folder2)을 지정하는 것은 허용되지 않습니다.
> - 현재 데이터 카탈로그 대상 크롤러의 경우 단일 카탈로그 테이블이 있는 단일 카탈로그 대상만 허용됩니다.

Amazon S3 이벤트 알림을 사용하여 크롤링 가속화

Amazon S3 또는 데이터 카탈로그 대상의 객체를 나열하는 대신 Amazon S3 이벤트를 사용하여 변경 사항을 찾도록 크롤러를 구성할 수 있습니다. 이 기능은 전체 Amazon S3 또는 데이터 카탈로그 대상

을 나열하는 대신 Amazon S3 이벤트를 통해 이벤트를 트리거한 하위 폴더의 모든 파일을 나열하여 두 크롤링 간의 변경 사항을 식별하므로 다시 크롤링하는 시간이 단축됩니다.

첫 번째 크롤링은 대상의 모든 Amazon S3 객체를 나열합니다. 첫 번째 크롤링이 성공한 후에는 수동으로 또는 정해진 일정에 따라 다시 크롤링하도록 선택할 수 있습니다. 크롤러는 모든 객체를 나열하지 않고 해당 이벤트의 객체만 나열합니다.

대상이 데이터 카탈로그 테이블인 경우 크롤러는 변경 내용을 사용하여 데이터 카탈로그의 기존 테이블을 업데이트합니다(예: 테이블의 추가 파티션).

Amazon S3 이벤트 기반 크롤러로 이동할 경우의 이점은 다음과 같습니다.

- 대상의 모든 객체를 나열할 필요 없이 객체가 추가되거나 삭제되는 특정 폴더가 나열되므로 더 빠르게 다시 크롤링할 수 있습니다.
- 객체가 추가되거나 삭제되는 특정 폴더가 나열되므로 전체 크롤링 비용이 절감됩니다.

Amazon S3 이벤트 크롤링은 크롤러 일정에 따라 SQS 대기열의 Amazon S3 이벤트를 사용하여 실행됩니다. 대기열에 이벤트가 없으면 비용이 발생하지 않습니다. SQS 대기열이나 여러 소비자가 동일한 이벤트를 필요로 하는 경우 SNS와 SQS의 조합으로 바로 이동하도록 Amazon S3 이벤트를 구성할 수 있습니다. 자세한 내용은 the section called “Amazon S3 이벤트 알림을 위해 계정 설정” 섹션을 참조하세요.

이벤트 모드로 크롤러를 생성하고 구성한 후 첫 번째 크롤링은 Amazon S3 또는 데이터 카탈로그 대상을 모두 나열하여 목록 모드로 실행됩니다. "크롤링이 Amazon S3 이벤트를 사용하여 실행되고 있습니다."라는 로그는 첫 번째 크롤링이 성공적으로 끝난 후 크롤링이 Amazon S3 이벤트를 사용하여 작동하고 있음을 확인합니다.

Amazon S3 이벤트 크롤링을 생성하고 크롤링에 영향을 줄 수 있는 크롤러 속성을 업데이트하면 크롤링이 목록 모드로 작동하고 "크롤링이 S3 이벤트 모드로 실행되고 있지 않습니다."라는 로그가 추가됩니다.

> **Note**
>
> 사용할 수 있는 최대 메시지 수는 크롤링당 100,000개입니다.

고려 사항 및 제한 사항

Amazon S3 이벤트 알림을 사용하여 변경 사항을 찾도록 크롤러를 구성할 때 다음 고려 사항과 제한 사항이 적용됩니다.

- 삭제된 파티션의 중요 동작

  Data Catalog 테이블에서 Amazon S3 이벤트 크롤러를 사용하는 경우:

  - `DeletePartition` API 직접 호출을 사용하여 파티션을 삭제하는 경우 해당 파티션의 모든 S3 객체도 삭제하고, S3 이벤트 알림을 구성할 때 모든 객체 제거 이벤트를 선택해야 합니다. 삭제 이벤트가 구성되지 않은 경우 크롤러는 다음 실행 도중 삭제된 파티션을 다시 생성합니다.
- Amazon S3 대상이든 Data Catalog 대상이든 크롤러는 단일 대상만 지원합니다.
- 프라이빗 VPC의 SQS는 지원되지 않습니다.
- Amazon S3 샘플링은 지원되지 않습니다.
- 크롤러 대상은 Amazon S3 대상의 폴더이거나 데이터 카탈로그 대상에 대한 하나 이상의 AWS Glue 데이터 카탈로그 테이블이어야 합니다.
- 'everything' 경로 와일드카드는 지원되지 않습니다(s3://%).
- 데이터 카탈로그 대상의 경우 모든 카탈로그 테이블은 Amazon S3 이벤트 모드에 대해 동일한 Amazon S3 버킷을 가리켜야 합니다.
- 데이터 카탈로그 대상의 경우 카탈로그 테이블은 Delta Lake 형식(_symlink 폴더 포함 또는 카탈로그 테이블의 `InputFormat` 확인)의 Amazon S3 위치를 가리키지 않아야 합니다.

주제

- Amazon S3 이벤트 알림을 위해 계정 설정
- Amazon S3 대상에 대한 Amazon S3 이벤트 알림용 크롤러 설정
- 데이터 카탈로그 테이블에 대한 Amazon S3 이벤트 알림용 크롤러 설정

Amazon S3 이벤트 알림을 위해 계정 설정

다음 설정 태스크를 완료합니다. 괄호 안의 값은 스크립트에서 구성 가능한 설정을 가리킵니다.

1. Amazon S3 버킷의 이벤트 알림을 설정해야 합니다.

   자세한 내용은 Amazon S3 이벤트 알림을 참조하세요.
2. Amazon S3 이벤트 기반 크롤러를 사용하려면 S3 대상 및 SQS의 스토어와 동일한 접두사에서 필터링된 이벤트를 사용하여 Amazon S3 버킷에서 이벤트 알림을 사용해야 합니다. 연습: 알림용 버킷 구성 단계에 따라 콘솔을 통해 SQS와 이벤트 알림을 설정할 수 있습니다.
3. 크롤러가 사용하는 역할에 다음 SQS 정책을 추가합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl",
                "sqs:ListDeadLetterSourceQueues",
                "sqs:ReceiveMessage",
                "sqs:GetQueueAttributes",
                "sqs:ListQueueTags",
                "sqs:SetQueueAttributes",
                "sqs:PurgeQueue"
            ],
            "Resource": "arn:aws:sqs:{region}:{accountID}:cfn-sqs-queue"
        }
    ]
}
```

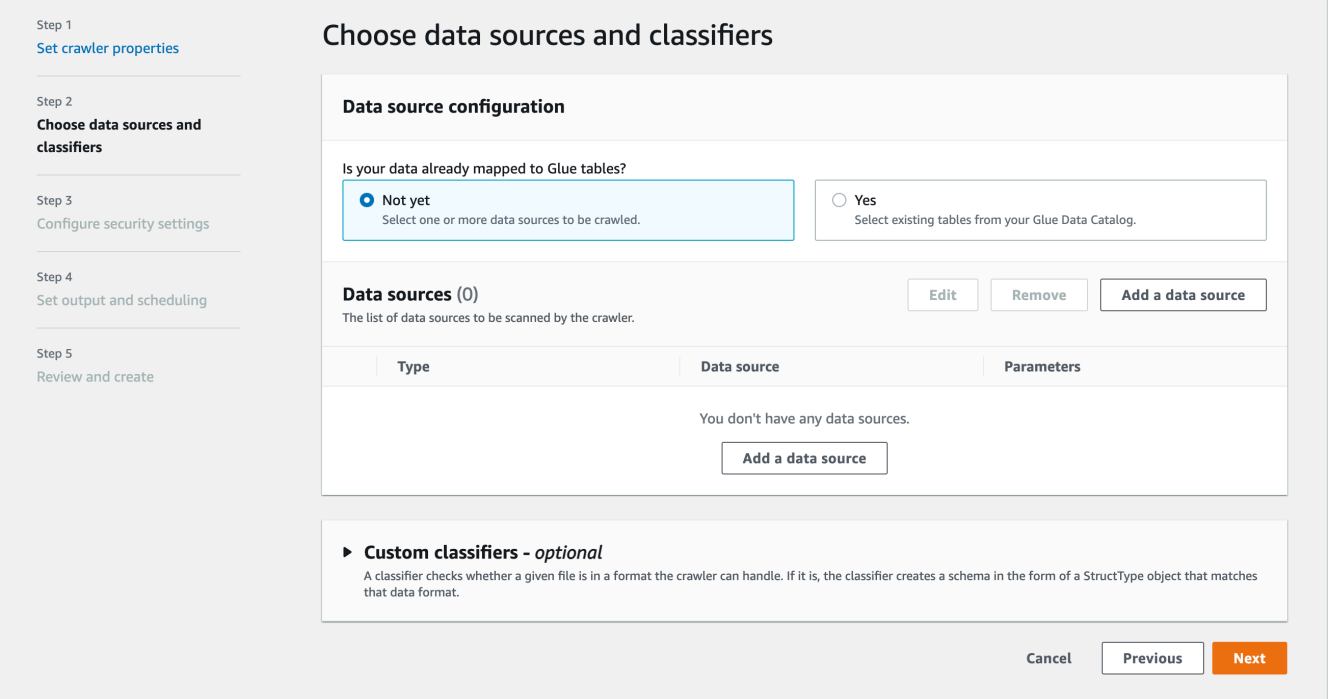
Amazon S3 대상에 대한 Amazon S3 이벤트 알림용 크롤러 설정

AWS Management Console 또는 AWS CLI를 사용하여 Amazon S3 대상에 대한 Amazon S3 이벤트 알림용 크롤러를 설정하려면 다음 단계를 따르세요.

AWS Management Console

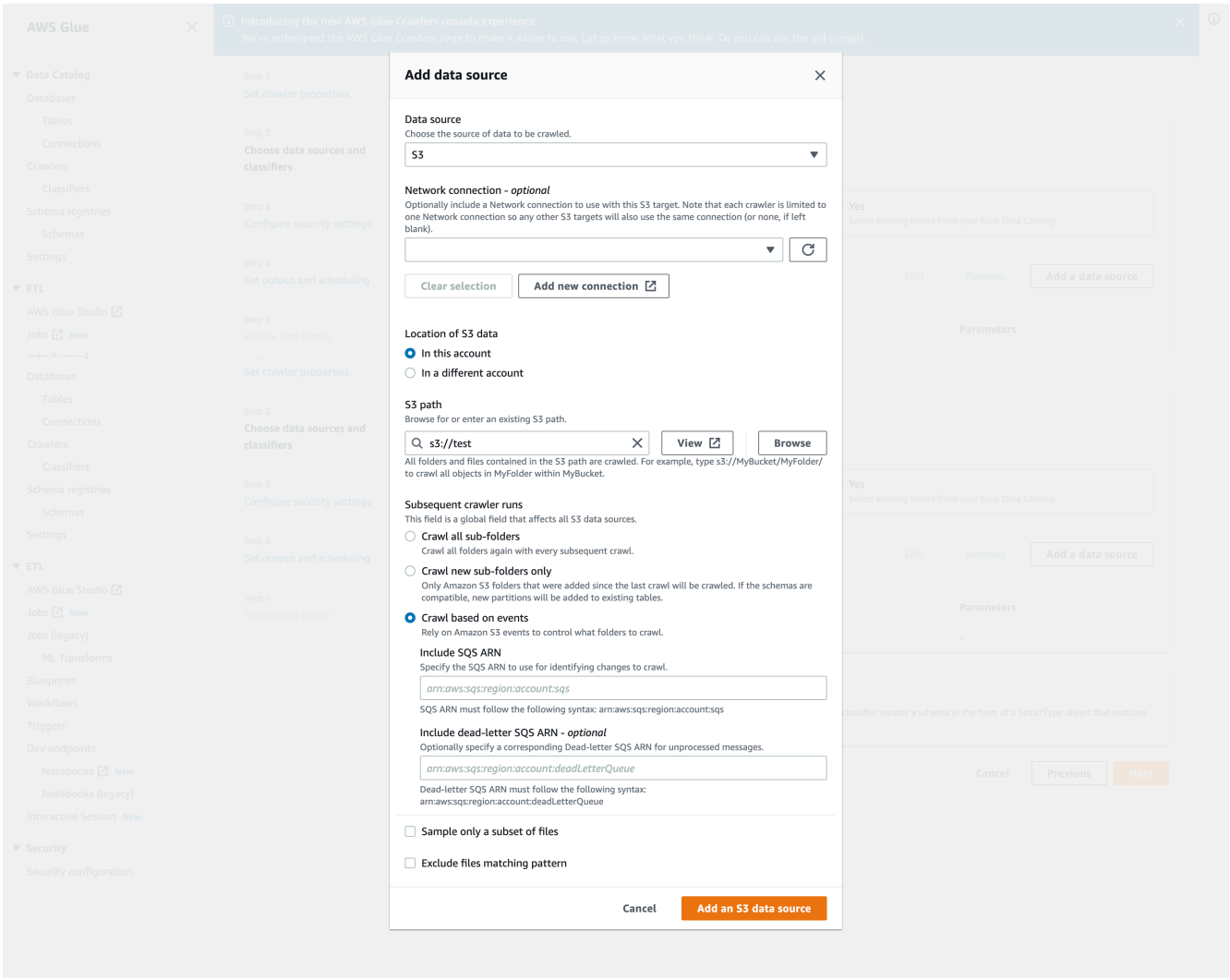
1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/guardduty/에서 GuardDuty 콘솔을 엽니다.
2. 크롤러 속성을 설정합니다. 자세한 내용을 알아보려면 AWS Glue 콘솔에서 크롤러 구성 옵션 설정을 참조하세요.
3. Data source configuration(데이터 소스 구성) 섹션에 Is your data already mapped to AWS Glue tables?라는 메시지가 표시됩니다.

   기본적으로 Not yet(아직)이 이미 선택되어 있습니다. Amazon S3 데이터 소스를 사용 중이고 데이터가 AWS Glue 테이블에 아직 매핑되지 않았으므로 이 항목을 기본값으로 둡니다.
4. Data sources(데이터 소스) 섹션에서 Add a data source(데이터 소스 추가)를 선택합니다.



5. Add data source(데이터 소스 추가) 모달에서 Amazon S3 데이터 소스를 구성합니다.

   - Data source(데이터 소스): 기본적으로 Amazon S3가 선택됩니다.
   - Network connection(네트워크 연결)(선택 사항): Add new connection(새 연결 추가)을 선택합니다.
   - Location of Amazon S3 data(Amazon S3 데이터 위치): 기본적으로 In this account(이 계정에서)가 선택됩니다.
   - Amazon S3 path(Amazon S3 경로): 폴더와 파일이 크롤링되는 Amazon S3 경로를 지정합니다.
   - Subsequent crawler runs(후속 크롤러 실행): 크롤러에 대한 Amazon S3 이벤트 알림을 사용하려면 Crawl based on events(이벤트 기반 크롤링)를 선택합니다.
   - Include SQS ARN(SQS ARN 포함): 유효한 SQS ARN을 포함하는 데이터 스토어 파라미터를 지정합니다. (예: `arn:aws:sqs:region:account:sqs`).
   - Include dead-letter SQS ARN(배달 못한 편지 SQS ARN 포함)(선택 사항): 유효한 Amazon 배달 못한 편지 SQS ARN을 지정합니다. (예: `arn:aws:sqs:region:account:deadLetterQueue`).
   - Add an Amazon S3 data source(Amazon S3 데이터 소스 추가)를 선택합니다.



AWS CLI

다음은 이벤트 알림을 사용하여 Amazon S3 대상 버킷을 크롤링하도록 크롤러를 구성하는 Amazon S3 AWS CLI 호출의 예입니다.

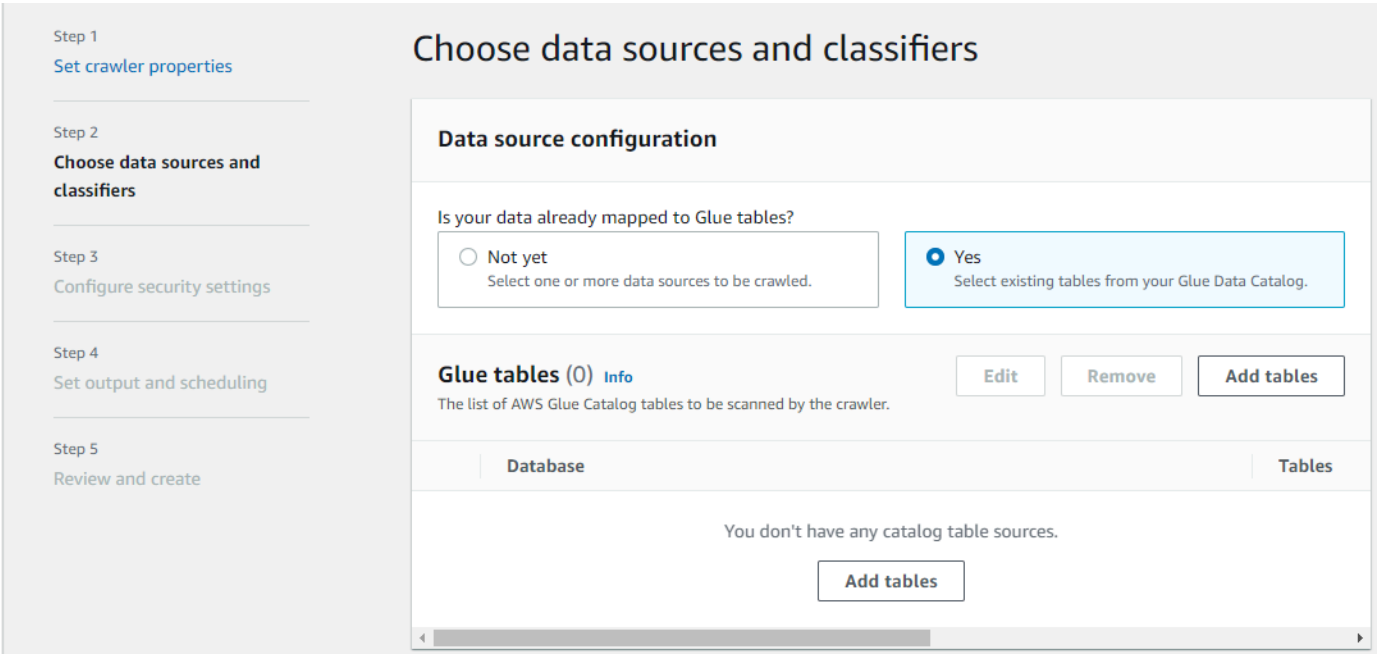
```
Create Crawler:
aws glue update-crawler \
    --name myCrawler \
    --recrawl-policy RecrawlBehavior=CRAWL_EVENT_MODE \
    --schema-change-policy UpdateBehavior=UPDATE_IN_DATABASE,DeleteBehavior=LOG
    --targets '{"S3Targets":[{"Path":"s3://amzn-s3-demo-bucket/", "EventQueueArn":
 "arn:aws:sqs:us-east-1:012345678910:MyQueue"}]}'
```

데이터 카탈로그 테이블에 대한 Amazon S3 이벤트 알림용 크롤러 설정

데이터 카탈로그 테이블이 있는 경우, AWS Glue 콘솔을 사용하여 Amazon S3 이벤트 알림용 크롤러를 설정합니다.

1. 크롤러 속성을 설정합니다. 자세한 내용을 알아보려면 AWS Glue 콘솔에서 크롤러 구성 옵션 설정을 참조하세요.
2. Data source configuration(데이터 소스 구성) 섹션에 Is your data already mapped to AWS Glue tables?라는 메시지가 표시됩니다.

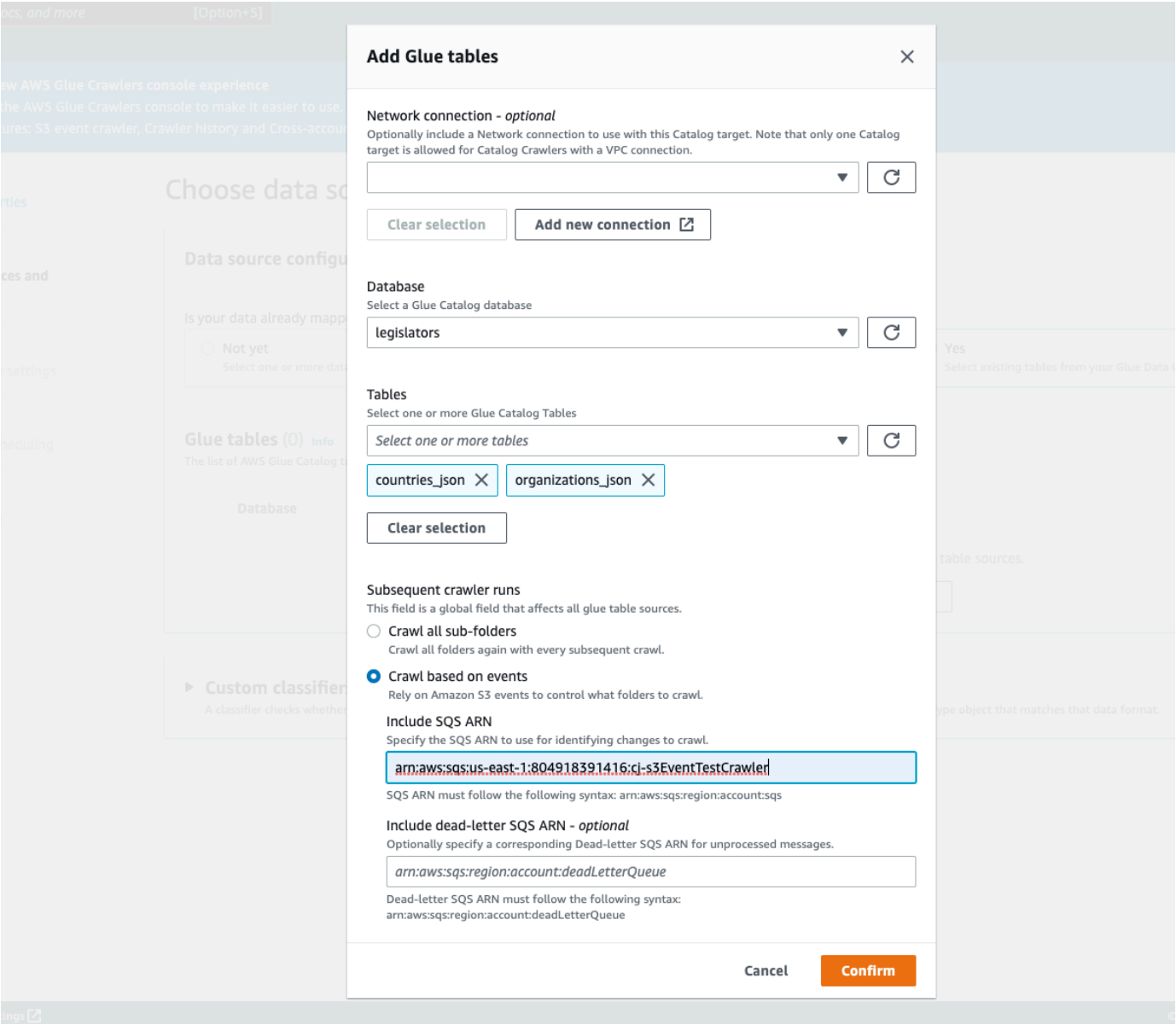
   데이터 카탈로그의 기존 테이블을 데이터 소스로 선택하려면 예(Yes)를 선택합니다.
3. Glue 테이블(Glue tables) 섹션에서 테이블 추가(Add tables)를 선택합니다.



4. 테이블 추가(Add table) 모달에서 데이터베이스 및 테이블을 구성합니다.

   - Network connection(네트워크 연결)(선택 사항): Add new connection(새 연결 추가)을 선택합니다.
   - 데이터베이스(Database): 데이터 카탈로그의 데이터베이스를 선택합니다.
   - 테이블(Tables): 데이터 카탈로그의 해당 데이터베이스에서 하나 이상의 테이블을 선택합니다.

- Subsequent crawler runs(후속 크롤러 실행): 크롤러에 대한 Amazon S3 이벤트 알림을 사용하려면 Crawl based on events(이벤트 기반 크롤링)를 선택합니다.
- Include SQS ARN(SQS ARN 포함): 유효한 SQS ARN을 포함하는 데이터 스토어 파라미터를 지정합니다. (예: `arn:aws:sqs:region:account:sqs`).
- Include dead-letter SQS ARN(배달 못한 편지 SQS ARN 포함)(선택 사항): 유효한 Amazon 배달 못한 편지 SQS ARN을 지정합니다. (예: `arn:aws:sqs:region:account:deadLetterQueue`).
- 확인(Confirm)을 선택합니다.



튜토리얼: AWS Glue 크롤러 추가

이 AWS Glue 시나리오에서는 주요 항공사의 도착 데이터를 분석하여 월별 출발 공항의 인기도를 계산해야 합니다. Amazon S3에 저장된 CSV 포맷의 2016년 항공편 데이터가 있습니다. 데이터를 변환하고 분석하기 전에 해당 메타데이터를 AWS Glue Data Catalog에 분류합니다.

이 튜토리얼에서는 Amazon S3 비행 로그에서 메타데이터를 유추하고 Data Catalog에 테이블을 생성하는 크롤러를 추가해 보겠습니다.

주제

- 사전 조건
- 1단계: 크롤러 추가
- 2단계: 크롤러 실행
- 3단계: AWS Glue Data Catalog 객체 보기

사전 조건

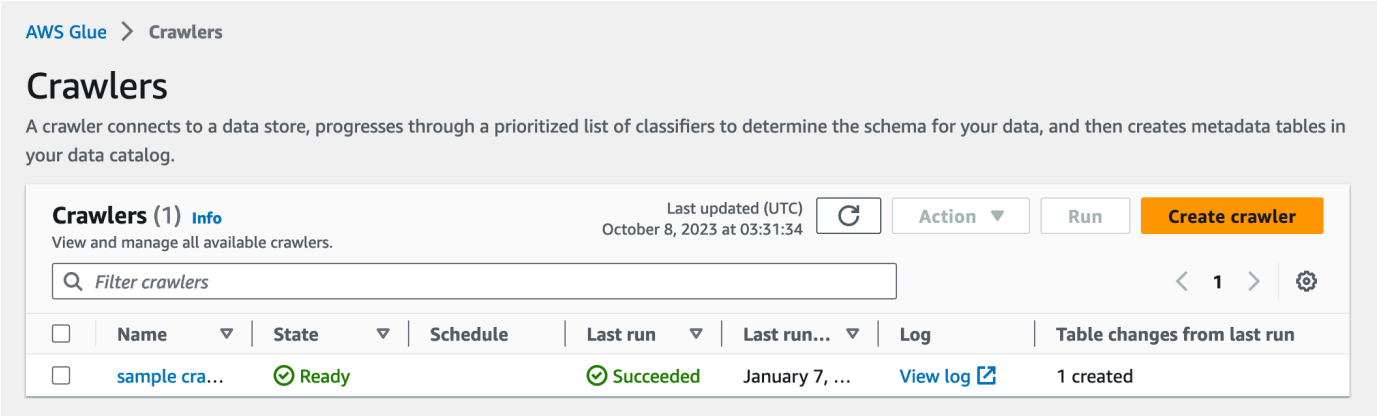
이 튜토리얼에서는 AWS 계정이 있고 AWS Glue에 대한 액세스 권한이 있다고 가정합니다.

1단계: 크롤러 추가

다음 단계에 따라 Amazon S3 저장된 CSV 파일에서 메타데이터를 추출하는 크롤러를 구성하고 실행합니다.

Amazon S3에 저장된 파일을 읽는 크롤러를 생성하려면

1. AWS Glue 서비스 콘솔의 왼쪽 메뉴에서 [크롤러(Crawlers)]를 선택합니다.
2. 크롤러 페이지에서 크롤러 생성을 선택합니다. 그러면 크롤러 세부 정보를 묻는 일련의 페이지가 시작됩니다.



3. [크롤러 이름(Crawler name)] 필드에 **Flights Data Crawler**를 입력하고 [다음(Next)]을 선택합니다.

   크롤러는 분류자를 호출하여 데이터의 스키마를 추론합니다. 이 튜토리얼에서는 기본적으로 CSV에 기본 제공 분류자를 사용합니다.

4. 크롤러 소스 유형으로 [데이터 스토어(Data stores)]를 선택하고 [다음(Next)]을 선택합니다.

5. 이제 크롤러가 데이터를 가리키도록 하겠습니다. [데이터 스토어 추가(Add a data store)] 페이지에서 Amazon S3 데이터 스토어를 선택합니다. 이 튜토리얼에서는 연결을 사용하지 않으므로 [연결(Connection)] 필드가 표시되면 비워 둡니다.

   [데이터 크롤링(Crawl data in)] 옵션으로 [다른 계정으로 지정된 경로(Specified path in another account)]를 선택합니다. 그런 다음 [포함 경로(Include path)]에 크롤러가 항공편 데이터를 찾을 수 있는 경로인 **s3://crawler-public-us-east-1/flight/2016/csv**를 입력합니다. 경로를 입력하면 이 필드의 제목이 [포함 경로(Include path)]로 바뀝니다. Next(다음)를 선택합니다.

6. 단일 크롤러로 여러 데이터 스토어를 크롤링할 수 있습니다. 그러나 이 튜토리얼에서는 단일 데이터 스토어만 사용하므로 [아니요(No)]를 선택한 후 [다음(Next)]을 선택합니다.

7. 크롤러는 데이터 스토어에 액세스하고 AWS Glue Data Catalog에서 객체를 생성할 수 있는 권한이 필요합니다. 이러한 권한을 구성하려면 [IAM 역할 생성(Create an IAM role)]을 선택합니다. IAM 역할 이름은 `AWSGlueServiceRole-`로 시작하고 필드에 역할 이름의 마지막 부분을 입력합니다. **CrawlerTutorial**을 입력한 다음 [다음(Next)]을 선택합니다.

   > **Note**
   >
   > IAM 역할을 생성하려면 AWS 사용자에게 `CreateRole`, `CreatePolicy` 및 `AttachRolePolicy` 권한이 있어야 합니다.

   마법사는 `AWSGlueServiceRole-CrawlerTutorial`이라는 IAM 역할을 생성하고 AWS 관리형 정책 `AWSGlueServiceRole`을 이 역할에 연결하고 Amazon S3 위치 `s3://crawler-public-us-east-1/flight/2016/csv`에 대한 읽기 액세스를 허용하는 인라인 정책을 추가합니다.

8. 크롤러에 대한 일정을 생성합니다. [빈도(Frequency)]에서 [온디맨드로 실행(Run on demand)]을 선택하고 [다음(Next)]을 선택합니다.

9. 크롤러가 Data Catalog에 테이블을 생성합니다. 테이블은 Data Catalog의 데이터베이스에 포함됩니다. 먼저 [데이터베이스 추가(Add database)]를 선택하여 데이터베이스를 생성합니다. 팝업 창에서 데이터베이스 이름으로 **test-flights-db**를 입력하고 [생성(Create)]을 선택합니다.

그런 다음 [테이블에 추가된 접두사(Prefix added to tables)]에 **flights**를 입력합니다. 나머지 옵션에 기본값을 사용하고 [다음(Next)]을 선택합니다.

10. []크롤러 추가(Add crawler)] 마법사에서 선택 내용을 확인합니다. 실수가 있는 경우 [뒤로(Back)]를 클릭하여 이전 페이지로 돌아가서 변경할 수 있습니다.

    정보를 검토한 후 [마침(Finish)]을 선택하여 크롤러를 생성합니다.

2단계: 크롤러 실행

크롤러를 생성하면 마법사가 크롤러 보기 페이지로 이동합니다. 온디맨드 일정으로 크롤러를 생성하기 때문에 크롤러를 실행할 수 있는 옵션이 제공됩니다.

크롤러를 실행하려면

1. 이 페이지 상단 근처에 있는 배너를 통해 크롤러가 생성되었음을 알리고 지금 실행할 것인지 묻습니다. [지금 실행?(Run it now?)]을 선택하여 크롤러를 실행합니다.

   배너가 변경되어 크롤러에 대해 “실행 시도 중(Attempting to run)” 및 “실행 중(Running)” 메시지가 표시됩니다. 크롤러 실행이 시작되면 배너가 사라지고 크롤러 디스플레이가 업데이트되어 크롤러에 대해 시작 중(Starting) 상태가 표시됩니다. 잠시 후 새로 고침 아이콘을 클릭하여 테이블에 표시된 크롤러의 상태를 업데이트할 수 있습니다.
2. 크롤러가 완료되면 크롤러의 변경 사항을 설명하는 새 배너가 나타납니다. test-flights-db링크를 선택하여 Data Catalog 객체를 볼 수 있습니다.

3단계: AWS Glue Data Catalog 객체 보기

크롤러는 소스 위치에서 데이터를 읽고 Data Catalog에 테이블을 생성합니다. 테이블은 해당 스키마를 포함한 데이터를 나타내는 메타데이터 정의입니다. Data Catalog의 테이블에는 데이터가 없습니다. 대신 이러한 테이블을 작업 정의에서 소스 또는 대상으로 사용합니다.

크롤러가 생성한 Data Catalog 객체를 보려면

1. 왼쪽 탐색의 [데이터 카탈로그(Data catalog)]에서 [데이터베이스(Databases)]를 선택합니다. 여기에서 크롤러에 의해 생성된 `flights-db` 데이터베이스를 볼 수 있습니다.
2. 왼쪽 탐색의 [데이터 카탈로그(Data catalog)] 및 [데이터베이스(Databases)] 아래에서 [테이블(Tables)]을 선택합니다. 여기에서 크롤러가 생성한 `flightscsv` 테이블을 볼 수 있습니다. 테이

블 이름을 선택하면 테이블 설정, 파라미터 및 속성을 볼 수 있습니다. 이 보기에서 아래로 스크롤하면 테이블의 열 및 데이터 유형에 대한 정보인 스키마를 볼 수 있습니다.

3. 테이블 보기 페이지에서 [파티션 보기(View partitions)]를 선택하면 데이터에 대해 생성된 파티션을 볼 수 있습니다. 첫 번째 열은 파티션 키입니다.

수동으로 메타데이터 정의

AWS Glue 데이터 카탈로그는 데이터 소스 및 데이터 세트에 대한 메타데이터를 저장하는 중앙 리포지토리입니다. 크롤러가 지원되는 데이터 소스의 메타데이터를 자동으로 크롤링하고 채울 수 있지만 특정 시나리오에서는 데이터 카탈로그에서 메타데이터를 수동으로 정의해야 할 수 있습니다.

- 지원되지 않는 데이터 형식 - 크롤러가 지원하지 않는 데이터 소스가 있는 경우 데이터 카탈로그에서 해당 데이터 소스의 메타데이터를 수동으로 정의해야 합니다.
- 사용자 지정 메타데이터 요구 사항 - AWS Glue 크롤러에서는 사전 정의된 규칙 및 규약을 기반으로 메타데이터를 유추합니다. AWS Glue 크롤러 추론 메타데이터에 포함되지 않는 특정 메타데이터 요구 사항이 있는 경우 필요에 맞게 메타데이터를 수동으로 정의할 수 있습니다.
- 데이터 거버넌스 및 표준화 - 데이터 거버넌스, 규정 준수 또는 보안상의 이유로 메타데이터 정의에 대한 보다 세밀한 제어가 필요할 수 있습니다. 메타데이터를 수동으로 정의하면 메타데이터가 조직의 표준 및 정책을 준수하는지 확인할 수 있습니다.
- 장래 데이터 수집을 위한 자리 표시자 - 즉시 사용할 수 없거나 액세스할 수 없는 데이터 소스가 있는 경우 빈 스키마 테이블을 자리 표시자로 생성할 수 있습니다. 데이터 소스를 사용할 수 있게 되면 사전 정의된 구조를 유지하면서 실제 데이터로 테이블을 채울 수 있습니다.

메타데이터를 수동으로 정의하려면 AWS Glue 콘솔, Lake Formation 콘솔, AWS Glue API 또는 AWS Command Line Interface(AWS CLI)를 사용할 수 있습니다. 데이터베이스, 테이블 및 파티션을 생성하고 열 이름, 데이터 유형, 설명 및 기타 속성과 같은 메타데이터 속성을 지정할 수 있습니다.

데이터베이스 생성

데이터베이스는 AWS Glue에서 메타데이터 테이블을 구성하는 데 사용됩니다. AWS Glue Data Catalog에 테이블을 정의할 때 테이블을 데이터베이스에 추가합니다. 테이블은 하나의 데이터베이스에만 있을 수 있습니다.

데이터베이스는 다양한 데이터 스토어의 데이터를 정의하는 테이블을 포함할 수 있습니다. 이 데이터에는 Amazon Simple Storage Service(Amazon S3)의 객체와 Amazon Relational Database Service의 관계형 테이블이 포함될 수 있습니다.

> **Note**
>
> AWS Glue Data Catalog에서 데이터베이스를 삭제할 때 데이터베이스의 모든 테이블도 삭제됩니다.

데이터베이스 목록을 보려면 AWS Management Console에 로그인하고 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다. [데이터베이스(Databases)]를 선택한 다음 목록에서 데이터베이스 이름을 선택하여 상세 정보를 봅니다.

AWS Glue 콘솔의 [데이터베이스(Databases)] 탭에서 데이터베이스를 추가, 편집 및 삭제할 수 있습니다.

- 새로운 데이터베이스를 생성하려면 [Add database(데이터베이스 추가)]를 선택하고 이름과 설명을 제공해야 합니다. Apache Hive와 같은 다른 메타데이터 스토어의 호환성을 고려해 이름은 소문자로 합니다.

  > **Note**
  >
  > Amazon Athena에서 데이터베이스로 액세스하고자 할 경우, 공급자 이름을 영숫자 문자와 밑줄 표시로 생성합니다. 자세한 내용을 알아보려면 테이블, 데이터베이스 및 열의 이름을 참조하세요.

- 데이터베이스에 대한 설명을 편집하려면 데이터베이스 이름 옆에 있는 확인란을 선택하고 Edit(편집)을 선택합니다.
- 데이터베이스를 삭제하려면 데이터베이스 이름 옆에 있는 확인란을 선택하고 Remove(제거)를 선택합니다.
- 데이터베이스에 포함된 테이블 목록을 표시하려면 데이터베이스 이름을 선택합니다. 그러면 데이터베이스 속성에 데이터베이스의 모든 테이블이 표시됩니다.

크롤러가 작성하는 데이터베이스를 변경하려면 크롤러 정의를 변경해야 합니다. 자세한 내용은 크롤러를 사용하여 데이터 카탈로그 채우기 단원을 참조하십시오.

데이터베이스 리소스 링크

AWS Glue 콘솔이 최근에 업데이트되었습니다. 현재 버전의 콘솔에서는 데이터베이스 리소스 링크를 지원하지 않습니다.

Data Catalog에는 데이터베이스에 대한 리소스 링크도 포함될 수 있습니다. 데이터베이스 리소스 링크는 로컬 또는 공유 데이터베이스에 대한 링크입니다. 현재 AWS Lake Formation에서만 리소스 링크를 생성할 수 있습니다. 데이터베이스에 대한 리소스 링크를 생성한 후에는 데이터베이스 이름을 사용할 모든 위치에 리소스 링크 이름을 사용할 수 있습니다. 사용자가 소유하거나 사용자와 공유된 데이터베이스와 함께 데이터베이스 리소스 링크는 `glue:GetDatabases()`에 의해 반환되고 AWS Glue 콘솔의 [데이터베이스(Databases)] 페이지에 항목으로 나타납니다.

Data Catalog에는 테이블 리소스 링크도 포함될 수 있습니다.

리소스 링크에 대한 자세한 내용은 AWS Lake Formation Developer Guide의 Creating Resource Links를 참조하세요.

테이블 생성

크롤러를 실행하여 데이터의 목록을 데이터 스토어로 가져오는 것이 권장되는 방법이지만 메타데이터 테이블을 수동으로 AWS Glue Data Catalog에 추가할 수도 있습니다. 이 접근 방식을 사용하면 메타데이터 정의를 더 세밀하게 제어하고 특정 요구 사항에 따라 메타데이터 정의를 사용자 지정할 수 있습니다.

또한 다음과 같은 방법으로 데이터 카탈로그에 수동으로 테이블을 추가할 수 있습니다.

- AWS Glue 콘솔을 사용하여 AWS Glue Data Catalog에 테이블을 수동으로 생성합니다. 자세한 내용은 콘솔을 사용하여 테이블 생성 단원을 참조하십시오.
- AWS Glue API에서 `CreateTable` 작업을 사용하여 AWS Glue Data Catalog에 테이블을 생성합니다. 자세한 내용은 CreateTable 작업(Python: create_table) 단원을 참조하십시오.
- AWS CloudFormation 템플릿을 사용합니다. 자세한 내용은 AWS Glue용 AWS CloudFormation 단원을 참조하십시오.

콘솔 또는 API를 사용하여 테이블을 수동으로 정의할 때 데이터 원본에 있는 데이터의 유형과 형식을 나타내는 분류 필드의 값과 테이블 스키마를 지정합니다. 크롤러가 테이블을 생성하면 데이터 형식 및 스키마는 기본 제공 분류자 또는 사용자 지정 분류자에 의해 결정됩니다. AWS Glue 콘솔을 사용하여 테이블 생성에 대한 자세한 내용은 콘솔을 사용하여 테이블 생성 단원을 참조하십시오.

주제

- 테이블 파티션
- 테이블 리소스 링크
- 콘솔을 사용하여 테이블 생성

- 파티션 인덱스 생성
- 크롤러를 사용하여 수동으로 생성된 Data Catalog 테이블 업데이트
- 데이터 카탈로그 테이블 속성

테이블 파티션

Amazon Simple Storage Service(Amazon S3) 폴더의 AWS Glue 테이블 정의는 분할된 테이블을 설명할 수 있습니다. 예를 들어, 쿼리 성능을 향상시키려면 파티션된 테이블이 매월 데이터를 키로써 매월 이름을 사용하여 다른 파일로 나눠야 합니다. AWS Glue에서 테이블의 파티션된 키가 테이블 정의에 포함됩니다. AWS Glue가 Amazon S3 폴더의 데이터를 평가하여 테이블을 분류하면 개별 테이블인지 또는 추가된 파티션된 테이블인지 결정합니다.

테이블의 모든 파티션을 로드하는 대신 테이블에 파티션 인덱스를 만들어 파티션의 하위 집합을 가져올 수 있습니다. 파티션 인덱스 작업에 대한 자세한 내용은 파티션 인덱스 생성 섹션을 참조하세요.

AWS Glue가 Amazon S3 폴더의 파티션된 테이블을 생성하는 것으로 간주하려면 모든 조건이 true여야 합니다.

- AWS Glue가 결정함에 따라 파일의 스키마는 비슷합니다.
- 파일의 데이터 형식은 동일합니다.
- 파일의 압축 형식은 동일합니다.

예를 들어, iOS 및 안드로이드 앱 세일 데이터 모두를 저장할 수 있는 `my-app-bucket`이라는 Amazon S3 버킷을 소유할 수 있습니다. 데이터는 연, 월, 일별로 분할되어 있습니다. iOS 및 안드로이드 세일 데이터 파일은 동일한 스키마, 데이터 포맷 및 압축 포맷을 가지고 있습니다. AWS Glue Data Catalog에서 AWS Glue 크롤러는 연, 월, 일별 파티션 키와 함께 하나의 테이블 정의를 생성합니다.

`my-app-bucket`의 다음 Amazon S3 목록은 몇 가지 파티션을 보여줍니다. = 부호는 파티션 키 값을 지정할 때 사용됩니다.

```
my-app-bucket/Sales/year=2010/month=feb/day=1/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=1/Android.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/Android.csv
...
my-app-bucket/Sales/year=2017/month=feb/day=4/iOS.csv
my-app-bucket/Sales/year=2017/month=feb/day=4/Android.csv
```

테이블 리소스 링크

AWS Glue 콘솔이 최근에 업데이트되었습니다. 현재 버전의 콘솔에서는 테이블 리소스 링크를 지원하지 않습니다.

Data Catalog에는 테이블에 대한 리소스 링크도 포함될 수 있습니다. 테이블 리소스 링크는 로컬 또는 공유 테이블에 대한 링크입니다. 현재 AWS Lake Formation에서만 리소스 링크를 생성할 수 있습니다. 테이블에 대한 리소스 링크를 생성한 후에는 테이블 이름을 사용할 모든 위치에 리소스 링크 이름을 사용할 수 있습니다. 사용자가 소유하거나 사용자와 공유된 테이블과 함께 테이블 리소스 링크는 `glue:GetTables()`에 의해 반환되고 AWS Glue 콘솔의 [데이터베이스(Databases)] 페이지에 항목으로 나타납니다.

Data Catalog에는 데이터베이스 리소스 링크도 포함될 수 있습니다.

리소스 링크에 대한 자세한 내용은 AWS Lake Formation Developer Guide의 Creating Resource Links를 참조하세요.

콘솔을 사용하여 테이블 생성

AWS Glue Data Catalog에서 테이블은 데이터 스토어의 데이터를 표현하는 메타데이터 정의입니다. 크롤러를 실행하여 테이블을 생성하거나 AWS Glue 콘솔을 사용하여 수동으로 테이블을 생성할 수 있습니다. AWS Glue 콘솔의 [테이블(Tables)] 목록에는 테이블의 메타데이터 값이 표시됩니다. ETL(Extract, Transform, and Load) 작업을 생성할 시 테이블 정의를 사용하여 소스 및 타겟을 명시합니다.

> **Note**
>
> AWS Management Console이 최근에 변경되었으므로 `SearchTables` 권한을 가지려면 기존 IAM 역할을 수정해야 할 수 있습니다. 새 역할 생성을 위해 `SearchTables` API 권한이 이미 기본값으로 추가되었습니다.

시작하려면 AWS Management Console에 로그인하고 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다. [Tables] 탭을 선택하고 [Add tables] 버튼을 사용하여 크롤러 혹은 수동 입력 속성으로 테이블을 생성합니다.

콘솔에서 테이블 추가

크롤러를 사용하여 테이블을 추가하려면 [Add tables]에서 [Add tables using a crawler]를 선택합니다. 다음, [Add crawler] 마법사 지시에 따릅니다. 크롤러가 실행되면 테이블은 AWS Glue Data Catalog에 추가됩니다. 자세한 내용은 크롤러를 사용하여 데이터 카탈로그 채우기 단원을 참조하십시오.

요구되는 속성을 알면 테이블 마법사를 통해 Data Catalog에 Amazon Simple Storage Service(Amazon S3) 테이블 정의를 생성할 수 있습니다. [Add tables]에서 [Add table manually]를 선택하고 [Add table] 마법사 지시를 따릅니다.

수동으로 콘솔을 통해 테이블을 추가할 때는 다음을 참조하세요.

- Amazon Athena에서 테이블에 액세스하고자 할 경우, 공급자 이름을 영숫자 문자와 밑줄 표시로 생성합니다. 자세한 내용은 Athena 이름을 참조하세요.
- 원본 데이터의 위치는 Amazon S3 경로여야 합니다.
- 데이터의 데이터 형식은 마법사의 형식과 일치해야 합니다. 해당 분류와 SerDe, 기타 테이블 속성은 자동으로 선택한 형식을 기반으로 장착됩니다. 다음 형식을 사용하여 테이블을 정의할 수 있습니다.

  Avro

  아파치 아브로 JSON 이진 형식

  CSV

  값 분리 가치. 콤마 혹은 파이프, 세미콜론, 탭, Ctrl+A의 구획 문자를 명시합니다.

  JSON

  JavaScript Object Notation(JSON)입니다.

  XML

  확장형 마크업 언어 형식 데이터에서 열을 정의하는 XML 태그를 명시합니다. 컬럼은 열 태그들 내에서 정의합니다.

  PARQUET

  Apache Parquet 컬럼 방식 스토리지

  ORC

  Optimized Row Columnar(ORC) 파일 형식입니다. Hive 데이터를 효율적으로 저장하도록 설계된 형식입니다.
- 테이블의 파티션 키를 정의할 수 있습니다.

- 현재, 콘솔을 사용하여 생성한 파티션된 테이블은 ETL 작업 시 사용할 수 없습니다.

테이블 속성

다음은 테이블이 갖는 몇 가지 중요한 특징입니다.

명칭

테이블 생성 시 이름이 결정되고 변경할 수 없습니다. 테이블 이름은 많은 AWS Glue 작업을 참조하십시오.

데이터베이스

컨테이너 객체는 테이블이 위치한 곳입니다. 이 객체는 AWS Glue Data Catalog 내 존재하는 테이블 조직을 포함하고 데이터 스토어의 조직과 다를 수 있습니다. 데이터베이스를 삭제하면 데이터베이스에 포함된 모든 테이블도 Data Catalog에서 삭제됩니다.

설명

테이블에 대한 설명입니다. 테이블 내용을 이해할 수 있도록 설명을 적을 수 있습니다.

테이블 형식

표준 AWS Glue 테이블 또는 Apache Iceberg 형식의 테이블 생성을 지정합니다.

Data Catalog에서는 Iceberg 테이블의 쿼리 성능을 개선하고 테이블 스토리지를 관리하는 다음 테이블 최적화 옵션을 제공합니다.

- 압축 - 데이터 파일이 병합 및 재작성되어 불필요한 데이터를 제거하고 조각난 데이터를 더 크고 효율적인 파일로 통합합니다.
- 스냅샷 보존 - 스냅샷은 Iceberg 테이블의 타임스탬프가 표시된 버전입니다. 스냅샷 보존 구성을 통해 고객은 스냅샷을 보존하는 기간과 보존할 스냅샷 수를 적용할 수 있습니다. 스냅샷 보존 최적화 프로그램을 구성하면 오래되고 불필요한 스냅샷과 연결된 파일을 제거하여 스토리지 오버헤드를 관리하는 데 도움이 될 수 있습니다.
- 분리된 파일 삭제 - 분리된 파일은 Iceberg 테이블 메타데이터에서 더 이상 참조되지 않는 파일입니다. 이러한 파일은 시간이 지남에 따라 누적될 수 있으며, 특히 테이블 삭제 같은 작업이나 ETL 작업 실패 이후에 누적될 수 있습니다. 분리된 파일 삭제를 활성화하면 AWS Glue에서 이러한 불필요한 파일을 주기적으로 식별 및 제거하여 스토리지를 확보할 수 있습니다.

자세한 내용은 Iceberg 테이블 최적화 단원을 참조하십시오.

최적화 구성

기본 설정을 사용하거나 테이블 최적화 프로그램을 활성화하기 위한 설정을 사용자 지정할 수 있습니다.

IAM 역할

테이블 최적화 프로그램을 실행하기 위해 서비스는 사용자를 대신하여 IAM 역할을 수임합니다. 드롭다운을 사용하여 IAM 역할을 선택할 수 있습니다. 압축 기능을 활성화하는 데 필요한 권한이 역할에 있는지 확인합니다.

IAM 역할에 필요한 권한에 대해 알아보려면 테이블 최적화 필수 조건 섹션을 참조하십시오.

위치

이 테이블 정의는 데이터 스토어의 데이터 위치를 표현하는 포인터를 표시합니다.

분류

테이블 생성 시 분류 값이 제공됩니다. 일반적으로 크롤러가 소스 데이터 형식을 실행하고 명시하면 이것이 적힙니다.

최종 업데이트 날짜

Data Catalog에 이 테이블이 업데이트된 시간 및 날짜(UTC).

데이터 추가됨

Data Catalog에 이 테이블이 추가된 시간 및 날짜(UTC).

Deprecated

기존 데이터 스토어에 Data Catalog의 테이블이 존재하지 않고 AWS Glue가 이를 인식하면 데이터 카탈로그에 이 테이블을 사용 중단 상태로 표시합니다. 사용 중단된 테이블을 참조하는 작업을 실행하면 그 작업은 실행되지 않을 수 있습니다. 사용 중단된 테이블을 참조하는 작업을 편집하여 소스와 타켓으로써 제거합니다. 더 이상 필요하지 않은 사용 중단된 테이블은 삭제하는 것이 좋습니다.

연결

AWS Glue가 데이터 스토어에 연결하고자 할 경우, 연결 이름은 테이블과 관련이 있습니다.

테이블 세부 정보 보기 및 편집

목록에서 테이블 이름을 선택하고 [Action, View details]를 선택하여 존재하는 테이블의 세부 정보를 열람합니다.

테이블 세부 정보는 테이블 속성과 스키마를 포함합니다. 이 화면은 테이블을 정의하기 위한 열 이름, 데이터 유형, 파티션 키 열 등을 포함한 테이블 스키마를 표시합니다. 복잡한 유형의 열은 [속성 보기(View properties)]를 선택하여 다음 예처럼 필드 구조의 세부 사항을 표시합니다.

```
{
"StorageDescriptor":
    {
      "cols": {
         "FieldSchema": [
           {
             "name": "primary-1",
             "type": "CHAR",
             "comment": ""
           },
           {
             "name": "second ",
             "type": "STRING",
             "comment": ""
           }
         ]
      },
      "location": "s3://aws-logs-111122223333-us-east-1",
      "inputFormat": "",
      "outputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
      "compressed": "false",
      "numBuckets": "0",
      "SerDeInfo": {
          "name": "",
          "serializationLib": "org.apache.hadoop.hive.serde2.OpenCSVSerde",
          "parameters": {
              "separatorChar": "|"
           }
      },
      "bucketCols": [],
      "sortCols": [],
      "parameters": {},
      "SkewedInfo": {},
      "storedAsSubDirectories": "false"
    },
    "parameters": {
       "classification": "csv"
    }
}
```

이러한 속성에 대한 `StorageDescriptor`와 같은 자세한 내용은 StorageDescriptor 구조를 참조하십시오.

[Edit schema]를 선택하여 열을 추가 및 제거하고 열 이름과 데이터 유형을 바꿔 테이블 스키마를 바꿀 수 있습니다.

[Compare versions]를 선택하고 테이블 스키마의 두 가지 다른 버전을 나란히 비교하여 스키마를 포함한 테이블의 다른 유형들을 비교할 수 있습니다. 자세한 내용은 테이블 스키마 버전 비교 단원을 참조하십시오.

Amazon S3 파티션을 구성하는 파일을 표시하려면 [파티션 보기(View partition)]를 선택합니다. Amazon S3 테이블의 [키(Key)] 열은 원본 데이터 스토어 테이블의 파티션에 사용된 파티션 키를 표시합니다. 파티셔닝은 날짜, 위치, 출발점과 같이 키 열 값을 기준으로 테이블을 관련 부분으로 나눕니다. 파티션에 대한 보다 자세한 내용은 "hive partitioning"으로 웹서치를 하시기 바랍니다.

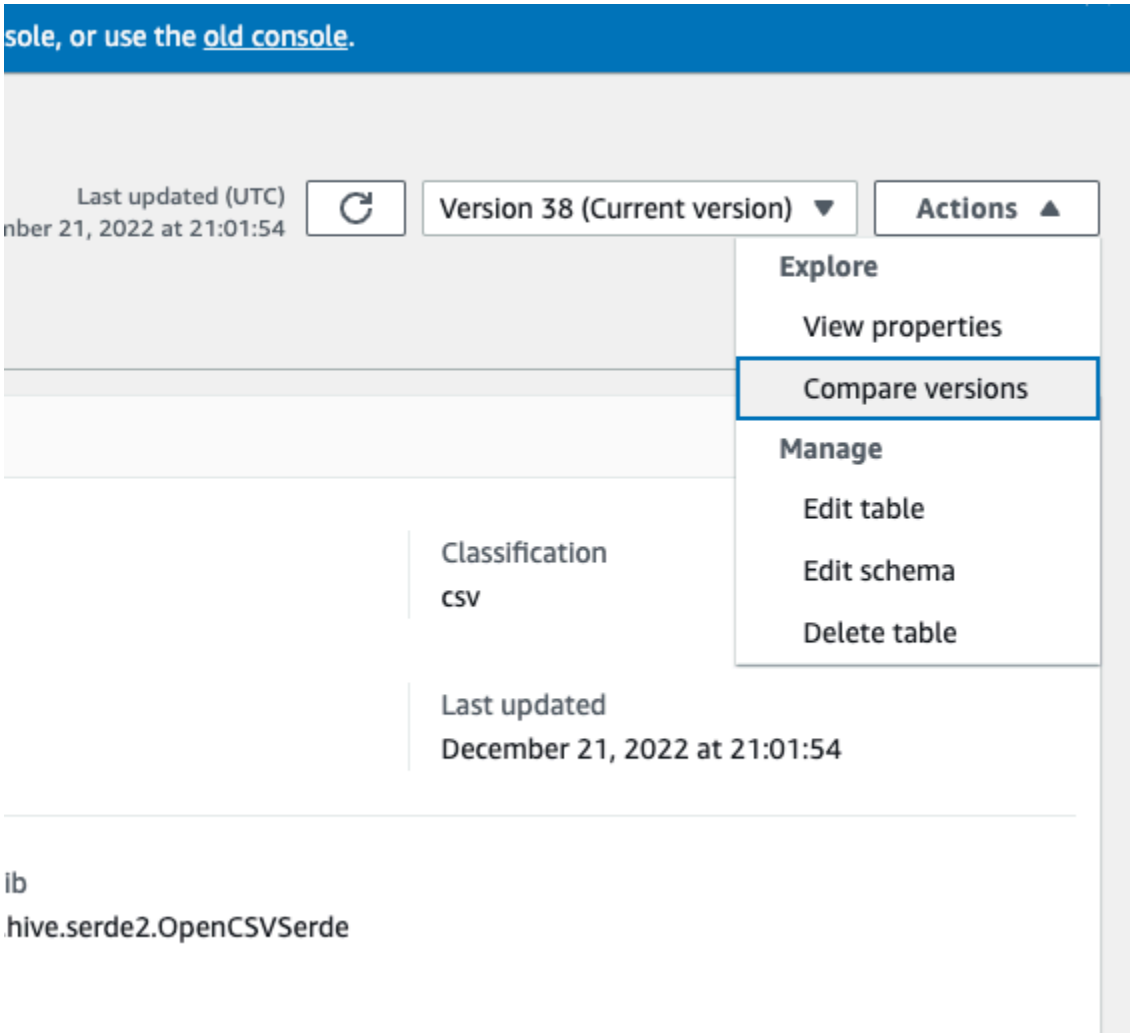
> **Note**
>
> 콘솔의 [Explore table] 자습서를 통해 테이블 세부 정보를 보기 위한 단계들을 알아보십시오.

테이블 스키마 버전 비교

두 버전의 테이블 스키마를 비교할 때 중첩된 행을 확장 및 축소하여 중첩된 행 변경 내용을 비교하고, 두 버전의 스키마를 나란히 비교하고, 테이블 속성을 나란히 볼 수 있습니다.

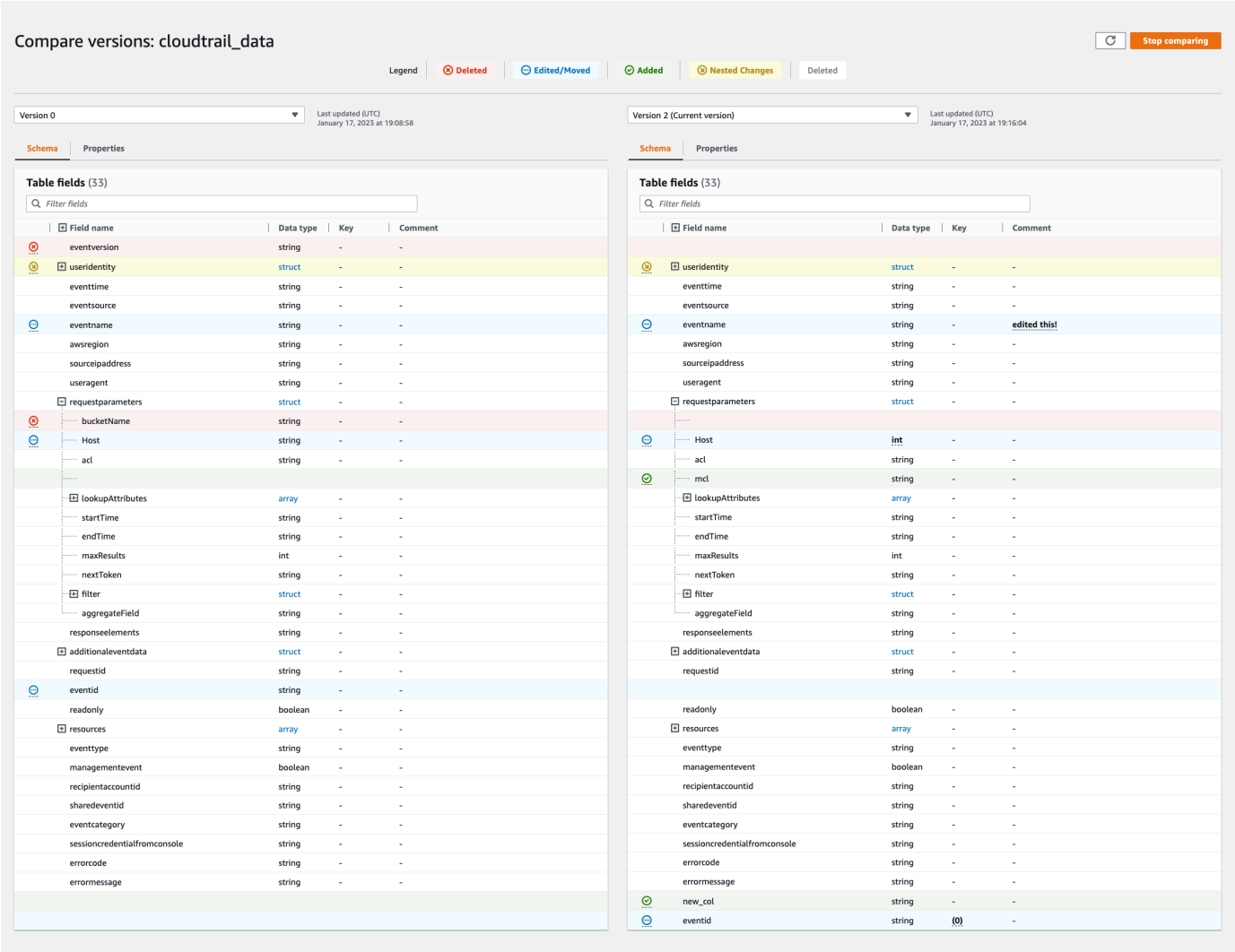
버전을 비교하려면

1. AWS Glue Console에서 테이블, 작업을 선택한 다음 버전 비교를 선택합니다.



2. 버전 드롭다운 메뉴를 선택하여 비교할 버전을 선택합니다. 스키마를 비교할 때 스키마 탭은 주황색으로 강조 표시됩니다.
3. 두 버전 간에 테이블을 비교하면 화면 왼쪽과 오른쪽에 테이블 스키마가 표시됩니다. 이렇게 하면 열 이름, 데이터 유형, 키 및 설명 필드를 나란히 비교하여 변경 내용을 시각적으로 확인할 수 있습니다. 변경 사항이 있을 경우 색상 아이콘에 적용된 변경 유형이 표시됩니다.

   - 삭제됨 - 빨간색 아이콘으로 표시되면 이전 버전의 테이블 스키마에서 해당 열이 제거된 위치를 나타냅니다.
   - 편집됨 또는 이동됨 - 파란색 아이콘으로 표시되면 새 버전의 테이블 스키마에서 열이 수정 또는 이동된 위치를 나타냅니다.
   - 추가됨 - 녹색 아이콘으로 표시되면 해당 열이 새 버전의 테이블 스키마에 추가된 위치를 나타냅니다.
   - 중첩된 변경 내용 - 노란색 아이콘으로 표시되면 중첩된 열에 변경 내용이 포함된 위치를 나타냅니다. 확장할 열을 선택하면 삭제, 편집, 이동 또는 추가된 열이 표시됩니다.



4. 필터 필드 검색 창을 사용하면 여기에 입력한 문자를 기반으로 필드를 표시할 수 있습니다. 테이블 버전 중 하나에 열 이름을 입력하면 필터링된 필드가 두 테이블 버전 모두에 표시되어 변경 사항이 발생한 위치를 보여 줍니다.
5. 속성을 비교하려면 속성 탭을 선택합니다.
6. 버전 비교를 중지하려면 비교 중지를 선택하여 테이블 목록으로 돌아갑니다.

파티션 인덱스 생성

시간이 지남에 따라 수십만 개의 파티션이 테이블에 추가됩니다. GetPartitions API는 테이블의 파티션을 가져오는 데 사용됩니다. API는 요청에 제공된 표현식과 일치하는 파티션을 반환합니다.

Country, Category, Year, Month, creationDate 키로 분할된 sales_data 테이블을 예로 들어 보겠습니다. 2020년 2020-08-15 이후에 도서 카테고리에서 판매된 모든 상품에 대한 판매 데이터를

얻으려면 데이터 카탈로그에 "Category = 'Books' 및 creationDate > '2020-08-15'"라는 표현으로 `GetPartitions` 요청을 해야 합니다.

테이블에 파티션 인덱스가 없으면 AWS Glue는 테이블의 모든 파티션을 로드한 다음 `GetPartitions` 요청에서 사용자가 제공한 쿼리 표현식을 사용하여 로드된 파티션을 필터링합니다. 인덱스가 없는 테이블에서 파티션 수가 증가하면 쿼리를 실행하는 데 더 많은 시간이 걸립니다. 인덱스를 사용하면 `GetPartitions` 쿼리는 테이블의 모든 파티션을 로드하는 대신 파티션의 하위 집합을 가져오려고 시도합니다.

주제

- 파티션 인덱스 정보
- 파티션 인덱스로 테이블 생성
- 기존 테이블에 파티션 인덱스 추가
- 테이블의 파티션 인덱스 설명
- 파티션 인덱스 사용에 대한 제한 사항
- 최적화된 GetPartitions 호출에 인덱스 사용
- 엔진과의 통합

파티션 인덱스 정보

파티션 인덱스를 생성할 때 지정된 테이블에 이미 존재하는 파티션 키 목록을 지정합니다. 파티션 인덱스는 테이블에 정의된 파티션 키의 하위 목록입니다. 테이블에 정의된 파티션 키의 순열에 대해 파티션 인덱스를 생성할 수 있습니다. 위의 sales_data 테이블에 대해 가능한 인덱스는 (country, category, creationDate), (country, category, year), (country, category), (country), (category, country, year, month) 등입니다.

Data Catalog는 인덱스 생성 시 제공된 순서대로 파티션 값을 연결합니다. 인덱스는 파티션이 테이블에 추가될 때 일관되게 구축됩니다. 문자열(string, char, varchar), 숫자(int, bigint, long, tinyint, smallint), 날짜(yyyy-MM-dd) 열 유형에 대한 인덱스를 생성할 수 있습니다.

지원되는 데이터 유형

- 날짜 - ISO 형식의 날짜(예: `YYYY-MM-DD`)입니다. 예를 들어 날짜 `2020-08-15`입니다. 형식은 하이픈(-)을 사용하여 연도, 월, 일을 구분합니다. 인덱싱에 허용되는 날짜 범위는 `0000-01-01`에서 `9999-12-31`까지입니다.
- String - 작은따옴표 또는 큰따옴표로 묶인 문자열 리터럴입니다.
- Char - 길이가 1~255자로 지정된 고정 길이 문자 데이터입니다(예: char(10)).

- Varchar - 길이가 1~65535자로 지정된 가변 길이 문자 데이터입니다(예: varchar(10)).
- 숫자 - int, bigint, long, tinyint, smallint

숫자, 문자열, 날짜 데이터 유형에 대한 인덱스는 =, >, >=, <, <=, between 연산자를 지원합니다. 인덱싱 솔루션은 현재 AND 논리 연산자만 지원합니다. "LIKE", "IN", "OR" 및 "NOT" 연산자가 있는 하위 표현식은 인덱스를 사용하여 필터링하는 표현식에서 무시됩니다. 무시된 하위 표현식에 대한 필터링은 인덱스 필터링을 적용한 후 가져온 파티션에서 수행됩니다.

테이블에 추가된 각 파티션에 대해 해당 인덱스 항목이 생성됩니다. 'n'개의 파티션이 있는 테이블의 경우 1개의 파티션 인덱스는 'n'개의 파티션 인덱스 항목을 생성합니다. 동일한 테이블의 'm' 파티션 인덱스는 'm*n' 파티션 인덱스 항목이 됩니다. 각 파티션 인덱스 항목은 데이터 카탈로그 스토리지에 대한 현재 AWS Glue 가격 정책에 따라 요금이 부과됩니다. 스토리지 객체 요금에 대한 자세한 내용은 AWS Glue 요금을 참조하세요.

파티션 인덱스로 테이블 생성

테이블 생성 중 파티션 인덱스를 생성할 수 있습니다. `CreateTable` 요청은 `PartitionIndex` 객체 목록을 입력으로 사용합니다. 지정된 테이블에 최대 3개의 파티션 인덱스를 생성할 수 있습니다. 각 파티션 인덱스에는 테이블에 대해 정의된 이름과 `partitionKeys` 목록이 필요합니다. 테이블에 생성된 인덱스는 `GetPartitionIndexes API`를 사용하여 가져올 수 있습니다.

기존 테이블에 파티션 인덱스 추가

파티션 인덱스를 기존 테이블에 추가하려면 `CreatePartitionIndex` 작업을 사용합니다. 단, `CreatePartitionIndex` 작업당 `PartitionIndex` 하나를 생성할 수 있습니다. 인덱스가 생성되는 동안 테이블을 계속 사용할 수 있으므로 인덱스를 추가해도 테이블의 가용성에는 영향이 없습니다.

추가된 파티션의 인덱스 상태는 CREATING으로 설정되고 인덱스 데이터 생성이 시작됩니다. 인덱스 생성 프로세스가 성공하면 indexStatus가 ACTIVE로 업데이트되고 실패한 프로세스에 대해 인덱스 상태가 FAILED로 업데이트됩니다. 인덱스 생성은 여러 가지 이유로 실패할 수 있으며 `GetPartitionIndexes` 작업을 사용하여 실패 세부 정보를 검색할 수 있습니다. 가능한 실패는 다음과 같습니다.

- ENCRYPTED_PARTITION_ERROR - 암호화된 파티션이 있는 테이블에 대한 인덱스 생성은 지원되지 않습니다.
- INVALID PARTITION TYPE DATA_ERROR - `partitionKey` 값이 해당 `partitionKey` 데이터 유형에 대해 유효한 값이 아닐 때 관찰됩니다. 예: 'int' 데이터 유형을 가진 `partitionKey`는 'foo' 값을 갖습니다.

- MISSING PARTITION VALUE_ERROR - indexedKey에 대한 partitionValue가 없을 때 관찰됩니다. 이는 테이블이 일관되게 분할되지 않은 경우에 발생할 수 있습니다.
- UNSUPPORTED_PARTITION_CHARACTER_ERROR - 인덱싱된 파티션 키의 값에 \u0000, \u0001 또는 \u0002 문자가 포함될 때 관찰됩니다.
- INTERNAL_ERROR - 인덱스를 생성하는 동안 내부 오류가 발생했습니다.

테이블의 파티션 인덱스 설명

테이블에 생성된 파티션 인덱스를 가져오려면 GetPartitionIndexes 연산을 사용합니다. 응답은 현재 각 인덱스 상태(IndexStatus)와 함께 테이블의 모든 인덱스를 반환합니다.

파티션 인덱스의 IndexStatus는 다음 중 하나가 됩니다.

- CREATING - 현재 인덱스를 생성 중이며, 아직 사용할 수 없습니다.
- ACTIVE - 인덱스를 사용할 준비가 되었습니다. 요청은 인덱스를 사용하여 최적화된 쿼리를 수행할 수 있습니다.
- DELETING - 현재 인덱스를 삭제하고 있으며 더 이상 사용할 수 없습니다. 활성 상태의 인덱스는 상태를 ACTIVE에서 DELETING으로 이동하는 DeletePartitionIndex 요청을 사용하여 삭제할 수 있습니다.
- FAILED - 기존 테이블에 대한 인덱스 생성에 실패했습니다. 각 테이블은 마지막 10개의 실패한 인덱스를 저장합니다.

기존 테이블에 생성된 인덱스의 가능한 상태 전환은 다음과 같습니다.

- CREATING → ACTIVE → DELETING
- CREATING → FAILED

파티션 인덱스 사용에 대한 제한 사항

파티션 인덱스를 생성한 후에는 테이블 및 파티션 기능에 대한 다음 변경 사항을 확인합니다.

새 파티션 생성(인덱스 추가 후)

테이블에 파티션 인덱스가 생성된 후 테이블에 추가된 모든 새 파티션은 인덱싱된 키에 대한 데이터 유형 검사에 대해 검증됩니다. 인덱싱된 키의 파티션 값은 데이터 유형 포맷에 대해 검증됩니다. 데이터 유형 검사가 실패하면 파티션 생성 작업이 실패합니다. sales_data 테이블의 경우 범주가 string 유

형이고 연도가 `int` 유형인 키(category, year)에 대한 인덱스가 생성되면 YEAR 값이 "foo"인 새 파티션 생성이 실패합니다.

인덱스가 사용되면 U+0000, U+00001 및 U+0002 문자가 있는 인덱스 키 값이 있는 파티션 추가가 실패하기 시작합니다.

테이블 업데이트

테이블에 파티션 인덱스가 생성되면 기존 파티션 키의 파티션 키 이름을 수정할 수 없으며 인덱스에 등록된 키의 유형이나 순서를 변경할 수 없습니다.

최적화된 GetPartitions 호출에 인덱스 사용

인덱스가 있는 테이블에서 `GetPartitions`를 호출할 때 표현식을 포함할 수 있으며 해당하는 경우 Data Catalog는 가능한 경우 인덱스를 사용합니다. 인덱스의 첫 번째 키는 필터링에 사용할 인덱스에 대한 표현식에 전달되어야 합니다. 필터링의 인덱스 최적화는 최선의 노력으로 적용됩니다. Data Catalog는 가능한 한 인덱스 최적화를 사용하려고 시도하지만 인덱스가 누락되거나 지원되지 않는 연산자의 경우 모든 파티션을 로드하는 기존 구현으로 폴백합니다.

위의 sales_data 테이블에 대해 인덱스 [Country, Category, Year]를 추가합니다. 표현식에 "Country"가 전달되지 않으면 등록된 인덱스가 인덱스를 사용하여 파티션을 필터링할 수 없습니다. 다양한 쿼리 패턴을 지원하기 위해 최대 3개의 인덱스를 추가할 수 있습니다.

몇 가지 예제 표현식을 사용하여 인덱스가 어떻게 작동하는지 살펴보겠습니다.

Expressions	인덱스 사용 방법
Country = 'US'	파티션을 필터링하는 데 인덱스가 사용됩니다.
Country = 'US' 및 Category = 'Shoes'	파티션을 필터링하는 데 인덱스가 사용됩니다.
Category = 'Shoes'	표현식에 "country"가 제공되지 않으므로 인덱스는 사용되지 않습니다. 응답을 반환하기 위해 모든 파티션이 로드됩니다.
Country = 'US', Category = 'Shoes' 및 Year > '2018'	파티션을 필터링하는 데 인덱스가 사용됩니다.
Country = 'US', Category = 'Shoes', Year > '2018' 및 month = 2	country = "US", category = "shoes", year > 2018인 모든 파티션을 가져오는 데 인덱스가 사

Expressions	인덱스 사용 방법
	용됩니다. 그런 다음 월 표현식에 대한 필터링이 수행됩니다.
Country = 'US' AND Category = 'Shoes' OR Year > '2018'	표현식에 OR 연산자가 있으므로 인덱스가 사용되지 않습니다.
Country = 'US' AND Category = 'Shoes' AND (Year = 2017 OR Year = '2018')	country = "US" 및 category = "shoes"인 모든 파티션을 가져오는 데 인덱스가 사용된 다음 연도 표현식에 대한 필터링이 수행됩니다.
Country in ('US', 'UK') AND Category = 'Shoes'	IN 연산자는 현재 지원되지 않으므로 필터링에 인덱스가 사용되지 않습니다.
Country = 'US' AND Category in ('Shoes', 'Books')	country = "US"인 모든 파티션을 가져오는 데 인덱스가 사용된 다음 범주 표현식에 대한 필터링이 수행됩니다.
Country = 'US' AND Category in ('Shoes', 'Books') AND (creationDate > '2023-9-01'	country = "US" 및 creationDate > '2023-9-01'인 모든 파티션을 가져오는 데 인덱스가 사용된 다음 범주 표현식에 대한 필터링이 수행됩니다.

엔진과의 통합

Redshift Spectrum, Amazon EMR 및 AWS Glue ETL Spark DataFrames는 인덱스가 AWS Glue에서 ACTIVE 상태인 후 파티션을 가져 오기 위해 인덱스를 사용할 수 있습니다. Athena 및 AWS Glue ETL 동적 프레임에서 쿼리 개선을 위해 인덱스를 활용하려면 추가 단계를 따라야 합니다.

파티션 필터링 사용 설정

Athena에서 파티션 필터링을 사용하려면 다음과 같이 테이블 속성을 업데이트해야 합니다.

1. AWS Glue 콘솔의 데이터 카탈로그에서 테이블을 선택합니다.
2. 테이블을 선택합니다.
3. 작업에서 테이블 편집을 선택합니다.
4. 테이블 속성에서 다음을 추가합니다.
   - 키 - `partition_filtering.enabled`

- 값 - `true`

5. 적용을 선택합니다.

또는 Athena에서 ALTER TABLE SET PROPERTIES 쿼리를 실행하여 이 파라미터를 설정할 수도 있습니다.

```
ALTER TABLE partition_index.table_with_index
SET TBLPROPERTIES ('partition_filtering.enabled' = 'true')
```

크롤러를 사용하여 수동으로 생성된 Data Catalog 테이블 업데이트

AWS Glue Data Catalog 테이블을 수동으로 생성한 다음 AWS Glue 크롤러를 사용하여 업데이트된 상태로 유지할 수 있습니다. 일정에 따라 실행되는 크롤러는 새 파티션을 추가하고 스키마 변경 내용으로 테이블을 업데이트할 수 있습니다. 이는 Apache Hive 메타스토어에서 마이그레이션된 테이블에도 적용됩니다.

이렇게 하려면 크롤러를 정의할 때 하나 이상의 데이터 스토어를 크롤의 원본으로 지정하는 대신 하나 이상의 기존 Data Catalog 테이블을 지정합니다. 그런 다음 크롤러는 카탈로그 테이블에 지정된 데이터 스토어를 크롤합니다. 이 경우 새 테이블이 생성되지 않습니다. 대신 수동으로 생성된 테이블이 업데이트됩니다.

카탈로그 테이블을 수동으로 생성하고 카탈로그 테이블을 크롤러 원본으로 지정할 수 있는 다른 이유는 다음과 같습니다.

- 카탈로그 테이블 이름을 선택하고 카탈로그 테이블 이름 지정 알고리즘을 사용하지 않습니다.
- 파티션 감지를 방해할 수 있는 형식의 파일이 데이터 원본 경로에 잘못 저장되는 경우 새 테이블이 생성되지 않도록 합니다.

자세한 내용은 2단계: 데이터 소스 및 분류자 선택 단원을 참조하십시오.

데이터 카탈로그 테이블 속성

AWS CLI에 알려진 테이블 속성 또는 매개 변수는 검증되지 않은 키 및 값 문자열입니다. 테이블에서 사용자 고유의 속성을 설정하여 AWS Glue 외부에서 데이터 카탈로그를 사용할 수 있도록 지원할 수 있습니다. 데이터 카탈로그를 사용하는 다른 서비스도 마찬가지일 수 있습니다. AWS Glue는 작업 또는 크롤러를 실행할 때 일부 테이블 속성을 설정합니다. 달리 설명하지 않는 한 이러한 속성은 내부용

이므로 현재 형태로 계속 존재하도록 지원하지 않으며, 이러한 속성을 수동으로 변경하는 경우 제품 작동을 지원하지 않습니다.

AWS Glue 크롤러가 설정하는 테이블 속성에 대한 자세한 내용은 the section called “크롤러가 데이터 카탈로그 테이블에 설정한 파라미터”을 참조하세요.

다른 AWS 서비스와 통합

AWS Glue 크롤러를 사용하여 AWS Glue Data Catalog를 채울 수 있지만 데이터 카탈로그와 자동으로 통합되어 채울 수 있는 여러 AWS 서비스가 있습니다. 다음 섹션에서는 AWS 서비스가 지원하며 데이터 카탈로그를 채울 수 있는 특정 사용 사례에 대한 자세한 정보를 제공합니다.

주제

- AWS Lake Formation
- Amazon Athena

AWS Lake Formation

AWS Lake Formation은 AWS에서 안전한 데이터 레이크를 더 쉽게 설정할 수 있게 해주는 서비스입니다. Lake Formation은 AWS Glue에 기반하며, Lake Formation과 AWS Glue는 동일한 AWS Glue Data Catalog를 공유합니다. Amazon S3 데이터 위치를 Lake Formation에 등록하고, Lake Formation 콘솔을 사용하여 AWS Glue 데이터 카탈로그에 데이터베이스 및 테이블을 생성하고, 데이터 액세스 정책을 정의하고, 중앙에서 데이터 레이크 전체의 데이터 액세스를 감사할 수 있습니다. Lake Formation의 세분화된 액세스 제어를 사용하여 기존 데이터 카탈로그 리소스와 Amazon S3 데이터 위치를 관리할 수 있습니다.

Lake Formation에 데이터를 등록하면 IAM 보안 주체, AWS 계정, AWS 조직 및 조직 단위 간에서 데이터 카탈로그 리소스를 안전하게 공유할 수 있습니다.

Lake Formation을 사용하여 데이터 카탈로그 리소스를 생성하는 방법에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 데이터 카탈로그 테이블 및 데이터베이스 생성을 참조하세요.

Amazon Athena

Amazon Athena는 데이터 카탈로그를 사용하여 AWS 계정의 Amazon S3 데이터에 대한 테이블 메타데이터를 저장하고 검색합니다. Athena 쿼리 엔진은 데이블 메타데이터를 통해 쿼리하려는 데이터를 찾고, 읽고, 처리할 방법을 파악합니다.

Athena CREATE TABLE 문을 직접 사용하여 AWS Glue Data Catalog를 채울 수 있습니다. 크롤러를 실행할 필요 없이 데이터 카탈로그에서 스키마 및 파티션 메타데이터를 수동으로 정의하고 채울 수 있습니다.

1. Athena 콘솔에서 데이터 카탈로그에 테이블 메타데이터를 저장할 데이터베이스를 생성합니다.
2. CREATE EXTERNAL TABLE 문을 사용하여 데이터 소스의 스키마를 정의합니다.
3. 데이터가 분할된 경우 PARTITIONED BY 절을 사용하여 파티션 키를 정의합니다.
4. LOCATION 절을 사용하여 실제 데이터 파일이 저장되는 Amazon S3 경로를 지정합니다.
5. CREATE TABLE 문을 실행하세요.

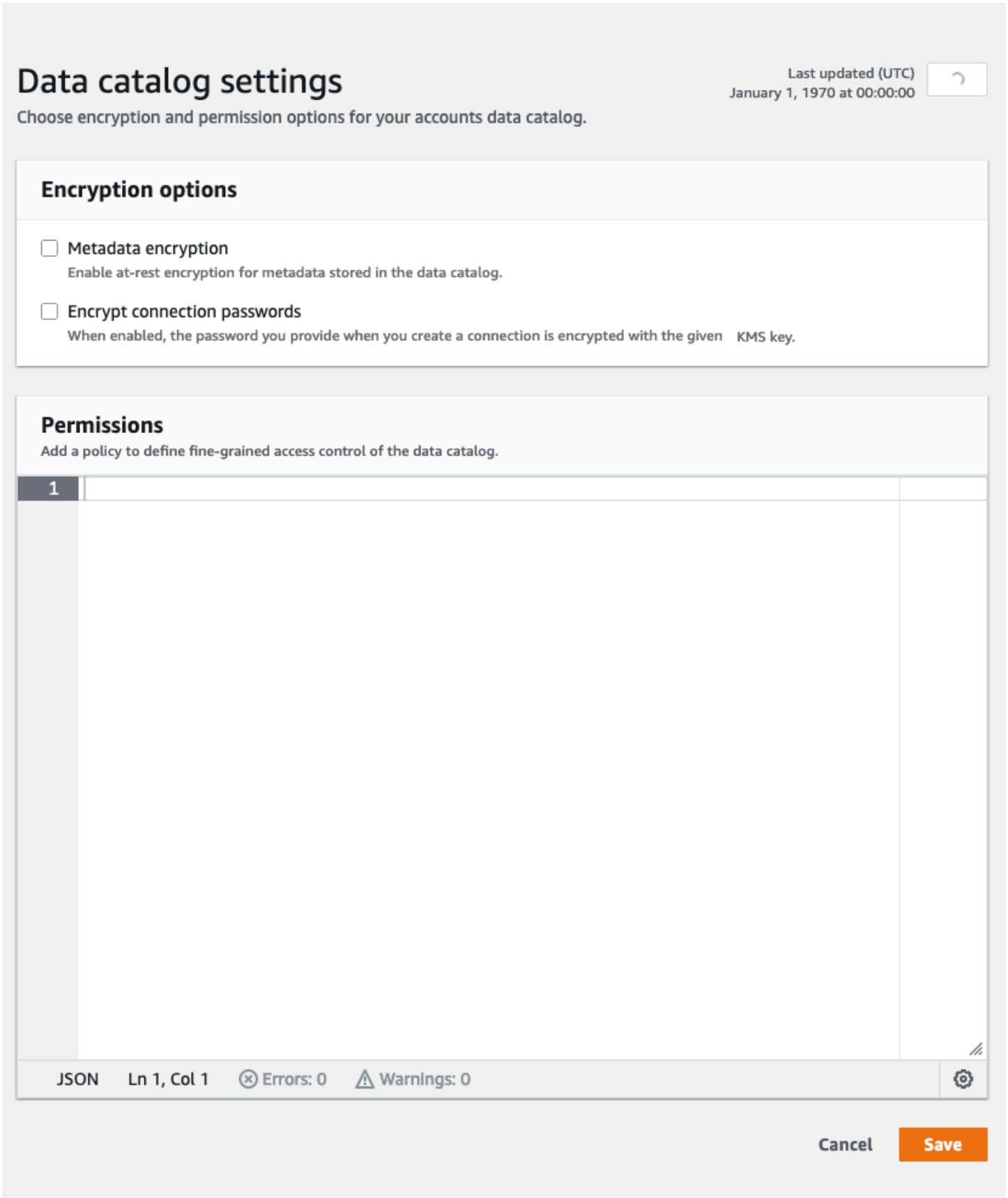
   이 쿼리는 데이터를 실제로 크롤링하지 않고 정의된 스키마와 파티션을 기반으로 데이터 카탈로그에 테이블 메타데이터를 생성합니다.

이제 Athena에서 이 테이블을 쿼리하면 데이터 카탈로그의 메타데이터를 사용하여 Amazon S3의 데이터 파일에 액세스하고 쿼리합니다.

자세한 내용은 Amazon Athena 사용 설명서의 데이터베이스 및 테이블 생성을 참조하세요.

데이터 카탈로그 설정

데이터 카탈로그 설정에는 계정의 데이터 카탈로그에 대한 암호화 및 사용 권한 옵션을 설정하는 옵션이 있습니다.


Data catalog settings
Choose encryption and permission options for your accounts data catalog.
Last updated (UTC)
January 1, 1970 at 00:00:00
Encryption options
Metadata encryption
Enable at-rest encryption for metadata stored in the data catalog.
Encrypt connection passwords
When enabled, the password you provide when you create a connection is encrypted with the given KMS key.
Permissions
Add a policy to define fine-grained access control of the data catalog.
1
JSON
Ln 1, Col 1
Errors: 0
Warnings: 0
Cancel
Save

Data Catalog의 세분화된 액세스 제어를 변경하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 암호화 옵션을 선택합니다.

   - 메타데이터 암호화 - 이 확인란을 선택하여 데이터 카탈로그의 메타데이터를 암호화합니다. 메타데이터는 지정된 AWS Key Management Service(AWS KMS) 키를 사용하여 암호화됩니다. 자세한 내용은 데이터 카탈로그 암호화 단원을 참조하십시오.
   - 연결 암호 암호화 - 이 확인란을 선택하여 연결이 생성되거나 업데이트될 때 AWS Glue 연결 객체의 암호를 암호화할 수 있습니다. 암호는 지정한 AWS KMS 키를 사용하여 암호화됩니다. 암호가 반환될 때 암호화됩니다. 이 옵션은 Data Catalog의 모든 AWS Glue 연결에 대한 전역 설정입니다. 이 확인란을 선택 취소하면 이전에 암호화된 암호가 생성되었거나 업데이트되었을 때 사용된 키를 사용하여 암호화된 상태로 유지됩니다. AWS Glue 연결에 대한 자세한 정보는 데이터에 연결 섹션을 참조하세요.

     이 옵션을 활성화하는 경우 AWS KMS 키를 선택하거나 Enter a key ARN(키 ARN 입력)을 선택하고 키에 대한 Amazon 리소스 이름(ARN)을 제공합니다. `arn:aws:kms:region:account-id:key/key-id` 형식에 ARN 이름을 입력합니다. 또한 `arn:aws:kms:region:account-id:alias/alias-name` 같은 키 별칭으로 ARN을 제공할 수도 있습니다.

     > **Important**
     >
     > 이 옵션이 선택되어 있는 경우 연결을 생성하거나 업데이트하는 사용자 또는 역할에 지정된 KMS 키에 대한 `kms:Encrypt` 권한이 있어야 합니다.

     자세한 내용은 연결 암호 암호화 단원을 참조하십시오.
3. [설정(Settings)]을 선택하고 [권한(Permissions)] 편집기에서 계정에 대한 Data Catalog의 세분화된 액세스 제어를 변경할 정책 설명을 추가합니다. 한 번에 정책 하나만 Data Catalog에 연결할 수 있습니다. JSON 리소스 정책을 이 컨트롤에 붙여 넣을 수 있습니다. 자세한 내용은 AWS Glue 내의 리소스 기반 정책 단원을 참조하십시오.
4. [저장(Save)]을 선택하여 변경 사항으로 Data Catalog를 업데이트합니다.

또한 AWS Glue API 작업을 사용하여 리소스 정책을 설정하고, 가져오고, 삭제할 수 있습니다. 자세한 내용은 AWS Glue의 보안 API 단원을 참조하십시오.

트랜잭션 테이블 채우기 및 관리

Apache Iceberg, Apache Hudi 및 Linux Foundation Delta Lake는 Apache Spark에서 대규모 데이터 분석 및 데이터 레이크 워크로드를 처리하도록 설계된 오픈 소스 테이블 형식입니다.

다음과 같은 방법을 사용하여 AWS Glue Data Catalog에서 Iceberg, Hudi 및 Delta Lake 테이블을 채울 수 있습니다.

- AWS Glue 크롤러 - AWS Glue 크롤러는 데이터 카탈로그에서 Iceberg, Hudi 및 Delta Lake 테이블 메타데이터를 자동으로 검색하고 채울 수 있습니다. 자세한 내용은 크롤러를 사용하여 데이터 카탈로그 채우기 단원을 참조하십시오.
- AWS Glue ETL 작업 - ETL 작업을 생성하여 Iceberg, Hudi 및 Delta Lake 테이블에 데이터를 쓰고 데이터 카탈로그에서 해당 메타데이터를 채울 수 있습니다. 자세한 내용은 AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용을 참조하세요.
- AWS Glue 콘솔, AWS Lake Formation 콘솔, AWS CLI 또는 API - AWS Glue 콘솔, Lake Formation 콘솔 또는 API를 사용하여 데이터 카탈로그에서 Iceberg 테이블 정의를 생성하고 관리할 수 있습니다.

주제

- Apache Iceberg 테이블 생성
- Iceberg 테이블 최적화
- Iceberg 테이블의 쿼리 성능 최적화

Apache Iceberg 테이블 생성

Amazon S3에 상주하는 데이터를 사용하여 AWS Glue Data Catalog에서 Apache Parquet 데이터 형식을 사용하는 Apache Iceberg 테이블을 생성할 수 있습니다. 테이터 카탈로그의 테이블은 데이터 스토어의 데이터를 표현하는 메타데이터 정의입니다. 기본적으로 AWS Glue는 Iceberg v2 테이블을 생성합니다. v1과 v2 테이블의 차이점은 Apache Iceberg 설명서의 Format version changes(포맷 버전 변경 사항)을 참조하세요.

Apache Iceberg는 매우 큰 분석 데이터세트를 위한 오픈 테이블 형식입니다. Iceberg를 사용하면 스키마를 쉽게 변경할 수 있습니다(이를 스키마 진화라고도 함). 다시 말해서 사용자는 기본 데이터를 손상시키지 않고 데이터 테이블에서 열을 추가하거나, 이름을 바꾸거나, 제거할 수 있습니다. 또한 Iceberg는 사용자가 시간 경과에 따른 데이터 변경 사항을 추적할 수 있는 데이터 버전 관리를 지원합니다. 이

를 통해 사용자는 과거 버전의 데이터에 액세스하여 데이터를 쿼리하고 업데이트와 삭제 사이의 데이터 변화를 분석할 수 있습니다.

AWS Glue 또는 Lake Formation 콘솔을 사용하거나 AWS Glue API의 `CreateTable` 작업을 사용하여 데이터 카탈로그에 Iceberg 테이블을 생성할 수 있습니다. 자세한 내용은 CreateTable 작업(Python: create_table)을 참조하세요.

데이터 카탈로그에서 Iceberg 테이블을 생성할 때 Amazon S3에서 테이블 형식과 메타데이터 파일 경로를 지정해야 읽기 및 쓰기를 수행할 수 있습니다.

Lake Formation을 사용하면 AWS Lake Formation에 Amazon S3 데이터 위치를 등록할 때 세분화된 액세스 제어 권한을 사용하여 Iceberg 테이블을 보호할 수 있습니다. Lake Formation에 등록되지 않은 Amazon S3의 원본 데이터와 메타데이터의 경우 액세스 권한이 Amazon S3 및 AWS Glue 작업에 대한 IAM 권한 정책에 의해 결정됩니다. 자세한 내용은 권한 관리를 참조하세요.

> **Note**
>
> 데이터 카탈로그는 파티션 생성 및 Iceberg 테이블 속성 추가를 지원하지 않습니다.

사전 조건

데이터 카탈로그에서 Iceberg 테이블을 생성하고 Lake Formation 데이터 액세스 권한을 설정하려면 다음 요구 사항을 완료해야 합니다.

1. Lake Formation에 등록된 데이터 없이 Iceberg 테이블을 생성하는 데 필요한 권한.

   데이터 카탈로그에서 테이블을 생성하는 데 필요한 권한 외에도 테이블 생성자는 다음 권한이 필요합니다.

   - 리소스 arn:aws:s3:::{bucketName}에 대한 `s3:PutObject`
   - 리소스 arn:aws:s3:::{bucketName}에 대한 `s3:GetObject`
   - 리소스 arn:aws:s3:::{bucketName}에 대한 `s3:DeleteObject`
2. Lake Formation에 등록된 데이터를 사용하여 Iceberg 테이블을 생성하는 데 필요한 권한.

   Lake Formation을 사용하여 데이터 레이크의 데이터를 관리하고 보호하려면 테이블을 위한 데이터가 있는 Amazon S3 위치를 Lake Formation에 등록합니다. 이는 Lake Formation이 Athena, Redshift Spectrum 및 Amazon EMR과 같은 AWS 분석 서비스에 보안 인증 정보를 벤딩하여 데이

터에 액세스할 수 있도록 하기 위함입니다. Amazon S3 위치 등록에 대한 자세한 내용은 데이터 레이크에 Amazon S3 위치 추가 섹션을 참조하세요.

Lake Formation에 등록된 기본 데이터를 읽고 쓰는 보안 주체는 다음과 같은 권한이 필요합니다.

- `lakeformation:GetDataAccess`
- `DATA_LOCATION_ACCESS`

  위치에 대한 데이터 위치 권한이 있는 보안 주체는 모든 하위 위치에 대한 위치 권한도 갖습니다.

  데이터 위치 권한에 대한 자세한 내용은 기본 데이터 액세스 제어 섹션을 참조하세요.

압축을 활성화하려면 서비스가 데이터 카탈로그의 테이블을 업데이트할 권한이 있는 IAM 역할을 맡아야 합니다. 자세한 내용은 테이블 최적화 필수 조건 을 참조하세요.

Iceberg 테이블 생성

이 페이지에 설명된 대로 AWS Glue 또는 Lake Formation 콘솔을 사용하거나 AWS Command Line Interface를 사용하여 Iceberg v1 및 v2 테이블을 생성할 수 있습니다. 또는 AWS Glue 크롤러를 사용하여 Iceberg 테이블을 생성할 수도 있습니다. 자세한 내용은 AWS Glue 개발자 안내서의 데이터 카탈로그 및 크롤러를 참조하세요.

Iceberg 테이블을 생성하려면

Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 데이터 카탈로그에서 테이블을 선택하고 테이블 생성 버튼을 사용하여 다음 속성을 지정합니다.
   - 테이블 이름 - 테이블 이름을 입력합니다. Athena를 사용하여 테이블에 액세스하는 경우 Amazon Athena 사용 설명서의 이름 지정 팁을 사용하세요.
   - 데이터베이스 - 기존 데이터베이스를 선택하거나 새 데이터베이스를 생성합니다.
   - 설명 - 테이블에 대한 설명입니다. 테이블 내용을 이해할 수 있도록 설명을 적을 수 있습니다.
   - 테이블 형식 - 테이블 형식으로 Apache Iceberg를 선택합니다.
   - 압축 활성화 - 테이블의 작은 Amazon S3 객체를 더 큰 객체로 압축하려면 압축 활성화를 선택합니다.

- IAM 역할 - 압축을 실행하기 위해 서비스는 사용자를 대신하여 IAM 역할을 맡습니다. 드롭다운을 사용하여 IAM 역할을 선택할 수 있습니다. 압축 기능을 활성화하는 데 필요한 권한이 역할에 있는지 확인합니다.

  필요한 권한에 대해 알아보려면 테이블 최적화 필수 조건 섹션을 참조하세요.

- 위치 - 메타데이터 테이블을 저장하는 Amazon S3의 폴더 경로를 지정합니다. Iceberg가 읽기 및 쓰기를 수행하려면 데이터 카탈로그에 메타데이터 파일과 위치가 필요합니다.

- 스키마 - 열 추가를 선택하여 열과 열의 데이터 유형을 추가합니다. 빈 테이블을 생성하고 나중에 스키마를 업데이트할 수 있습니다. 데이터 카탈로그는 Hive 데이터 유형을 지원합니다. 자세한 내용은 Hive 데이터 유형을 참조하세요.

  Iceberg를 사용하면 테이블을 생성한 후 스키마와 파티션을 개선할 수 있습니다. Athena 쿼리를 사용하여 테이블 스키마를 업데이트하고 Spark 쿼리를 사용하여 파티션을 업데이트할 수 있습니다.

AWS CLI

```
aws glue create-table \
    --database-name iceberg-db \
    --region us-west-2 \
    --open-table-format-input '{
      "IcebergInput": {
           "MetadataOperation": "CREATE",
           "Version": "2"
        }
      }' \
    --table-input '{"Name":"test-iceberg-input-demo",
           "TableType": "EXTERNAL_TABLE",
           "StorageDescriptor":{
              "Columns":[
                  {"Name":"col1", "Type":"int"},
                  {"Name":"col2", "Type":"int"},
                  {"Name":"col3", "Type":"string"}
               ],
              "Location":"s3://DOC_EXAMPLE_BUCKET_ICEBERG/"
           }
       }'
```

Iceberg 테이블 최적화

AWS Glue에서는 AWS 분석 엔진 및 ETL 작업에서 사용되는 Apache Iceberg 테이블의 관리 및 성능을 개선하는 여러 테이블 최적화 옵션을 지원합니다. 이러한 최적화 프로그램은 효율적인 스토리지 활용, 향상된 쿼리 성능 및 효과적인 데이터 관리를 제공합니다. AWS Glue에서 사용할 수 있는 기본 옵티마이저에는 다음 세 가지 유형이 있습니다.

- 압축 - 데이터 압축은 작은 데이터 파일을 압축하여 스토리지 사용량을 줄이고 읽기 성능을 향상시킵니다. 데이터 파일이 병합 및 재작성되어 불필요한 데이터를 제거하고 조각난 데이터를 더 크고 효율적인 파일로 통합합니다. 자동으로 실행되도록 압축을 구성할 수 있습니다.

  Binpack은 Apache Iceberg의 기본 압축 전략입니다. 최적의 성능을 위해 더 작은 데이터 파일을 더 큰 데이터 파일로 결합합니다. 압축은 유사한 데이터를 함께 클러스터링하는 Sort 및 Z-order 전략도 지원합니다. Sort는 지정된 열을 기반으로 데이터를 구성하여 필터링된 작업에 대한 쿼리 성능을 개선합니다. Z-order는 여러 열이 동시에 쿼리될 때 쿼리 성능을 향상시키는 정렬된 데이터 세트를 생성합니다. Binpack, Sort, Z-order의 세 가지 압축 전략은 모두 쿼리 엔진에서 스캔하는 데이터의 양을 줄여 쿼리 처리 비용을 줄입니다.
- 스냅샷 보존 - 스냅샷은 Iceberg 테이블의 타임스탬프가 표시된 버전입니다. 스냅샷 보존 구성을 통해 고객은 스냅샷을 보존하는 기간과 보존할 스냅샷 수를 적용할 수 있습니다. 스냅샷 보존 최적화 프로그램을 구성하면 오래되고 불필요한 스냅샷과 연결된 파일을 제거하여 스토리지 오버헤드를 관리하는 데 도움이 될 수 있습니다.
- 분리된 파일 삭제 - 분리된 파일은 Iceberg 테이블 메타데이터에서 더 이상 참조되지 않는 파일입니다. 이러한 파일은 시간이 지남에 따라 누적될 수 있으며, 특히 테이블 삭제 같은 작업이나 ETL 작업 실패 이후에 누적될 수 있습니다. 분리된 파일 삭제를 활성화하면 AWS Glue에서 이러한 불필요한 파일을 주기적으로 식별 및 제거하여 스토리지를 확보할 수 있습니다.

AWS Glue 콘솔, AWS CLI 또는 AWS Glue API 작업을 사용하여 Data Catalog의 개별 Iceberg 테이블에 대한 압축, 스냅샷 보존 및 분리된 파일 삭제 최적화 프로그램을 활성화하거나 비활성화할 수 있습니다.

주제

- 테이블 최적화 필수 조건
- 압축 최적화
- 스냅샷 보존 최적화
- 분리된 파일 삭제
- 최적화 세부 정보 보기

- Amazon CloudWatch 지표 보기
- 옵티마이저 삭제
- 고려 사항 및 제한 사항
- 테이블 옵티마이저 지원 리전

테이블 최적화 필수 조건

테이블 옵티마이저는 테이블에 최적화 옵션(압축, 스냅샷 보존, 분리된 파일 삭제)을 활성화할 때 사용자가 지정하는 AWS Identity and Access Management(IAM) 역할의 권한을 갖습니다. 모든 옵티마이저에 대해 단일 역할을 생성하거나 각 옵티마이저에 대해 별도의 역할을 생성할 수 있습니다.

> **Note**
>
> 분리된 파일 삭제 최적화 프로그램에는 `glue:updateTable` 또는 `s3:putObject` 권한이 필요하지 않습니다. 스냅샷 만료 및 압축 최적화 프로그램에는 동일한 권한 세트가 필요합니다.

IAM 역할에는 데이터 카탈로그의 데이터를 읽고 메타데이터를 업데이트할 수 있는 권한이 있어야 합니다. IAM 역할을 생성하여 다음 인라인 정책을 연결할 수 있습니다.

- AWS Lake Formation에 등록되지 않은 데이터에 대한 위치에서 Amazon S3 읽기/쓰기 권한을 부여하는 다음 인라인 정책을 추가하니다. 이 정책에는 데이터 카탈로그의 테이블을 업데이트하고 AWS Glue에게 Amazon CloudWatch 로그에 로그를 추가하고 지표를 게시할 수 있는 권한도 포함되어 있습니다. Lake Formation에 등록되지 않은 Amazon S3의 원본 데이터는 액세스 권한이 Amazon S3 및 AWS Glue 작업에 대한 IAM 권한 정책에 의해 결정됩니다.

  다음 인라인 정책에서는 Amazon S3 버킷 이름이 있는 `bucket-name`, `aws-account-id`, `region`을 데이터 카탈로그의 유효한 AWS 계정 번호, 리전, `database_name`을 데이터베이스 이름으로, `table_name`은 테이블 이름으로 대체하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:UpdateTable",
                "glue:GetTable"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<aws-account-id>:table/<database-name>/<table-
name>",
                "arn:aws:glue:<region>:<aws-account-id>:database/<database-name>",
                "arn:aws:glue:<region>:<aws-account-id>:catalog"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/iceberg-
compaction/logs:*",
                "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/iceberg-
retention/logs:*",
                "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/iceberg-
orphan-file-deletion/logs:*"
        }
    ]
```

```
}
```

- Lake Formation에 등록된 데이터에 대해 압축을 활성화하려면 다음 정책을 사용하십시오.

  최적화 역할에 테이블에 대한 IAM_ALLOWED_PRINCIPALS 그룹 권한이 부여되지 않은 경우, 해당 역할에는 테이블에 대한 Lake Formation ALTER, DESCRIBE, INSERT 및 DELETE 권한이 필요합니다.

  Amazon S3 버킷의 Lake Formation 등록에 대한 자세한 내용은 데이터 레이크에 Amazon S3 위치 추가 섹션을 참조하세요.

```
{
 "Version": "2012-10-17",
 "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
        "glue:UpdateTable",
        "glue:GetTable"
        ],

        "Resource": [
            "arn:aws:glue:<region>:<aws-account-
id>:table/<databaseName>/<tableName>",
            "arn:aws:glue:<region>:<aws-account-id>:database/<database-name>",
            "arn:aws:glue:<region>:<aws-account-id>:catalog"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
        ],
        "Resource": [
```

```
            "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/iceberg-
compaction/logs:*",
            "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/iceberg-
retention/logs:*",
            "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/iceberg-
orphan-file-deletion/logs:*"
        }
    ]
 }
```

- (선택 사항)서버 측 암호화를 사용하여 암호화된 Amazon S3 버킷의 데이터로 Iceberg 테이블을 최적화하려면 압축 역할에 Amazon S3 객체를 해독하고 암호화된 버킷에 객체를 쓰기 위한 새 데이터 키를 생성할 수 있는 권한이 필요합니다. 다음 텍스트를 AWS KMS 키 정책에 추가합니다. 버킷 수준 암호화만 지원합니다.

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::<aws-account-id>:role/<optimizer-role-name>"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

- (선택 사항) Lake Formation에 등록된 데이터 위치의 경우, 위치를 등록하는 데 사용되는 역할에는 Amazon S3 객체를 해독하고 암호화된 버킷에 객체를 쓰기 위한 새 데이터 키를 생성할 수 있는 권한이 필요합니다. 자세한 내용을 알아보려면 암호화된 Amazon S3 위치 등록을 참조하십시오.
- (선택 사항) AWS KMS 키가 다른 AWS 계정에 저장되어 있는 경우 압축 역할에 다음 권한을 포함해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
```

```
            ],
            "Resource": ["arn:aws:kms:<REGION>:<KEY_OWNER_ACCOUNT_ID>:key/<KEY_ID>" ]
        }
    ]
}
```

- 압축을 실행하는 데 사용하는 역할에는 해당 역할에 대한 `iam:PassRole` 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<account-id>:role/<optimizer-role-name>"
            ]
        }
    ]
}
```

- 압축 프로세스를 실행하는 IAM 역할을 수임하도록 AWS Glue 서비스의 역할에 다음 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
 }
```

압축 최적화

Apache Iceberg와 같은 오픈 테이블 형식을 사용하는 Amazon S3 데이터 레이크는 데이터를 S3 객체로 저장합니다. 데이터 레이크 테이블에 수천 개의 작은 Amazon S3 객체가 있으면 메타데이터 오버헤드가 증가하고 읽기 성능에 영향을 미칩니다. AWS Glue Data Catalog에서는 Iceberg 테이블의 관리형 압축을 제공하고, Amazon Athena 및 Amazon EMR, AWS Glue ETL 작업과 같은 AWS 분석 서비스를 통해 작은 객체를 큰 객체로 압축하여 읽기 성능을 높입니다. 데이터 카탈로그는 동시 쿼리를 방해하지 않으면서 압축을 수행하고 Parquet 형식 테이블에 대해서만 압축을 지원합니다.

테이블 옵티마이저는 테이블 파티션을 지속적으로 모니터링하여 파일 수 및 파일 크기가 임곗값을 초과할 경우 압축 프로세스를 시작합니다.

Data Catalog에서는 테이블 또는 파티션에 100개가 넘는 파일이 있을 때 압축 프로세스가 시작됩니다. 각 파일은 대상 파일 크기의 75%보다 작아야 합니다. 대상 파일 크기는 `write.target-file-size-bytes` 테이블 속성에 의해 정의되며 명시적으로 설정되지 않은 경우 기본값은 512MB입니다.

제한 사항은 관리형 데이터 압축에 지원되는 형식 및 제한 사항 섹션을 참조하세요.

주제

- 압축 최적화 프로그램 활성화
- 압축 최적화 프로그램 비활성화

압축 최적화 프로그램 활성화

AWS Glue 콘솔, AWS CLI 또는 AWS API를 사용하여 AWS Glue Data Catalog에서 Apache Iceberg 테이블 압축을 활성화할 수 있습니다. 새 테이블의 경우 Apache Iceberg를 테이블 형식으로 선택하고 테이블을 생성할 때 압축을 활성화할 수 있습니다. 압축 기능은 새 테이블에 대해 기본적으로 비활성화되어 있습니다.

Console

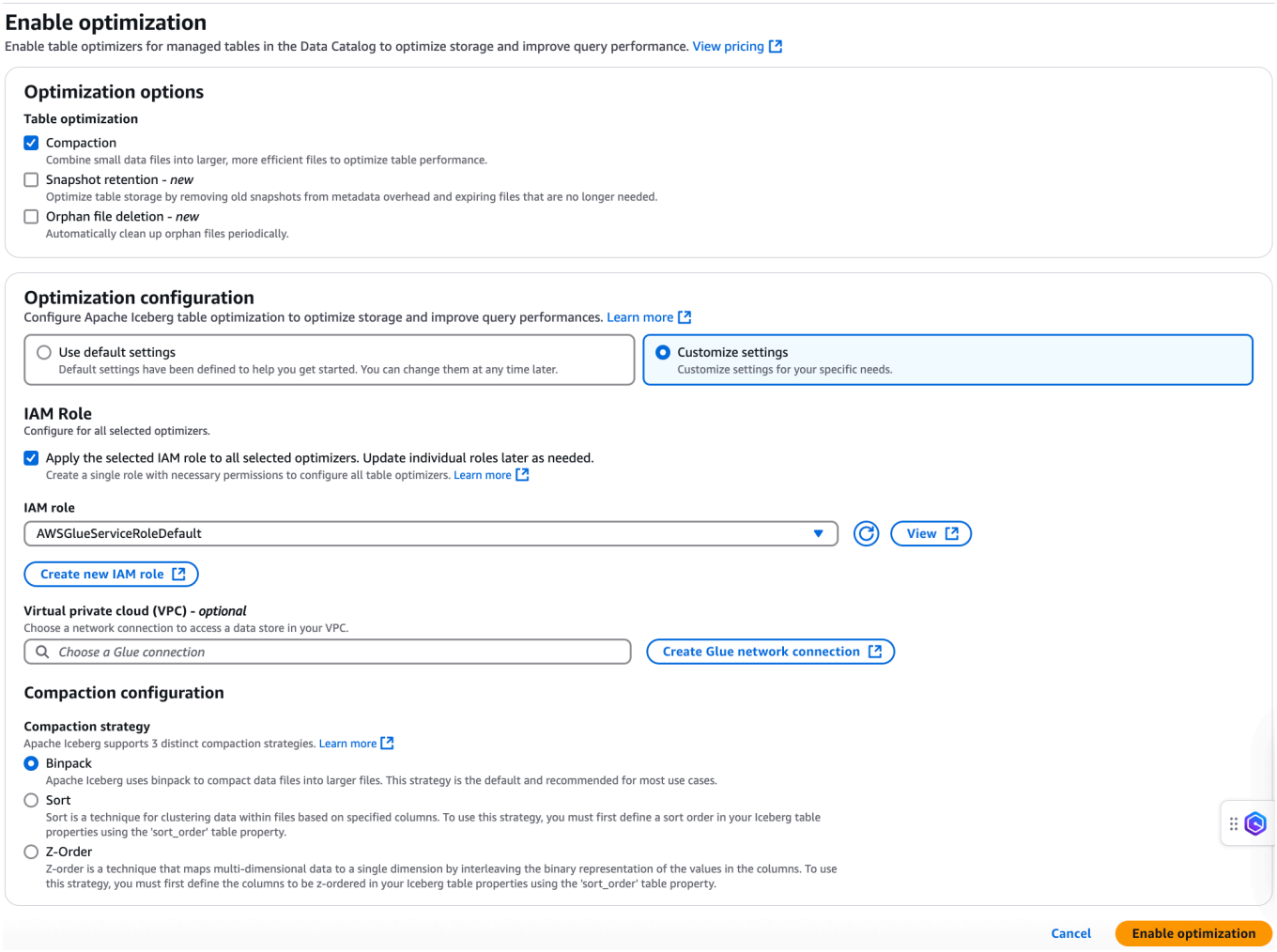
압축 기능 활성화하는 방법

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 열고 데이터 레이크 관리자, 테이블 작성자 또는 테이블에 대한 `glue:UpdateTable` 및 `lakeformation:GetDataAccess` 권한을 부여받은 사용자로 로그인합니다.
2. 탐색 창의 데이터 카탈로그에서 테이블을 선택합니다.
3. 테이블 페이지에서 압축을 활성화하려는 열린 테이블 형식의 테이블을 선택한 다음, 작업 메뉴 아래 최적화를 선택하고 활성화를 선택합니다.

테이블 세부 정보 페이지에서 테이블 최적화를 선택하여 압축을 활성화할 수도 있습니다. 페이지 하단에서 테이블 최적화 탭을 선택하고 압축 활성화를 선택합니다.

최적화 활성화 옵션은 Data Catalog에서 새 Iceberg 테이블을 생성할 때도 사용할 수 있습니다.

4. 최적화 활성화 페이지의 최적화 옵션 아래 압축을 선택합니다.



5. 그런 다음 테이블 최적화 필수 조건 섹션에 표시된 권한을 사용하여 드롭다운에서 IAM 역할을 선택합니다.

새 IAM 역할 생성 옵션을 선택하여 압축을 실행하는 데 필요한 권한이 있는 사용자 지정 역할을 생성할 수도 있습니다.

아래 단계에 따라 기존 IAM 역할을 업데이트하세요.

a. IAM 역할에 대한 권한 정책을 업데이트하려면 IAM 콘솔에서 컴팩션을 실행하는 데 사용되는 IAM 역할로 이동합니다.

b. 권한 추가 섹션에서 정책 생성을 선택합니다. 새로 열린 브라우저 창에서 역할에 사용할 새 정책을 생성합니다.

c. 정책 생성 페이지에서 JSON 탭을 선택합니다. 필수 조건에 표시된 JSON 코드를 정책 편집기 필드에 복사합니다.

6. Iceberg 테이블 옵티마이저가 특정 가상 프라이빗 클라우드(VPC)에서 Amazon S3 버킷에 액세스해야 하는 보안 정책 구성이 있는 경우 AWS Glue 네트워크 연결을 생성하거나 기존 연결을 사용하세요.

   AWS Glue VPC 연결을 아직 설정하지 않은 경우 AWS Glue 콘솔 또는 AWS CLI/SDK를 사용하여 커넥터에 대한 연결 생성 섹션의 단계에 따라 새 연결을 생성하세요.

7. 압축 전략을 선택합니다. 사용할 수 있는 옵션:

   - Binpack - Binpack은 Apache Iceberg의 기본 압축 전략입니다. 최적의 성능을 위해 더 작은 데이터 파일을 더 큰 데이터 파일로 결합합니다.
   - Sort - Apache Iceberg에서 정렬은 지정된 열을 기반으로 파일 내에서 정보를 클러스터링하여 처리해야 하는 파일 수를 줄여 쿼리 성능을 크게 개선하는 데이터 구성 기법입니다. 정렬 순서 필드를 사용하여 Iceberg 메타데이터에서 정렬 순서를 정의하고, 여러 개의 열이 지정되면 데이터가 순서에 따라 정렬되면 유사한 값의 레코드가 파일 내에 함께 저장됩니다. Sort 압축 전략은 파티션 내의 모든 파일에서 데이터를 정렬하여 최적화를 향상시킵니다.
   - Z-order - Z-order는 중요도가 동일한 여러 개의 열을 기준으로 정렬해야 할 때 데이터를 구성하는 방법입니다. 다른 열보다 한 열의 우선 순위를 높이는 기존 정렬과 달리 Z-order는 각 열에 균형 잡힌 가중치를 부여하므로 쿼리 엔진이 데이터를 검색할 때 더 적은 수의 파일을 읽을 수 있습니다.

     이 기술은 여러 열의 값을 이진수 형태로 함께 위빙합니다. 예를 들어, 두 열의 숫자가 3과 4인 경우 Z-순서는 먼저 이진수로 변환합니다(3은 011이 되고 4는 100이 됨). 이후 이러한 숫자를 인터리빙하여 새 값인 011010을 생성합니다. 이 인터리빙으로 관련 데이터를 물리적으로 서로 가깝게 유지하는 패턴이 생성됩니다.

     Z-order는 다차원 쿼리에 특히 효과적입니다. 예를 들어, 소득, 주 및 우편번호별로 Z-order가 적용된 고객 테이블은 여러 차원에서 쿼리할 때 계층적 정렬에 비해 뛰어난 성능을 제공할 수 있습니다. 이러한 구성을 통해 소득과 지리적 위치의 특정 조합을 대상으로 하는 쿼리가 불필요한 파일 스캔을 최소화하면서 관련 데이터를 빠르게 찾을 수 있습니다.

8. 최적화 활성화를 선택합니다.

AWS CLI

다음 예제는 압축 기능을 활성화하는 방법을 보여줍니다. 계정 ID를 유효한 AWS 계정 ID로 바꿉니다. 데이터베이스 이름과 테이블 이름을 실제 Iceberg 테이블 이름 및 데이터베이스 이름으로 바꿉니다. `roleArn`을 IAM 역할의 AWS 리소스 이름(ARN)과 압축 실행에 필요한 권한이 있는 IAM 역할의 이름으로 바꿉니다. `sort` 압축 전략을 `z-order` 또는 `binpack`과 같이 다른 지원되는 전략으로 바꿀 수 있습니다.

순서는 요구 사항에 따라 다릅니다.

```
aws glue create-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration '{
    "roleArn": "arn:aws:iam::123456789012:role/optimizer_role",
    "enabled": true,
    "vpcConfiguration": {"glueConnectionName": "glue_connection_name"},
    "compactionConfiguration": {
      "icebergConfiguration": {"strategy": "sort"}
    }
  }'\
--type compaction
```

AWS API

CreateTableOptimizer 작업을 직접 호출하여 테이블의 압축을 활성화합니다.

압축을 활성화하면 압축 실행이 완료되면 테이블 최적화 탭에 다음과 같은 압축 세부 정보가 표시됩니다.

시작 시간

압축 프로세스가 데이터 카탈로그에서 시작되는 시간입니다. 값은 UTC 시간으로 표시된 타임스탬프입니다.

종료 시간

압축 프로세스가 데이터 카탈로그에서 끝나는 시간입니다. 값은 UTC 시간으로 표시된 타임스탬프입니다.

상태 표시기

압축 실행의 상태입니다. 값은 성공 또는 실패입니다.

압축된 파일 수

압축된 총 파일 수입니다.

압축된 바이트 수

압축된 총 바이트 수입니다.

압축 최적화 프로그램 비활성화

AWS Glue 콘솔 또는 AWS CLI를 사용하여 특정 Apache Iceberg 테이블에 대한 자동 압축을 비활성화할 수 있습니다.

Console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 Data Catalog에서 테이블을 선택합니다.
3. 테이블 목록에서 압축을 비활성화하려는 Iceberg 테이블을 선택합니다.
4. 테이블 세부 정보 페이지 하단 섹션에서 테이블 최적화 탭을 선택합니다.
5. 작업에서 비활성화를 선택한 다음, 압축을 선택합니다.
6. 확인 메시지에서 압축 비활성화를 선택합니다. 압축을 나중에 다시 활성화할 수 있습니다.

   확인하면 압축이 비활성화되고 테이블의 압축 상태가 `Disabled`로 다시 켜집니다.

AWS CLI

다음 예제에서 계정 ID를 유효한 AWS 계정 ID로 바꿉니다. 데이터베이스 이름과 테이블 이름을 실제 Iceberg 테이블 이름 및 데이터베이스 이름으로 바꿉니다. `roleArn`을 IAM 역할의 AWS 리소스 이름(ARN)과 압축 실행에 필요한 권한이 있는 IAM 역할의 실제 이름으로 대체합니다.

```
aws glue update-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
```

```
 --table-optimizer-configuration
'{"roleArn":"arn:aws:iam::123456789012:role/optimizer_role", "enabled":'false',
"vpcConfiguration":{"glueConnectionName":"glue_connection_name"}}'\
 --type compaction
```

AWS API

UpdateTableOptimizer 작업을 직접 호출하여 특정 테이블에 대한 압축을 비활성화합니다.

스냅샷 보존 최적화

Apache Iceberg 스냅샷 보존 기능을 사용하면 사용자는 특정 시점의 과거 데이터를 쿼리하고 테이블에 대한 원치 않는 수정 사항을 되돌릴 수 있습니다. AWS Glue Data Catalog에서 스냅샷 보존 구성은 이러한 스냅샷(테이블 데이터 버전)이 만료 및 제거되기 전에 보관되는 기간을 제어합니다. 이렇게 하면 구성된 보존 기간 또는 보관할 최대 스냅샷 수를 기준으로 오래된 스냅샷을 자동으로 제거하여 스토리지 비용과 메타데이터 오버헤드를 관리하는 데 도움이 됩니다.

보존 기간(일)과 테이블에 유지할 최대 스냅샷 수를 구성할 수 있습니다. AWS Glue에서 최신 스냅샷이 구성된 한도까지 유지되면서 지정된 보존 기간보다 오래된 스냅샷은 테이블 메타데이터에서 제거됩니다. 메타데이터에서 오래된 스냅샷을 제거한 후 AWS Glue에서는 더 이상 참조되지 않고 만료된 스냅샷에만 있는 해당 데이터 및 메타데이터 파일을 삭제합니다. 이렇게 하면 유지된 스냅샷 시점까지의 쿼리가 가능하고 만료된 스냅샷 데이터에 사용된 스토리지 공간을 확보할 수 있습니다.

주제

- 스냅샷 보존 옵티마이저 활성화
- 스냅샷 보존 옵티마이저 업데이트
- 스냅샷 보존 옵티마이저 비활성화

스냅샷 보존 옵티마이저 활성화

AWS Glue 콘솔, AWS CLI 또는 AWS API를 사용하여 데이터 카탈로그의 Apache Iceberg 테이블에서 스냅샷 보존 옵티마이저를 활성화할 수 있습니다. 새 테이블의 경우 Apache Iceberg를 테이블 형식으로 선택하고 테이블을 생성할 때 스냅샷 보존 옵티마이저를 활성화할 수 있습니다. 스냅샷 보존은 새 테이블에 대해 기본적으로 비활성화되어 있습니다.

Console

스냅샷 보존 옵티마이저 활성화

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 열고 데이터 레이크 관리자, 테이블 작성자 또는 테이블에 대한 `glue:UpdateTable` 및 `lakeformation:GetDataAccess` 권한을 부여받은 사용자로 로그인합니다.
2. 탐색 창의 데이터 카탈로그에서 테이블을 선택합니다.
3. 테이블 페이지에서 스냅샷 보존 옵티마이저를 활성화하려는 Iceberg 테이블을 선택한 다음 작업 메뉴의 최적화에서 활성화를 선택합니다.

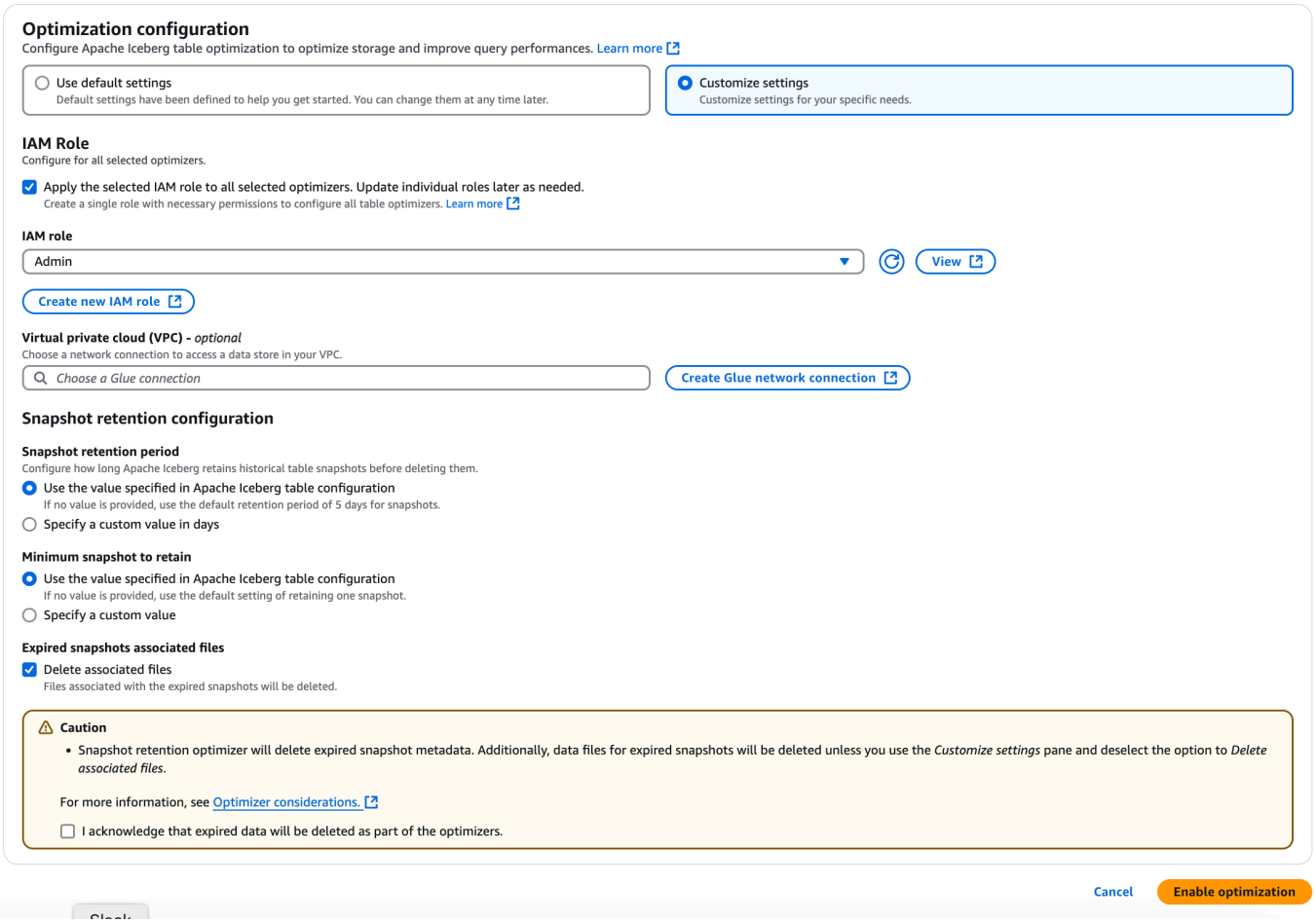
   테이블을 선택하고 테이블 세부 정보 페이지를 열어 최적화를 활성화할 수도 있습니다. 페이지 하단에서 테이블 최적화 탭을 선택하고 스냅샷 보존 활성화를 선택합니다.
4. 최적화 활성화 페이지의 최적화 구성에는 기본 설정 사용 또는 설정 사용자 지정이라는 두 가지 옵션이 있습니다. 기본 설정을 사용하도록 선택한 경우 AWS Glue에서 Iceberg 테이블 구성에 정의된 속성을 활용하여 스냅샷 보존 기간과 보존할 스냅샷 수를 결정합니다. 이 구성이 없는 경우 AWS Glue는 5일 동안 스냅샷 하나를 유지하고 만료된 스냅샷과 연결된 파일을 삭제합니다.
5. 다음으로 옵티마이저를 실행하기 위해 사용자를 대신하여 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다. IAM 역할에 필요한 권한에 대한 자세한 내용은 테이블 최적화 필수 조건 섹션을 참조하세요.

   아래 단계에 따라 기존 IAM 역할을 업데이트하세요.

   a. IAM 역할에 대한 권한 정책을 업데이트하려면 IAM 콘솔에서 컴팩션을 실행하는 데 사용되는 IAM 역할로 이동합니다.

   b. 권한 추가 섹션에서 정책 생성을 선택합니다. 새로 열린 브라우저 창에서 역할에 사용할 새 정책을 생성합니다.

   c. 정책 생성 페이지에서 JSON 탭을 선택합니다. 필수 조건에 표시된 JSON 코드를 정책 편집기 필드에 복사합니다.
6. 스냅샷 보존 구성 값을 수동으로 설정하려면 설정 사용자 지정을 선택합니다.



7. 모든 최적화 프로그램 활성화에 대해 단일 IAM 역할을 사용하려면 선택한 최적화 프로그램에 선택한 IAM 역할 적용 옵션 확인란을 선택합니다.
8. Iceberg 테이블 옵티마이저가 특정 가상 프라이빗 클라우드(VPC)에서 Amazon S3 버킷에 액세스해야 하는 보안 정책 구성이 있는 경우 AWS Glue 네트워크 연결을 생성하거나 기존 연결을 사용하세요.

   AWS Glue VPC 연결을 아직 설정하지 않은 경우 AWS Glue 콘솔 또는 AWS CLI/SDK를 사용하여 커넥터에 대한 연결 생성 섹션의 단계에 따라 새 연결을 생성하세요.
9. 다음으로 스냅샷 보존 구성에서 Iceberg 테이블 구성에 지정된 값을 사용하도록 선택하거나 스냅샷 보존 기간(history.expire.max-snapshot-age-ms) 및 보존할 최소 스냅샷 수(history.expire.min-snapshots-to-keep)에 대한 사용자 지정 값을 지정합니다.
10. 테이블 옵티마이저가 테이블 메타데이터에서 오래된 스냅샷을 삭제할 때 그에 속한 파일을 삭제하려면 관련 파일 삭제를 선택합니다.

    이 옵션을 선택하지 않으면 오래된 스냅샷이 테이블 메타데이터에서 제거되더라도 관련 파일은 분리된 파일로 스토리지에 남아 있게 됩니다.

11.다음으로 주의 설명을 읽고 동의합니다를 선택하여 계속 진행합니다.

> **Note**
>
> 데이터 카탈로그에서 스냅샷 보존 옵티마이저는 브랜치 및 태그 수준 보존 정책에 의해 제어되는 수명 주기를 준수합니다. 자세한 내용은 Iceberg 설명서의 Branching and tagging 섹션을 참조하세요.

12.구성을 검토하고 최적화 활성화를 선택합니다.

보존 옵티마이저가 실행되고 구성을 기반으로 이전 스냅샷이 만료될 때까지 몇 분 정도 기다립니다.

AWS CLI

AWS Glue에서 새 Iceberg 테이블에 대한 스냅샷 보존을 활성화하려면 `retention` 유형의 테이블 옵티마이저를 생성하고 `table-optimizer-configuration`에서 `enabled` 필드를 `true`로 설정해야 합니다. AWS CLI 명령 `create-table-optimizer` 또는 `update-table-optimizer`를 사용할 수 있습니다. 또한 요구 사항을 기반으로 `snapshotRetentionPeriodInDays` 및 `numberOfSnapshotsToRetain` 등과 같은 보존 구성 필드를 지정해야 합니다.

다음 예제에서는 스냅샷 보존 옵티마이저를 활성화하는 방법을 보여줍니다. 계정 ID를 유효한 AWS 계정 ID로 바꿉니다. 데이터베이스 이름과 테이블 이름을 실제 Iceberg 테이블 이름 및 데이터베이스 이름으로 바꿉니다. `roleArn`을 스냅샷 보존 옵티마이저 실행에 필요한 권한이 있는 IAM 역할의 AWS 리소스 이름(ARN)과 이름으로 바꿉니다.

```
aws glue create-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/optimizer_role","enabled":'true',
 "vpcConfiguration":{
"glueConnectionName":"glue_connection_name"}, "retentionConfiguration":
{"icebergConfiguration":
{"snapshotRetentionPeriodInDays":7,"numberOfSnapshotsToRetain":3,"cleanExpiredFiles":'true'}
  --type retention
```

이 명령은 지정된 카탈로그, 데이터베이스 및 리전의 지정된 Iceberg 테이블에 대한 보존 옵티마이저를 생성합니다. table-optimizer-configuration은 사용할 IAM 역할 ARN을 지정하고, 옵티마이저를

활성화하고, 보존 구성을 설정합니다. 이 예제에서는 스냅샷을 7일 동안 보존하고, 최소 3개의 스냅샷을 유지하고, 만료된 파일을 정리합니다.

- snapshotRetentionPeriodInDays - 만료 전에 스냅샷을 보존할 기간(일)입니다. 기본값은 5입니다.
- numberOfSnapshotsToRetain - 보존 기간보다 오래된 경우에도 보관할 최소 스냅샷 수입니다. 기본값은 1입니다.
- cleanExpiredFiles - 스냅샷이 만료된 후 만료된 데이터 파일을 삭제할지 여부를 나타내는 부울입니다. 기본값은 `true`입니다.

  true로 설정하면 이전 스냅샷은 테이블 메타데이터에서 제거되고 그에 속한 파일은 삭제됩니다. 이 파라미터를 false로 설정하면 이전 스냅샷은 테이블 메타데이터에서 제거되지만 그에 속한 파일은 스토리지에 분리된 파일로 남아 있습니다.

AWS API

CreateTableOptimizer 작업을 직접 호출하여 테이블에 대한 스냅샷 보존 옵티마이저를 활성화합니다.

압축을 활성화하면 테이블 최적화 탭에 다음과 같은 압축 세부 정보가 표시됩니다(약 15~20분 후).

시작 시간

스냅샷 보존 옵티마이저가 시작된 시간입니다. 값은 UTC 시간으로 표시된 타임스탬프입니다.

런타임

옵티마이저가 작업을 완료하는 데 걸리는 시간을 나타냅니다. 값은 UTC 시간으로 표시된 타임스탬프입니다.

상태 표시기

옵티마이저 실행의 상태입니다. 값은 성공 또는 실패입니다.

삭제된 데이터 파일

삭제된 전체 파일 수입니다.

삭제된 매니페스트 파일

삭제된 전체 매니페스트 파일 수입니다.

삭제된 매니페스트 목록

삭제된 전체 매니페스트 목록 수입니다.

스냅샷 보존 옵티마이저 업데이트

AWS Glue 콘솔, AWS CLI 또는 UpdateTableOptimizer API를 사용하여 특정 Apache Iceberg 테이블에 대한 스냅샷 보존 옵티마이저의 기존 구성을 업데이트할 수 있습니다.

Console

스냅샷 보존 구성 업데이트

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 데이터 카탈로그를 선택하고 테이블을 선택합니다. 테이블 목록에서 스냅샷 보존 최적화 프로그램 구성을 업데이트하려는 Iceberg 테이블을 선택합니다.
3. 테이블 세부 정보 페이지의 하단 섹션에서 테이블 최적화 탭을 선택한 다음, 편집을 선택합니다. 페이지의 오른쪽 상단에 있는 작업 메뉴에서 최적화 아래 편집을 선택할 수도 있습니다.
4. 최적화 편집 페이지에서 원하는 대로 변경합니다.
5. Save(저장)를 선택합니다.

AWS CLI

AWS CLI를 사용하여 스냅샷 보존 옵티마이저를 업데이트하려면 다음 명령을 사용합니다.

```
aws glue update-table-optimizer \
 --catalog-id 123456789012 \
 --database-name iceberg_db \
 --table-name iceberg_table \
 --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/optimizer_role"","enabled":'true',
 "vpcConfiguration":
{"glueConnectionName":"glue_connection_name"},"retentionConfiguration":
{"icebergConfiguration":
{"snapshotRetentionPeriodInDays":7,"numberOfSnapshotsToRetain":3,"cleanExpiredFiles":'true'}
 \
 --type retention
```

이 명령은 지정된 카탈로그, 데이터베이스 및 리전의 지정된 테이블에 대한 보존 구성을 업데이트합니다. 주요 파라미터:

- snapshotRetentionPeriodInDays - 만료 전에 스냅샷을 보존할 기간(일)입니다. 기본값은 1입니다.
- numberOfSnapshotsToRetain - 보존 기간보다 오래된 경우에도 보관할 최소 스냅샷 수입니다. 기본값은 5입니다.
- cleanExpiredFiles - 스냅샷이 만료된 후 만료된 데이터 파일을 삭제할지 여부를 나타내는 부울입니다. 기본값은 `true`입니다.

  true로 설정하면 이전 스냅샷은 테이블 메타데이터에서 제거되고 그에 속한 파일은 삭제됩니다. 이 파라미터를 false로 설정하면 이전 스냅샷은 테이블 메타데이터에서 제거되지만 그에 속한 파일은 스토리지에 분리된 파일로 남아 있습니다.

API

테이블 옵티마이저를 업데이트하려면 `UpdateTableOptimizer` API를 사용할 수 있습니다. 이 API를 사용하면 압축, 보존 또는 분리된 파일 제거에 대해 기존 테이블 옵티마이저의 구성을 업데이트할 수 있습니다. 요청 파라미터:

- catalogId(필수): 테이블을 포함하는 카탈로그의 ID
- databaseName(선택 사항): 테이블을 포함하는 데이터베이스의 이름
- tableName(선택 사항): 테이블의 이름
- type(필수): 테이블 옵티마이저의 유형(compaction, retention 또는 orphan_file_deletion)
- retentionConfiguration(필수): 역할 ARN, 활성화 상태, 보존 구성 및 분리된 파일 제거 구성을 포함하여 테이블 옵티마이저의 업데이트된 구성.

스냅샷 보존 옵티마이저 비활성화

AWS Glue 콘솔 또는 AWS CLI를 사용하여 특정 Apache Iceberg 테이블에 대한 스냅샷 보존 옵티마이저를 비활성화할 수 있습니다.

Console

스냅샷 보존 비활성화

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

2. 데이터 카탈로그를 선택하고 테이블을 선택합니다. 테이블 목록에서 스냅샷 보존 옵티마이저를 비활성화할 Iceberg 테이블을 선택합니다.
3. 테이블 세부 정보 페이지 하단 섹션에서 테이블 최적화를 선택하고 작업 아래에서 비활성화, 스냅샷 보존을 선택합니다.

   페이지의 오른쪽 상단에 있는 작업 메뉴에서 최적화 아래 비활성화를 선택할 수도 있습니다.
4. 확인 메시지에서 비활성화를 선택합니다. 스냅샷 보존 옵티마이저를 나중에 다시 활성화할 수 있습니다.

   확인한 후에는 스냅샷 보존 옵티마이저가 비활성화되고 스냅샷 보존 상태가 `Not enabled`가 됩니다.

AWS CLI

다음 예제에서 계정 ID를 유효한 AWS 계정 ID로 바꿉니다. 데이터베이스 이름과 테이블 이름을 실제 Iceberg 테이블 이름 및 데이터베이스 이름으로 바꿉니다. `roleArn`을 보존 옵티마이저 실행에 필요한 권한이 있는 IAM 역할의 AWS 리소스 이름(ARN)과 실제 이름으로 바꿉니다.

```
aws glue update-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/optimizer_role", "vpcConfiguration":
{"glueConnectionName":"glue_connection_name"}, "enabled":'false'}'\
  --type retention
```

AWS API

UpdateTableOptimizer 작업을 직접 호출하여 특정 테이블에 대한 스냅샷 보존 옵티마이저를 비활성화합니다.

분리된 파일 삭제

AWS Glue Data Catalog에서는 Iceberg 테이블에서의 분리된 파일 제거를 허용합니다. 분리된 파일은 지정된 테이블 위치 아래의 Amazon S3 데이터 소스에 존재하고 Iceberg 테이블 메타데이터에서 추적되지 않으며 구성된 수명 제한보다 오래된 파일입니다. 이러한 분리된 파일은 압축, 파티션 삭제 또는 테이블 재작성과 같은 작업으로 인해 시간이 지남에 따라 누적되어 불필요한 스토리지 공간을 차지할 수 있습니다.

AWS Glue의 분리된 파일 삭제 옵티마이저는 테이블 메타데이터와 실제 데이터 파일을 스캔하고, 분리된 파일을 식별 및 삭제하여 스토리지 공간을 확보합니다.

데이터 카탈로그에서 분리된 파일 삭제 테이블 옵티마이저를 생성하여 분리된 파일 삭제를 시작할 수 있습니다.

> **Important**
>
> 기본적으로 분리된 파일 삭제는 AWS Glue 테이블 위치 전체에서 파일을 평가합니다. 평가 범위를 제한하도록 하위 접두사를 구성할 수 있지만, 테이블 위치에 다른 데이터 소스 또는 테이블의 파일이 포함되어서는 안 됩니다. 테이블 위치가 다른 데이터 소스와 겹치는 경우, 서비스가 관련 없는 파일을 분리된 파일로 식별하고 삭제할 수 있습니다.

주제

- 분리된 파일 삭제 활성화
- 분리된 파일 삭제 옵티마이저 업데이트
- 분리된 파일 삭제 비활성화

분리된 파일 삭제 활성화

AWS Glue 콘솔, AWS CLI 또는 AWS API를 사용하여 데이터 카탈로그의 Apache Iceberg 테이블에서 분리된 파일 삭제를 활성화할 수 있습니다. 새 테이블의 경우 Apache Iceberg를 테이블 형식으로 선택하고 테이블을 생성할 때 분리된 파일 삭제 옵티마이저를 활성화할 수 있습니다. 스냅샷 보존은 새 테이블에 대해 기본적으로 비활성화되어 있습니다.

Console

분리된 파일 삭제 활성화

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 열고 데이터 레이크 관리자, 테이블 작성자 또는 테이블에 대한 `glue:UpdateTable` 및 `lakeformation:GetDataAccess` 권한을 부여받은 사용자로 로그인합니다.
2. 탐색 창의 데이터 카탈로그에서 테이블을 선택합니다.
3. 테이블 페이지에서 분리된 파일 삭제를 활성화하려는 Iceberg 테이블을 선택합니다.

   페이지 하단 섹션에서 테이블 최적화 탭을 선택하고 작업에서 활성화, 분리된 파일 삭제를 선택합니다.

페이지의 오른쪽 상단에 있는 작업 메뉴에서 최적화 아래 활성화를 선택할 수도 있습니다.

4. 최적화 활성화 페이지의 최적화 옵션에서 분리된 파일 삭제를 선택합니다.
5. 기본 설정을 사용하는 경우 3일 후에 모든 분리된 파일이 삭제됩니다. 분리된 파일을 특정 일수 동안 유지하려면 설정 사용자 지정을 선택합니다.
6. 다음으로 분리된 파일을 삭제하는 데 필요한 권한이 있는 IAM 역할을 선택합니다.
7. Iceberg 테이블 옵티마이저가 특정 가상 프라이빗 클라우드(VPC)에서 Amazon S3 버킷에 액세스해야 하는 보안 정책 구성이 있는 경우 AWS Glue 네트워크 연결을 생성하거나 기존 연결을 사용하세요.

   AWS Glue VPC 연결을 아직 설정하지 않은 경우 AWS Glue 콘솔 또는 AWS CLI/SDK를 사용하여 커넥터에 대한 연결 생성 섹션의 단계에 따라 새 연결을 생성하세요.
8. 설정 사용자 지정을 선택하는 경우 분리된 파일 삭제 구성에서 삭제하기 전에 파일을 유지할 기간(일)을 입력합니다.
9. 최적화 활성화를 선택합니다.

AWS CLI

AWS Glue에서 Iceberg 테이블에 대해 분리된 파일 삭제를 활성화하려면 `orphan_file_deletion` 유형의 테이블 옵티마이저를 생성하고 `enabled` 필드를 true로 설정해야 합니다. AWS CLI를 사용하여 Iceberg 테이블에 대해 분리된 파일 삭제 옵티마이저를 생성하려면 다음 명령을 사용할 수 있습니다.

```
aws glue create-table-optimizer \
 --catalog-id 123456789012 \
 --database-name iceberg_db \
 --table-name iceberg_table \
 --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/optimizer_role","enabled":true,
 "vpcConfiguration":{
"glueConnectionName":"glue_connection_name"}, "orphanFileDeletionConfiguration":
{"icebergConfiguration":{"orphanFileRetentionPeriodInDays":3, "location":'S3
 location'}}}'\
 --type orphan_file_deletion
```

이 명령은 지정된 Iceberg 테이블에 대해 분리된 파일 삭제 옵티마이저를 생성합니다. 주요 파라미터:

- roleArn - S3 버킷과 Glue 리소스에 대한 액세스 권한이 있는 IAM 역할의 ARN.
- enabled - true로 설정하면 옵티마이저를 활성화합니다.
- orphanFileRetentionPeriodInDay - 분리된 파일을 삭제하기 전에 유지할 기간(일 단위, 최소 1일).
- type - orphan_file_deletion으로 설정하면 분리된 파일 삭제 옵티마이저를 생성합니다.

테이블 옵티마이저를 생성한 후에는 정기적으로 분리된 파일 삭제를 실행합니다(활성화된 상태로 두면 하루에 한 번). `list-table-optimizer-runs` API를 사용하여 실행을 확인할 수 있습니다. 분리된 파일 삭제 작업은 테이블의 Iceberg 메타데이터에서 추적되지 않는 파일을 식별하고 삭제합니다.

API

CreateTableOptimizer 작업을 직접 호출하여 특정 테이블에 대한 분리된 파일 삭제 옵티마이저를 생성합니다.

분리된 파일 삭제 옵티마이저 업데이트

AWS Glue 콘솔, AWS CLI 또는 `UpdateTableOptimizer` 작업을 사용하여 분리된 파일의 보존 기간 변경이나 옵티마이저에서 사용하는 IAM 역할 변경 등 분리된 파일 삭제 옵티마이저의 구성을 수정할 수 있습니다.

AWS Management Console

분리된 파일 삭제 옵티마이저 업데이트

1. 데이터 카탈로그를 선택하고 테이블을 선택합니다. 테이블 목록에서 분리된 파일 삭제 옵티마이저 구성을 업데이트하려는 테이블을 선택합니다.
2. 테이블 세부 정보 페이지의 하단 섹션에서 테이블 최적화를 선택한 다음, 편집을 선택합니다.
3. 최적화 편집 페이지에서 원하는 대로 변경합니다.
4. Save(저장)를 선택합니다.

AWS CLI

AWS Glue에서 `update-table-optimizer` 호출을 사용하여 분리된 파일 삭제 옵티마이저를 업데이트할 수 있습니다. 이를 통해 업데이트된 `OrphanFileRetentionPeriodInDays`를 지정할 수 있는 `icebergConfiguration` 필드의 `OrphanFileDeletionConfiguration`을 수정하여

분리된 파일을 보존할 기간(일)을 설정하고, 분리된 파일을 삭제할 Iceberg 테이블 위치를 지정할 수 있습니다.

```
aws glue update-table-optimizer \
 --catalog-id 123456789012 \
 --database-name iceberg_db \
 --table-name Iceberg_table \
 --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/optimizer_role","enabled":true,
 "vpcConfiguration":
{"glueConnectionName":"glue_connection_name"},"orphanFileDeletionConfiguration":
{"icebergConfiguration":{"orphanFileRetentionPeriodInDays":5}}}' \
 --type orphan_file_deletion
```

API

UpdateTableOptimizer 작업을 직접 호출하여 테이블에 대해 분리된 파일 삭제 옵티마이저를 업데이트합니다.

분리된 파일 삭제 비활성화

AWS Glue 콘솔 또는 AWS CLI를 사용하여 특정 Apache Iceberg 테이블에 대한 분리된 파일 삭제 옵티마이저를 비활성화할 수 있습니다.

Console

분리된 파일 삭제 비활성화

1. 데이터 카탈로그를 선택하고 테이블을 선택합니다. 테이블 목록에서 분리된 파일 삭제 옵티마이저를 비활성화할 Iceberg 테이블을 선택합니다.
2. 테이블 세부 정보 페이지 하단 섹션에서 테이블 최적화 탭을 선택합니다.
3. 작업을 선택한 다음, 비활성화, 분리된 파일 삭제를 선택합니다.

   작업 메뉴의 최적화에서 비활성화를 선택할 수도 있습니다.
4. 확인 메시지에서 비활성화를 선택합니다. 분리된 파일 삭제 옵티마이저를 나중에 다시 활성화할 수 있습니다.

   확인한 이후 분리된 파일 삭제 옵티마이저가 비활성화되고 분리된 파일 삭제 상태가 `Not enabled`가 됩니다.

AWS CLI

다음 예제에서 계정 ID를 유효한 AWS 계정 ID로 바꿉니다. 데이터베이스 이름과 테이블 이름을 실제 Iceberg 테이블 이름 및 데이터베이스 이름으로 바꿉니다. `roleArn`을 옵티마이저 비활성화에 필요한 권한이 있는 IAM 역할의 AWS 리소스 이름(ARN)과 실제 이름으로 바꿉니다.

```
aws glue update-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/optimizer_role", "enabled":'false'}'\
  --type orphan_file_deletion
```

API

UpdateTableOptimizer 작업을 직접 호출하여 특정 테이블에 대한 스냅샷 보존 옵티마이저를 비활성화합니다.

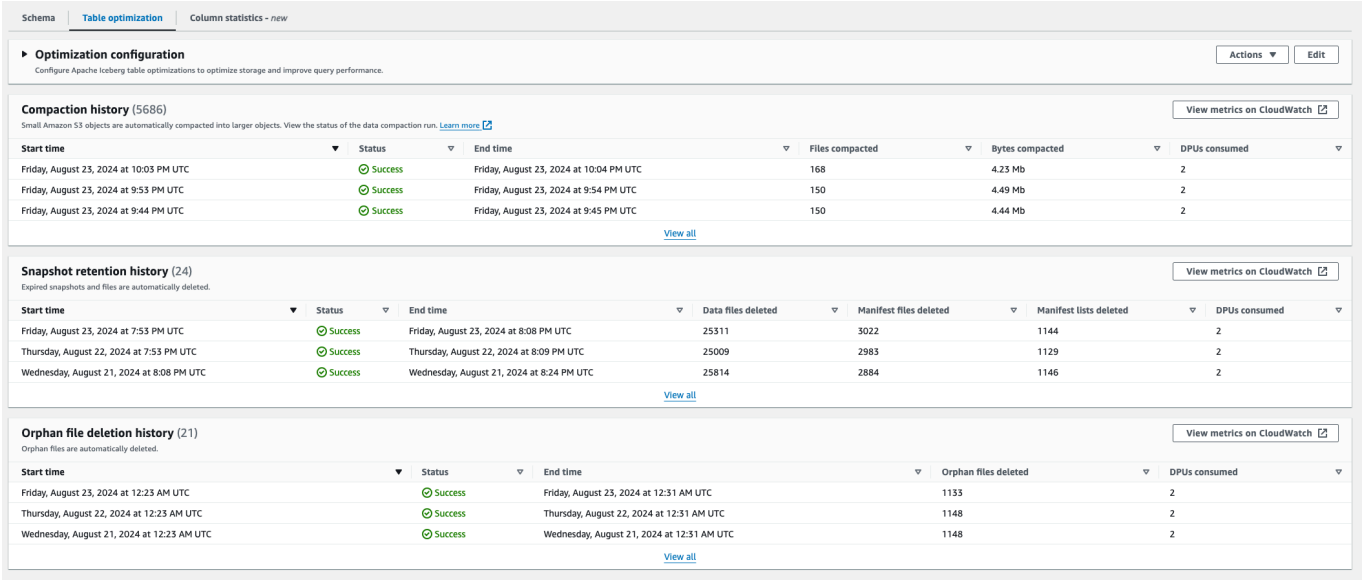
최적화 세부 정보 보기

AWS Glue 콘솔, AWS CLI 또는 AWS API 작업을 사용하여 Apache Iceberg의 최적화 상태를 볼 수 있습니다.

Console

Iceberg 테이블의 최적화 상태 보기(콘솔)

- AWS Glue 콘솔에서 데이터 카탈로그의 테이블 목록에서 Iceberg 테이블을 선택하면 Iceberg 테이블의 최적화 상태를 볼 수 있습니다. 테이블 최적화에서 모두 보기를 선택합니다.



AWS CLI

AWS CLI를 사용하여 최적화 세부 정보를 볼 수 있습니다.

다음 예제에서 계정 ID를 유효한 AWS 계정 ID로, 데이터베이스 이름과 테이블 이름을 실제 Iceberg 데이터베이스 이름과 테이블 이름으로 바꿉니다. `type`의 경우 최적화 유형이 제공됩니다. 사용 가능한 값은 `compaction`, `retention` 및 `orphan-file-deletion`입니다.

- 테이블의 마지막 압축 실행 세부 정보를 가져오려면

```
aws get-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

- 다음 예제를 사용하여 특정 테이블에 대한 옵티마이저 기록을 검색할 수 있습니다.

```
aws list-table-optimizer-runs \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

- 다음 예제에서는 최적화 실행 결과를 검색하는 방법과 여러 옵티마이저의 구성 세부 정보를 보여줍니다. 최대 20개의 옵티마이저를 지정할 수 있습니다.

```
aws glue batch-get-table-optimizer \
--entries '[{"catalogId":"123456789012", "databaseName":"iceberg_db",
 "tableName":"iceberg_table", "type":"compaction"}]'
```

API

- `GetTableOptimizer` 작업을 사용하여 옵티마이저의 마지막 실행 세부 정보를 검색합니다.
- `ListTableOptimizerRuns` 작업을 사용하여 특정 테이블에서 지정된 옵티마이저의 기록을 검색할 수 있습니다. API 호출 한 번으로 옵티마이저 20개를 지정할 수 있습니다.
- BatchGetTableOptimizer 작업을 사용하여 계정에서 여러 옵티마이저에 대한 구성 세부 정보를 검색합니다.

Amazon CloudWatch 지표 보기

테이블 옵티마어지를 성공적으로 실행한 후 서비스는 최적화 작업 성능에 대한 Amazon CloudWatch 지표를 생성합니다. CloudWatch 지표로 이동하여 지표, 모든 지표를 선택할 수 있습니다. 특정 네임스페이스(예: AWS Glue), 테이블 이름 또는 데이터베이스 이름을 기준으로 지표를 필터링할 수 있습니다.

자세한 내용은 Amazon CloudWatch 사용 설명서의 사용 가능한 지표 보기를 참조하세요.

압축

- 압축된 바이트 수
- 압축된 파일 수
- 작업에 할당된 DPU 수
- 작업 기간(시간)

스냅샷 보존

- 삭제된 데이터 파일 수
- 삭제된 매니페스트 파일 수
- 삭제된 매니페스트 목록 수
- 작업 기간(시간)

분리된 파일 삭제

- 삭제된 분리된 파일 수
- 작업 기간(시간)

옵티마이저 삭제

AWS CLI 또는 AWS API 작업을 사용하여 테이블의 옵티마이저 및 관련 메타데이터를 삭제할 수 있습니다.

다음 AWS CLI 명령을 실행하여 테이블의 최적화 기록을 삭제합니다. 카탈로그 ID, 데이터베이스 이름 및 테이블 이름과 함께 `type` 옵티마이저를 지정해야 합니다. 사용 가능한 값은 `compaction`, `retention` 및 `orphan_file_deletion`입니다.

```
aws glue delete-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

`DeleteTableOptimizer` 작업을 사용하여 테이블의 옵티마이저를 삭제합니다.

고려 사항 및 제한 사항

이 섹션에는 AWS Glue Data Catalog 내에서 테이블 옵티마이저를 사용할 때 고려할 사항이 포함되어 있습니다.

관리형 데이터 압축에 지원되는 형식 및 제한 사항

데이터 압축은 암호화된 테이블에서 데이터를 읽는 것을 비롯하여, 데이터 읽기 및 쓰기를 위한 다양한 데이터 형식 및 압축 형식을 지원합니다.

데이터 압축은 다음을 지원합니다.

- 암호화 - 데이터 압축은 기본 Amazon S3 암호화(SSE-S3) 및 서버 측 KMS 암호화(SSE-KMS)만 지원합니다.
- 압축 전략 - Binpack, Sort, Z-order 정렬
- 기본 데이터를 저장하는 Amazon S3 버킷이 다른 계정에 있는 경우 데이터 카탈로그가 있는 계정에서 압축을 실행할 수 있습니다. 이렇게 하려면 압축 역할에 Amazon S3 버킷에 대한 액세스 권한이 필요합니다.

데이터 압축은 현재 다음을 지원하지 않습니다.

- 교차 계정 테이블에서의 압축 - 교차 계정 테이블에서는 압축을 실행할 수 없습니다.
- 교차 리전 테이블에서의 압축 - 교차 리전 테이블에서는 압축을 실행할 수 없습니다.
- 리소스 링크에서 압축 활성화
- Amazon S3 Express One Zone 스토리지 클래스의 테이블 - S3 Express One Zone Iceberg 테이블에서는 압축을 실행할 수 없습니다.
- Z-order 압축 전략에서 지원하지 않는 데이터 유형:
  - 10진수
  - TimestampWithoutZone

스냅샷 보존 및 분리된 파일 삭제 최적화 프로그램에 대한 고려 사항

스냅샷 보존 및 분리된 파일 삭제 최적화 프로그램에 다음 고려 사항이 적용됩니다.

- 스냅샷 보존 및 분리된 파일 삭제 프로세스의 최대 삭제량은 실행당 1,000,000개의 파일입니다. 만료된 스냅샷을 삭제할 때 삭제할 수 있는 파일 수가 1,000,000개를 초과하면 해당 임계값을 초과하는 나머지 파일은 계속 분리된 파일로 테이블 스토리지에 남아 있게 됩니다.
- 스냅샷은 두 기준(보존할 최소 스냅샷 수 및 지정된 보존 기간)이 모두 충족되는 경우에만 스냅샷 보존 최적화 프로그램에서 보존합니다.
- 스냅샷 보존 최적화 프로그램은 Apache Iceberg에서 만료된 스냅샷 메타데이터를 삭제하여 만료된 스냅샷에 대한 시간 이동 쿼리를 방지하고 선택적으로 연결된 데이터 파일을 삭제합니다.
- 분리된 파일 삭제 최적화 프로그램은 생성 시간이 최적화 프로그램이 실행된 시점부터 분리된 파일 삭제 보존 기간보다 이전인 경우 Iceberg 메타데이터에서 더 이상 참조하지 않는 분리된 데이터 및 메타데이터 파일을 삭제합니다.
- Apache Iceberg는 특정 스냅샷 상태에 대한 명명된 포인터에 해당하는 브랜치와 태그를 통해 버전 제어를 용이하게 합니다. 각 브랜치와 태그는 각 수준에서 정의된 보존 정책에 따라 자체 독립 수명 주기를 따릅니다. AWS Glue Data Catalog 최적화 프로그램은 이러한 수명 주기 정책을 고려하여 지정된 보존 규칙을 준수하도록 합니다. 브랜치 및 태그 수준 보존 정책은 최적화 프로그램 구성보다 우선합니다.

  자세한 내용은 Apache Iceberg 설명서의 Branching and Tagging을 참조하세요.
- 스냅샷 보존 및 분리된 파일 삭제 최적화 프로그램은 구성된 파라미터에 따라 정리 대상인 파일을 삭제합니다. 적절한 버킷에 S3 버전 관리 및 수명 주기 정책을 구현하여 파일 삭제에 대한 제어를 강화합니다.

버전 관리 설정 및 수명 주기 규칙 생성에 대한 자세한 지침은 https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html 섹션을 참조하세요.

- 적절한 고립 파일을 결정하려면 제공된 테이블 위치와 하위 경로가 다른 테이블 또는 데이터 소스의 데이터와 겹치거나 포함되지 않도록 해야 합니다. 경로가 겹치면 의도하지 않은 파일 삭제로 인해 복구할 수 없는 데이터 손실 위험이 있습니다.

테이블 옵티마이저 지원 리전

AWS Glue Data Catalog의 테이블 최적화 기능(압축, 스냅샷 보존 및 분리된 파일 삭제)은 다음 AWS 리전에서 사용할 수 있습니다.

- 아시아 태평양(도쿄)
- 아시아 태평양(서울)
- 아시아 태평양(뭄바이)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(자카르타)
- 캐나다(중부)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(프랑크푸르트)
- 유럽(스톡홀름)
- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(오리건)
- 남아메리카(상파울루)

Iceberg 테이블의 쿼리 성능 최적화

Apache Iceberg는 대규모 분석 데이터세트를 위한 고성능 오픈 테이블 형식입니다. AWS Glue에서는 Iceberg 테이블의 각 열에 대한 고유 값의 수(NDV) 계산 및 업데이트를 지원합니다. 이러한 통계를 통해 대규모 데이터세트를 사용하는 데이터 엔지니어와 과학자의 쿼리 최적화, 데이터 관리 및 성능 효율성을 높일 수 있습니다.

AWS Glue에서는 Iceberg 테이블의 각 열에 있는 고유 값의 수를 추정하고 Iceberg 테이블 스냅샷과 연결된 Amazon S3의 Puffin 파일에 저장합니다. Puffin은 인덱스, 통계, 스케치와 같은 메타데이터를 저장하도록 설계된 Iceberg 파일 형식입니다. 스냅샷과 연결된 Puffin 파일에 스케치를 저장하면 트랜잭션 일관성을 유지하면서 NDV 통계를 최신으로 유지할 수 있습니다.

AWS Glue 콘솔 또는 AWS CLI를 사용하여 열 통계 생성 작업을 실행하도록 구성할 수 있습니다. 프로세스를 시작하면 백그라운드에서 AWS Glue가 Spark 작업을 시작하고 데이터 카탈로그의 AWS Glue 테이블 메타데이터를 업데이트합니다. AWS Glue 콘솔 또는 AWS CLI를 사용하거나 GetColumnStatisticsForTable API 작업을 직접적으로 호출하여 열 통계를 볼 수 있습니다.

> **Note**
>
> AWS Lake Formation 권한을 사용하여 테이블에 대한 액세스를 제어하는 경우 열 통계 작업에서 맡은 역할을 수행하려면 통계를 생성하기 위한 전체 테이블 액세스 권한이 필요합니다.

주제

- 열 통계 생성을 위한 사전 요구 사항
- Iceberg 테이블의 열 통계 생성
- 다음 사항도 참조하세요.

열 통계 생성을 위한 사전 요구 사항

Iceberg 테이블의 열 통계를 생성하거나 업데이트하기 위해 통계 생성 작업은 사용자를 대신하여 AWS Identity and Access Management(IAM) 역할을 수임합니다. 역할에 부여된 권한에 따라 열 통계 생성 작업은 Amazon S3 데이터 스토어에서 데이터를 읽을 수 있습니다.

열 통계 생성 작업을 구성하면 AWS Glue에서 `AWSGlueServiceRole` AWS 관리형 정책과 지정된 데이터 소스에 대한 필수 인라인 정책을 포함하는 역할 생성을 허용합니다.

열 통계 생성에 대한 기존 역할을 지정하는 경우 `AWSGlueServiceRole` 정책 또는 이에 상응하는 것(또는 이 정책의 범위가 축소된 버전)과 필수 인라인 정책이 포함되어 있는지 확인합니다.

필요한 권한에 대한 자세한 정보는 열 통계 생성을 위한 사전 요구 사항 단원을 참조하세요.

Iceberg 테이블의 열 통계 생성

AWS Glue 콘솔 또는 를 사용하여 데이터 카탈로그에서 통계를 생성하기 위한 일정을 구성 AWS CLI 하거나 StartColumnStatisticsTaskRun 작업을 실행하려면 다음 단계를 따르세요.

열 통계 생성

1. 에서 AWS Glue 콘솔에 로그인합니다https://console.aws.amazon.com/glue/.
2. 데이터 카탈로그에서 테이블을 선택합니다.
3. 목록에서 Iceberg 테이블을 선택합니다.
4. 작업 메뉴에서 열 통계, 요청 시 생성을 선택합니다.

   테이블 페이지 하단 섹션의 열 통계 탭에서 통계 생성 버튼을 선택해도 됩니다.
5. 통계 생성 페이지에서 통계 생성 세부 정보를 입력합니다. 일정에 따른 열 통계 생성 섹션의 6~11 단계에 따라 Iceberg 테이블에 대한 통계 생성 일정을 구성합니다.

   의 지침에 따라 온디맨드 방식으로 열 통계를 생성하도록 선택할 수도 있습니다. 온디맨드 열 통계 생성

   > **Note**
   >
   > 샘플링 옵션은 Iceberg 테이블에 사용할 수 없습니다.

   AWS Glue 는 Iceberg 테이블의 각 열에 대한 고유 값 수를 Amazon S3 위치에서 지정된 스냅샷 ID에 커밋된 새 Puffin 파일로 계산합니다.

다음 사항도 참조하세요.

- 열 통계 보기
- 컬럼 통계 태스크 실행 보기
- 열 통계 작업 실행 중지
- 컬럼 통계 삭제

데이터 카탈로그 관리

AWS Glue Data Catalog는 Amazon S3 데이터 세트의 구조 및 운영 메타데이터를 저장하는 중앙 메타데이터 리포지토리입니다. 데이터 카탈로그를 효과적으로 관리하는 것은 데이터 품질, 성능, 보안 및 거버넌스를 유지 관리하는 데 매우 중요합니다.

이러한 데이터 카탈로그 관리 모범 사례를 이해하고 적용하면 데이터 환경의 진화에 발맞춰 메타데이터가 정확하고, 뛰어난 성능을 유지하고, 안전하며, 잘 통제되도록 보장할 수 있습니다.

이 섹션에서는 데이터 카탈로그 관리의 다음과 같은 측면을 다룹니다.

- 테이블 스키마 및 파티션 업데이트: 데이터가 진화함에 따라 데이터 카탈로그에 정의된 테이블 스키마나 파티션 구조를 업데이트해야 할 수 있습니다. AWS Glue ETL을 사용하여 프로그래밍 방식으로 업데이트하는 방법에 대한 자세한 내용은 AWS Glue ETL 작업을 사용하여 데이터 카탈로그에서 스키마 업데이트 및 새 파티션 추가 섹션을 참조하세요.
- 열 통계 관리: 정확한 열 통계는 쿼리 계획을 최적화하고 성능을 개선하는 데 도움이 됩니다. 열 통계를 생성, 업데이트 및 관리하는 방법에 대한 자세한 정보는 열 통계를 사용한 쿼리 성능 최적화 섹션을 참조하세요.
- 데이터 카탈로그 암호화: 민감한 메타데이터를 보호하려면 AWS Key Management Service(AWS KMS)를 사용하여 데이터 카탈로그를 암호화할 수 있습니다. 이 섹션에서는 데이터 카탈로그에 대한 암호화를 활성화하고 관리하는 방법을 설명합니다.
- AWS Lake Formation으로 데이터 카탈로그 보호: Lake Formation은 데이터 레이크 보안 및 액세스 제어에 대한 포괄적인 접근 방식을 제공합니다. Lake Formation을 사용하면 데이터 카탈로그 및 기본 데이터에 대한 액세스를 보호하고 관리할 수 있습니다.

주제

- AWS Glue ETL 작업을 사용하여 데이터 카탈로그에서 스키마 업데이트 및 새 파티션 추가
- 열 통계를 사용한 쿼리 성능 최적화
- 데이터 카탈로그 암호화
- Lake Formation을 사용한 데이터 카탈로그 보호
- AWS Glue에서 AWS Glue Data Catalog 뷰 작업

AWS Glue ETL 작업을 사용하여 데이터 카탈로그에서 스키마 업데이트 및 새 파티션 추가

추출, 변환 및 로드(ETL) 작업은 대상 데이터 스토어에 새 테이블 파티션을 생성할 수 있습니다. 데이터 집합 스키마는 시간이 지남에 따라 진화하면서 AWS Glue Data Catalog 스키마에서 벗어날 수 있습니다. AWS Glue 이제 ETL 작업에서 여러 기능을 사용해 ETL 스크립트 내에서 Data Catalog의 스키마 및 파티션을 업데이트할 수 있습니다. 이러한 기능을 통해 크롤러를 다시 실행할 필요 없이 Data Catalog의 ETL 작업 결과를 볼 수 있습니다.

새 파티션

AWS Glue Data Catalog의 새 파티션을 보려면 다음 중 하나를 수행할 수 있습니다.

- 작업이 완료된 후 크롤러를 다시 실행하고, 크롤러가 완료되면 콘솔에서 새 파티션을 확인합니다.
- 작업이 완료된 후 크롤러를 다시 실행하지 않고 콘솔에서 바로 새 파티션을 확인합니다. 다음 예제와 같이 몇 줄의 코드를 ETL 스크립트에 추가하여 이 기능을 활성화할 수 있습니다. 이 코드는 새 파티션이 생성될 때 작업 실행 중에 Data Catalog를 업데이트하는 `enableUpdateCatalog` 인수를 사용합니다.

방법 1

옵션 인수에서 `enableUpdateCatalog` 및 `partitionKeys`를 전달합니다.

Python

```
additionalOptions = {"enableUpdateCatalog": True}
additionalOptions["partitionKeys"] = ["region", "year", "month", "day"]


sink = glueContext.write_dynamic_frame_from_catalog(frame=last_transform,
 database=<target_db_name>,

 table_name=<target_table_name>, transformation_ctx="write_sink",

 additional_options=additionalOptions)
```

Scala

```
val options = JsonOptions(Map(
    "path" -> <S3_output_path>,
    "partitionKeys" -> Seq("region", "year", "month", "day"),
    "enableUpdateCatalog" -> true))
val sink = glueContext.getCatalogSink(
    database = <target_db_name>,
    tableName = <target_table_name>,
    additionalOptions = options)sink.writeDynamicFrame(df)
```

방법 2

`getSink()`에서 `enableUpdateCatalog` 및 `partitionKeys`를 전달하고 `DataSink` 객체에서 `setCatalogInfo()`를 호출합니다.

Python

```
sink = glueContext.getSink(
    connection_type="s3",
    path="<S3_output_path>",
    enableUpdateCatalog=True,
    partitionKeys=["region", "year", "month", "day"])
sink.setFormat("json")
sink.setCatalogInfo(catalogDatabase=<target_db_name>,
 catalogTableName=<target_table_name>)
sink.writeFrame(last_transform)
```

Scala

```
val options = JsonOptions(
   Map("path" -> <S3_output_path>,
       "partitionKeys" -> Seq("region", "year", "month", "day"),
       "enableUpdateCatalog" -> true))
val sink = glueContext.getSink("s3", options).withFormat("json")
sink.setCatalogInfo(<target_db_name>, <target_table_name>)
sink.writeDynamicFrame(df)
```

이제 크롤러를 다시 실행할 필요 없이 Data Catalog에서 AWS Glue ETL 작업 자체를 사용해 새 카탈로그 테이블을 생성하고, 수정된 스키마로 기존 테이블을 업데이트하고, 새 테이블 파티션을 추가할 수 있습니다.

테이블 스키마 업데이트

Data Catalog 테이블의 스키마를 덮어쓰려면 다음 중 하나를 수행할 수 있습니다.

- 작업이 완료되면 크롤러를 다시 실행하고 크롤러가 테이블 정의도 업데이트하도록 구성되어 있는지 확인합니다. 크롤러가 완료되면 콘솔의 새 파티션과 스키마 업데이트를 확인합니다. 자세한 내용은 API를 사용해 크롤러 구성을 참조하십시오.
- 작업이 완료된 후 크롤러를 다시 실행하지 않고 콘솔에서 바로 수정된 스키마를 확인합니다. 다음 예제와 같이 몇 줄의 코드를 ETL 스크립트에 추가하여 이 기능을 활성화할 수 있습니다. 이 코드에서는 `enableUpdateCatalog`가 true로, `updateBehavior`가 `UPDATE_IN_DATABASE`로 설정되어 있는데, 이를 통해 작업 실행 중에 Data Catalog에서 스키마를 덮어쓰고 새 파티션을 추가하게 됩니다.

Python

```
additionalOptions = {
    "enableUpdateCatalog": True,
    "updateBehavior": "UPDATE_IN_DATABASE"}
additionalOptions["partitionKeys"] = ["partition_key0", "partition_key1"]

sink = glueContext.write_dynamic_frame_from_catalog(frame=last_transform,
 database=<dst_db_name>,
    table_name=<dst_tbl_name>, transformation_ctx="write_sink",
    additional_options=additionalOptions)
job.commit()
```

Scala

```
val options = JsonOptions(Map(
    "path" -> outputPath,
    "partitionKeys" -> Seq("partition_0", "partition_1"),
    "enableUpdateCatalog" -> true))
val sink = glueContext.getCatalogSink(database = nameSpace, tableName = tableName,
 additionalOptions = options)
sink.writeDynamicFrame(df)
```

또한 테이블 스키마를 덮어쓰지 않고 새 파티션을 추가하려는 경우에는 updateBehavior 값을 LOG로 설정하면 됩니다. updateBehavior의 기본값은 UPDATE_IN_DATABASE이므로 명시적으로 정의하지 않으면 테이블 스키마를 덮어씁니다.

enableUpdateCatalog가 true로 설정되어 있지 않으면 updateBehavior에서 선택한 옵션에 관계없이 ETL 작업이 Data Catalog의 테이블을 업데이트하지 않습니다.

새 테이블 생성

동일한 옵션을 사용해 Data Catalog에서 새 테이블을 생성할 수도 있습니다. setCatalogInfo를 사용해 데이터베이스 및 새 테이블 이름을 지정할 수 있습니다.

Python

```
sink = glueContext.getSink(connection_type="s3", path="s3://path/to/data",
    enableUpdateCatalog=True, updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=["partition_key0", "partition_key1"])
```

```
sink.setFormat("<format>")
sink.setCatalogInfo(catalogDatabase=<dst_db_name>, catalogTableName=<dst_tbl_name>)
sink.writeFrame(last_transform)
```

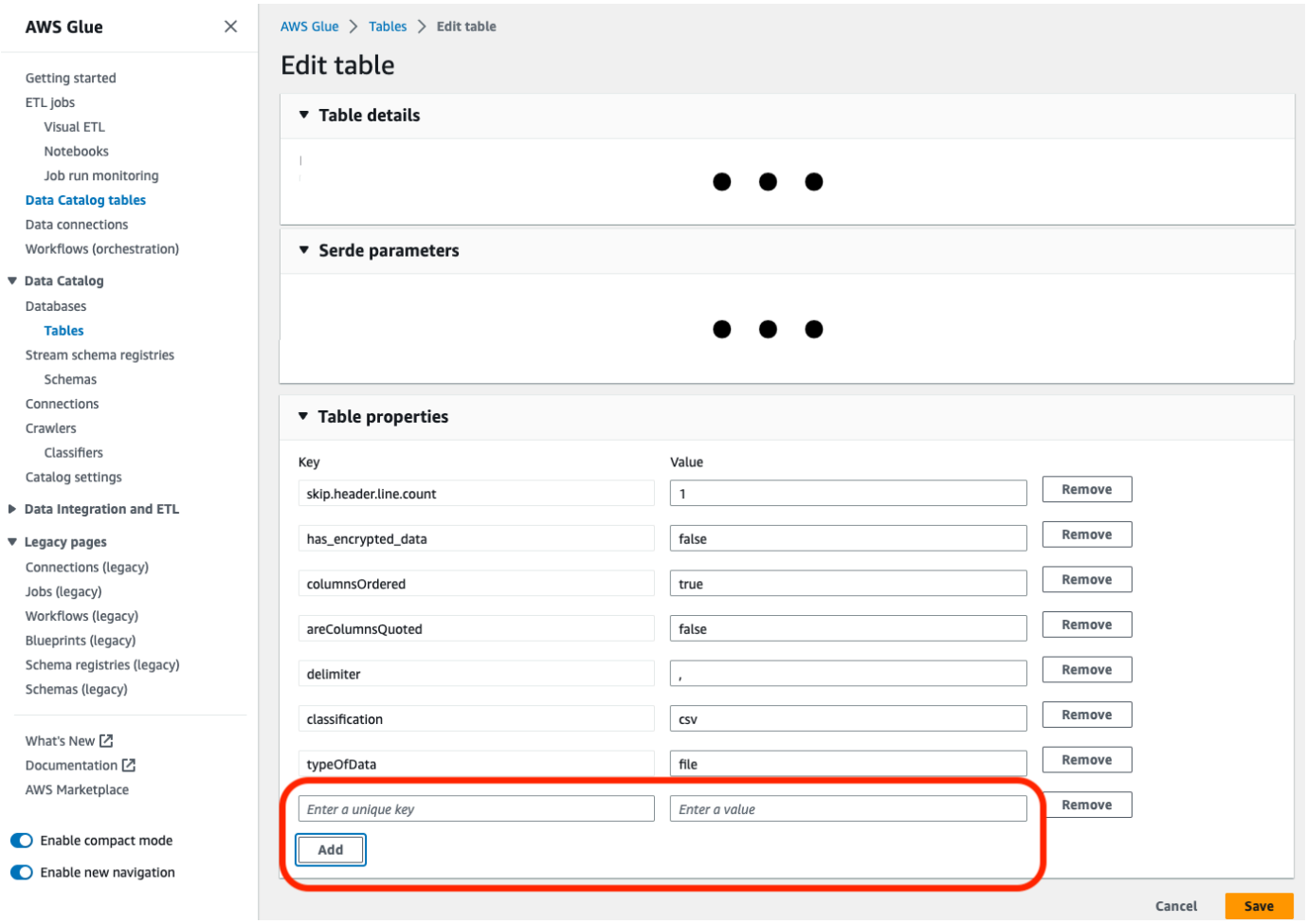
Scala

```
val options = JsonOptions(Map(
    "path" -> outputPath,
    "partitionKeys" -> Seq("<partition_1>", "<partition_2>"),
    "enableUpdateCatalog" -> true,
    "updateBehavior" -> "UPDATE_IN_DATABASE"))
val sink = glueContext.getSink(connectionType = "s3", connectionOptions =
 options).withFormat("<format>")
sink.setCatalogInfo(catalogDatabase = “<dst_db_name>”, catalogTableName =
 “<dst_tbl_name>”)
sink.writeDynamicFrame(df)
```

제한 사항

다음 제한 사항에 유의하십시오.

- Amazon Simple Storage Service(Amazon S3) 대상만 지원됩니다.
- `enableUpdateCatalog` 기능은 관리 테이블에서 지원되지 않습니다.
- `json`, `csv`, `avro` 및 `parquet` 형식만 지원됩니다.
- `parquet` 분류로 테이블을 생성하거나 업데이트하려면 Dynamic Frames에 대해 AWS Glue 최적화 parquet 라이터를 사용해야 합니다. 이는 다음 중 하나를 통해 수행할 수 있습니다.
  - 카탈로그의 기존 테이블을 `parquet` 분류로 업데이트하는 경우 테이블을 업데이트하기 전에 테이블에서 `"useGlueParquetWriter"` 테이블 속성이 `true`로 설정되어 있어야 합니다. AWS Glue API/SDK, 콘솔 또는 Athena DDL 문을 통해 이 속성을 설정할 수 있습니다.



카탈로그 테이블 속성이 설정되면 다음 코드 스니펫을 사용하여 카탈로그 테이블을 새 데이터로 업데이트할 수 있습니다.

```
glueContext.write_dynamic_frame.from_catalog(
    frame=frameToWrite,
    database="dbName",
    table_name="tableName",
    additional_options={
        "enableUpdateCatalog": True,
        "updateBehavior": "UPDATE_IN_DATABASE"
    }
)
```

- 테이블이 카탈로그에 아직 없는 경우 `connection_type="s3"`를 통해 스크립트에서 `getSink()` 메서드를 사용하여 Amazon S3에 데이터를 쓰는 작업과 함께 테이블과 해당 파티션을 카탈로그에 추가할 수 있습니다. 워크플로에 적절한 `partitionKeys` 및 `compression`을 제공합니다.

```
s3sink = glueContext.getSink(
    path="s3://bucket/folder/",
    connection_type="s3",
    updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=[],
    compression="snappy",
    enableUpdateCatalog=True
)

s3sink.setCatalogInfo(
    catalogDatabase="dbName", catalogTableName="tableName"
)

s3sink.setFormat("parquet", useGlueParquetWriter=True)
s3sink.writeFrame(frameToWrite)
```

  - 이 `glueparquet` 형식 값은 AWS Glue parquet 작성기를 활성화하는 기존 방법입니다.
- `updateBehavior`를 `LOG`로 설정하면 `DynamicFrame` 스키마가 Data Catalog 테이블의 스키마에 정의된 열의 하위 집합과 동일하거나 해당 집합을 포함하고 있는 경우에만 새 파티션이 추가됩니다.
- 분할되지 않은 테이블에는 스키마 업데이트가 지원되지 않습니다("partitionKeys" 옵션 사용 안 함).
- partitionKeys는 ETL 스크립트에 전달된 파라미터와 Data Catalog 테이블 스키마의 partitionKeys 간에 동일한 순서로 동일해야 합니다.
- 이 기능은 현재 업데이트 스키마가 중첩된 테이블 업데이트/생성을 아직 지원하지 않습니다(예: 구조체 내부 배열).

자세한 내용은 the section called "AWS Glue for Spark" 단원을 참조하십시오.

ETL 작업의 MondoDB 연결 작업

MongoDB 연결을 생성한 다음 해당 연결을 AWS Glue 작업에서 사용하면 됩니다. 자세한 내용은 AWS Glue 프로그래밍 안내서의 the section called "MongoDB 연결"를 참조하십시오. 연결 `url`, `username` 및 `password`는 MongoDB 연결에 저장됩니다. 다른 옵션은 `glueContext.getCatalogSource`의 `additionalOptions` 파라미터를 사용하여 ETL 작업 스크립트에서 지정하면 됩니다. 이외 옵션으로 다음과 같은 항목이 포함됩니다.

- `database`: (필수 사항) 읽을 MongoDB 데이터베이스입니다.
- `collection`: (필수 사항) 읽을 MongoDB 컬렉션입니다.

ETL 작업 스크립트 내에 `database` 및 `collection` 정보를 배치하면 여러 작업에 같은 조건을 사용할 수 있습니다.

1. MongoDB 데이터 원본에 대해 AWS Glue Data Catalog 연결을 생성합니다. 연결 파라미터에 대한 설명은 "connectionType": "mongodb"를 참조하세요. 연결은 콘솔, API 또는 CLI를 사용해 생성하면 됩니다.
2. AWS Glue Data Catalog에 데이터베이스를 생성하여 여기에 MongoDB 데이터의 테이블 정의를 저장합니다. 자세한 정보는 데이터베이스 생성을 참조하세요.
3. MongoDB로 연결되는 연결의 정보를 사용해 MongoDB의 데이터를 크롤링하는 크롤러를 생성합니다. 이 크롤러가 AWS Glue Data Catalog에 테이블을 생성하며 이것이 작업에서 사용하는 MongoDB 데이터베이스 내 테이블을 설명합니다. 자세한 정보는 크롤러를 사용하여 데이터 카탈로그 채우기 을 참조하세요.
4. 사용자 정의 스크립트를 사용하여 작업을 생성합니다. 작업은 콘솔, API 또는 CLI를 사용해 생성하면 됩니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.
5. 작업의 데이터 대상을 선택합니다. 데이터 대상을 나타내는 테이블은 Data Catalog에서 정의할 수 있고, 아니면 작업이 실행될 때 대상 테이블을 생성할 수도 있습니다. 작업을 작성하면 대상 위치를 선택합니다. 대상에 연결이 요구되면, 연결도 작업에서 참조됩니다. 작업이 여러 데이터 대상이 필요하다면 나중에 스크립트를 편집하여 추가할 수 있습니다.
6. 작업과 생성된 스크립트의 인수를 제공하여 작업 처리 환경을 사용자 지정합니다.

   다음은 Data Catalog에서 정의한 테이블 구조를 기반으로 한 MongoDB 데이터베이스에서 `DynamicFrame`을 생성하는 예입니다. 이 코드는 `additionalOptions`를 사용하여 추가적인 데이터 원본 정보를 제공합니다.

   Scala

   ```
   val resultFrame: DynamicFrame = glueContext.getCatalogSource(
           database = catalogDB,
           tableName = catalogTable,
           additionalOptions = JsonOptions(Map("database" -> DATABASE_NAME,
                   "collection" -> COLLECTION_NAME))
         ).getDynamicFrame()
   ```

   Python

   ```
   glue_context.create_dynamic_frame_from_catalog(
           database = catalogDB,
           table_name = catalogTable,
   ```

```
        additional_options = {"database":"database_name",
            "collection":"collection_name"})
```

7. 작업을 실행합니다. 온디맨드여도 되고, 트리거를 통해도 됩니다.

열 통계를 사용한 쿼리 성능 최적화

추가 데이터 파이프라인을 설정하지 않고도 Parquet, ORC, JSON, ION, CSV 및 XML과 같은 데이터 형식의 AWS Glue Data Catalog 테이블에 대한 열 수준 통계를 계산할 수 있습니다. 열 통계는 열 내 값에 대한 통찰력을 얻어 데이터 프로필을 이해하는 데 도움이 됩니다.

Data Catalog는 최소값, 최대값, 총 null 값, 총 고유 값, 값의 평균 길이, 실제 값의 총 발생 횟수 등과 같은 열 값에 대한 통계 생성을 지원합니다. Amazon Redshift 및 Amazon Athena와 같은 AWS 분석 서비스에서는 이러한 열 통계를 사용하여 쿼리 실행 계획을 생성하고 쿼리 성능을 향상시키는 최적의 계획을 선택할 수 있습니다.

다음은 열 통계 생성에 대한 세 가지 시나리오입니다.

자동

AWS Glue는 카탈로그 수준에서 자동 열 통계 생성을 지원하므로 AWS Glue Data Catalog의 새 테이블에 대한 통계를 자동으로 생성할 수 있습니다.

예약됨

AWS Glue는 열 통계 생성 예약을 지원하며, 반복 일정에 따라 자동으로 실행할 수 있습니다.

예약된 통계 계산을 통해 열 통계 작업은 최소, 최대, 평균과 같은 전체 테이블 수준 통계를 새 통계로 업데이트하고 쿼리 엔진에 쿼리 실행을 최적화하기 위한 정확한 최신 통계를 제공합니다.

온디맨드

필요할 때마다 온디맨드 방식으로 열 통계를 생성하려면 이 옵션을 사용합니다. 이 옵션은 임시 분석이나 통계를 즉시 계산해야 하는 경우에 유용합니다.

AWS Glue 콘솔, AWS CLI 및 AWS Glue API 작업을 사용하여 열 통계 생성 작업을 실행하도록 구성할 수 있습니다. 프로세스를 시작하면 백그라운드에서 AWS Glue가 Spark 작업을 시작하고 데이터 카탈로그의 AWS Glue 테이블 메타데이터를 업데이트합니다. AWS Glue 콘솔 또는 AWS CLI를 사용하거나 GetColumnStatisticsForTable API 작업을 직접적으로 호출하여 열 통계를 볼 수 있습니다.

> **Note**
>
> Lake Formation 권한을 사용하여 테이블에 대한 액세스를 제어하는 경우 열 통계 작업에서 맡은 역할을 수행하려면 통계를 생성하기 위한 전체 테이블 액세스 권한이 필요합니다.

주제

- 열 통계 생성을 위한 사전 요구 사항
- 자동 열 통계 생성
- 일정에 따른 열 통계 생성
- 온디맨드 열 통계 생성
- 열 통계 보기
- 컬럼 통계 태스크 실행 보기
- 열 통계 작업 실행 중지
- 컬럼 통계 삭제
- 고려 사항 및 제한

열 통계 생성을 위한 사전 요구 사항

열 통계를 생성하거나 업데이트하기 위해 통계 생성 작업은 사용자를 대신하여 AWS Identity and Access Management(IAM) 역할을 수임합니다. 역할에 부여된 권한에 따라 열 통계 생성 작업은 Amazon S3 데이터 스토어에서 데이터를 읽을 수 있습니다.

열 통계 생성 작업을 구성하면 AWS Glue에서 `AWSGlueServiceRole` AWS 관리형 정책과 지정된 데이터 소스에 대한 필수 인라인 정책을 포함하는 역할 생성을 허용합니다.

열 통계 생성에 대한 기존 역할을 지정하는 경우 `AWSGlueServiceRole` 정책 또는 이에 상응하는 것(또는 이 정책의 범위가 축소된 버전)과 필수 인라인 정책이 포함되어 있는지 확인합니다. 새 IAM 역할을 생성하려면 다음 단계를 수행합니다.

> **Note**
>
> Lake Formation에서 관리하는 테이블에 대한 통계를 생성하려면 통계 생성에 사용되는 IAM 역할에 전체 테이블 액세스 권한이 필요합니다.

열 통계 생성 작업을 구성하면 AWS Glue에서 `AWSGlueServiceRole` AWS 관리형 정책과 지정된 데이터 소스에 대한 필수 인라인 정책을 포함하는 역할 생성을 허용합니다. 역할을 생성하고 아래 정책에 나열된 권한에 연결한 후 열 통계 생성 작업에 추가할 수도 있습니다.

열 통계 생성을 위한 IAM 역할을 생성하는 방법

1. IAM 역할을 생성하려면 AWS Glue의 IAM 역할 생성을 참조하십시오.
2. 기존 역할을 업데이트하려면 IAM 콘솔에서 열 통계 생성 프로세스에 사용되는 IAM 역할로 이동합니다.
3. 권한 추가 탭에서 정책 연결을 선택합니다. 새로 열린 브라우저 창에서 `AWSGlueServiceRole` AWS 관리형 정책을 선택합니다.
4. Amazon S3 데이터 위치에서 데이터를 읽을 수 있는 권한도 포함해야 합니다.

   권한 추가 섹션에서 정책 생성을 선택합니다. 새로 열린 브라우저 창에서 역할에 사용할 새 정책을 생성합니다.
5. 정책 생성 페이지에서 JSON 탭을 선택합니다. 다음 `JSON` 코드를 정책 편집기 필드에 복사합니다.

> **Note**
>
> 다음 정책에서는 계정 ID를 유효한 AWS 계정로 바꾸고, `region`을 테이블의 리전으로, `bucket-name`을 Amazon S3 버킷 이름으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>/*",
                "arn:aws:s3:::<bucket-name>"
            ]
        }
     ]
```

```
}
```

6. (선택 사항) Lake Formation 권한을 사용하여 데이터에 대한 액세스를 제공하는 경우 IAM 역할에 `lakeformation:GetDataAccess` 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "LakeFormationDataAccess",
            "Effect": "Allow",
            "Action": "lakeformation:GetDataAccess",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Amazon S3 데이터 위치가 Lake Formation에 등록되어 있고 열 통계 생성 작업이 맡는 IAM 역할에 테이블에 대한 `IAM_ALLOWED_PRINCIPALS` 그룹 권한이 부여되지 않은 경우, 이 역할에는 Lake Formation `ALTER`와 테이블에 대한 `DESCRIBE` 권한이 필요합니다. Amazon S3 버킷을 등록하는 데 사용되는 역할에는 테이블에 대한 Lake Formation `INSERT` 및 `DELETE` 권한이 필요합니다.

Amazon S3 데이터 위치가 Lake Formation에 등록되지 않았고 IAM 역할에 테이블에 대한 `IAM_ALLOWED_PRINCIPALS` 그룹 권한이 부여되지 않은 경우, 해당 역할에는 테이블에 대한 Lake Formation `ALTER`, `DESCRIBE`, `INSERT`, `DELETE` 권한이 필요합니다.

7. 카탈로그 수준 `Automatic statistics generation` 옵션을 활성화한 경우 IAM 역할에는 기본 Data Catalog에 대한 `glue:UpdateCatalog` 권한 또는 Lake Formation `ALTER CATALOG` 권한이 있어야 합니다. `GetCatalog` 작업을 사용하여 카탈로그 속성을 확인할 수 있습니다.

8. (선택 사항) 암호화된 Amazon CloudWatch Logs을 기록하는 열 통계 생성 작업에는 키 정책 내 다음 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "CWLogsKmsPermissions",
        "Effect":  "Allow",
```

```
        "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:AssociateKmsKey"
        ],
        "Resource": [
            "arn:aws:logs:<region>:111122223333:log-group:/aws-glue:*"
        ]
    },
    {
        "Sid": "KmsPermissions",
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt",
            "kms:Encrypt"
        ],
        "Resource": [
        "arn:aws:kms:<region>:111122223333:key/"arn of key used for ETL cloudwatch
 encryption"
        ],
        "Condition": {
            "StringEquals": {
                "kms:ViaService": ["glue.<region>.amazonaws.com"]
            }
        }
    }
    ]
}
```

9. 열 통계를 실행하는 데 사용하는 역할에는 해당 역할에 대한 `iam:PassRole` 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": [
```

```
            "arn:aws:iam::111122223333:role/<columnstats-role-name>"
        ]
    }]
}
```

10. 열 통계 생성을 위한 IAM 역할을 생성할 때 해당 역할에는 서비스가 역할을 수임할 수 있도록 하는 다음과 같은 신뢰 정책도 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "TrustPolicy",
        "Effect": "Allow",
        "Principal": {
          "Service": "glue.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
      }
   ]
}
```

자동 열 통계 생성

열 통계 자동 생성을 사용하면 AWS Glue Data Catalog의 새 테이블에 대한 통계 생성을 예약하고 자동으로 계산할 수 있습니다. 자동 통계 생성을 활성화하면 Data Catalog는 개별 버킷 경로와 함께 Parquet, JSON, CSV, XML, ORC, ION, Apache Iceberg 등 특정 데이터 형식의 새 테이블을 검색합니다. 일회성 카탈로그 구성을 사용하면 Data Catalog가 이러한 테이블에 대한 통계를 생성합니다.

데이터 레이크 관리자는 Lake Formation 콘솔에서 기본 카탈로그를 선택하고 `Optimization configuration` 옵션을 사용하여 테이블 통계를 활성화하여 통계 생성을 구성할 수 있습니다. Data Catalog에서 새 테이블을 생성하거나 기존 테이블을 업데이트하면 Data Catalog는 Apache Iceberg 테이블의 고유 값(NDV) 수와 지원되는 다른 파일 형식의 null 수, 최댓값, 최솟값 및 평균 길이와 같은 추가 통계를 매주 수집합니다.

테이블 수준에서 통계 생성을 구성했거나 이전에 테이블에 대한 통계 생성 설정을 삭제한 경우 이러한 테이블별 설정이 자동 열 통계 생성에 대한 기본 카탈로그 설정보다 우선합니다.

자동 통계 생성 작업은 테이블의 레코드 중 50%를 분석하여 통계를 계산합니다. 자동 열 통계 생성을 통해, Amazon Athena 및 Amazon Redshift Spectrum과 같은 쿼리 엔진에서 쿼리 성능을 높이고 잠재

적 비용을 절감하는 데 사용할 수 있는 주간 지표가 Data Catalog에서 유지 관리됩니다. AWS Glue API 또는 콘솔을 사용하여 수작업 없이 자동화된 프로세스를 통한 통계 생성을 예약할 수 있습니다.

주제

- 카탈로그 수준 자동 통계 생성 활성화
- 자동 테이블 수준 설정 보기
- 카탈로그 수준 열 통계 생성 비활성화

카탈로그 수준 자동 통계 생성 활성화

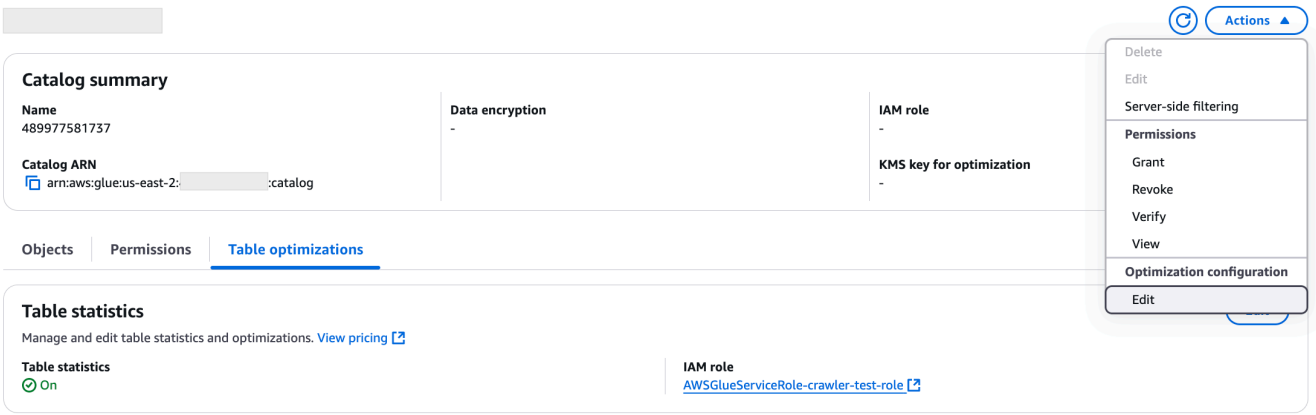
Data Catalog에서 모든 새 Apache Iceberg 테이블 및 비OTF 테이블(Parquet, JSON, CSV, XML, ORC, ION) 형식의 테이블에 대해 자동 열 통계 생성을 활성화할 수 있습니다. 테이블을 생성한 후 열 통계 설정을 수동으로 명시적으로 업데이트할 수도 있습니다.

카탈로그 수준을 활성화하도록 Data Catalog 설정을 업데이트하려면 사용된 IAM 역할에 루트 카탈로그에 대한 `glue:UpdateCatalog` 권한 또는 AWS Lake Formation `ALTER CATALOG` 권한이 있어야 합니다. `GetCatalog` API를 사용하여 카탈로그 속성을 확인할 수 있습니다.

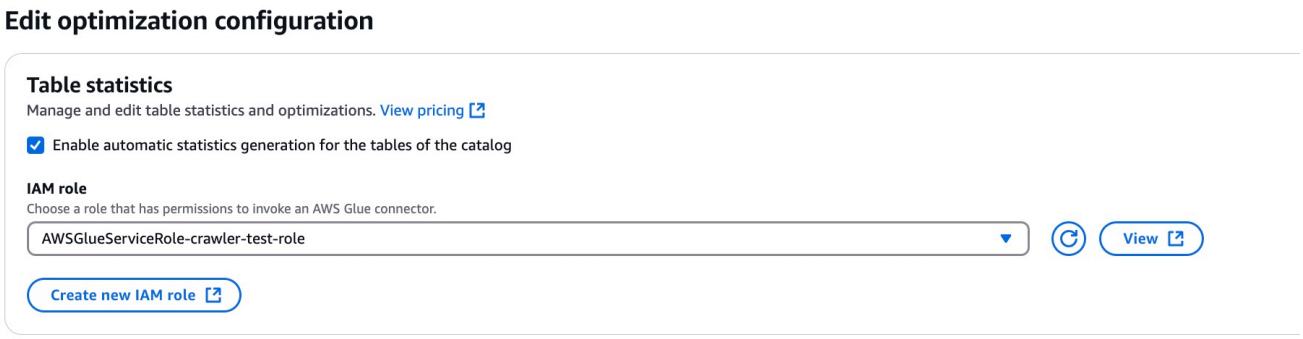
AWS Management Console

계정 수준에서 자동 열 통계 생성을 활성화하려면

1. Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)을 엽니다.
2. 왼쪽 탐색 모음에서 카탈로그를 선택합니다.
3. 카탈로그 요약 페이지의 최적화 구성에서 편집을 선택합니다.



4. 테이블 최적화 구성 페이지에서 카탈로그의 테이블에 대해 자동 통계 생성 활성화 옵션을 선택합니다.



5. 기존 IAM 역할을 선택하거나 열 통계 작업을 실행하는 데 필요한 권한이 있는 새 역할을 생성합니다.
6. 제출을 선택합니다.

AWS CLI

AWS CLI를 통해 카탈로그 수준 통계 수집을 활성화할 수도 있습니다. AWS CLI를 사용하여 테이블 수준 통계 수집을 구성하려면 다음 명령을 실행합니다.

```
aws glue update-catalog --cli-input-json '{
    "name": "123456789012",
    "catalogInput": {
        "description": "Updating root catalog with role arn",
        "catalogProperties": {
            "customProperties": {
                "ColumnStatistics.RoleArn": "arn:aws:iam::"123456789012":role/
service-role/AWSGlueServiceRole",
                "ColumnStatistics.Enabled": "true"
            }
        }
    }
}'
```

위의 명령은 카탈로그 수준 통계 생성을 위해 다음과 같은 키-값 페어를 포함하는 `CatalogProperties` 구조를 처리하는 AWS Glue의 `UpdateCatalog` 작업을 호출합니다.

- ColumnStatistics.RoleArn – 카탈로그 수준 통계 생성에서 트리거되는 모든 작업에 사용할 IAM 역할 ARN
- ColumnStatistics.Enabled – 카탈로그 수준 설정의 활성화 여부를 나타내는 부울

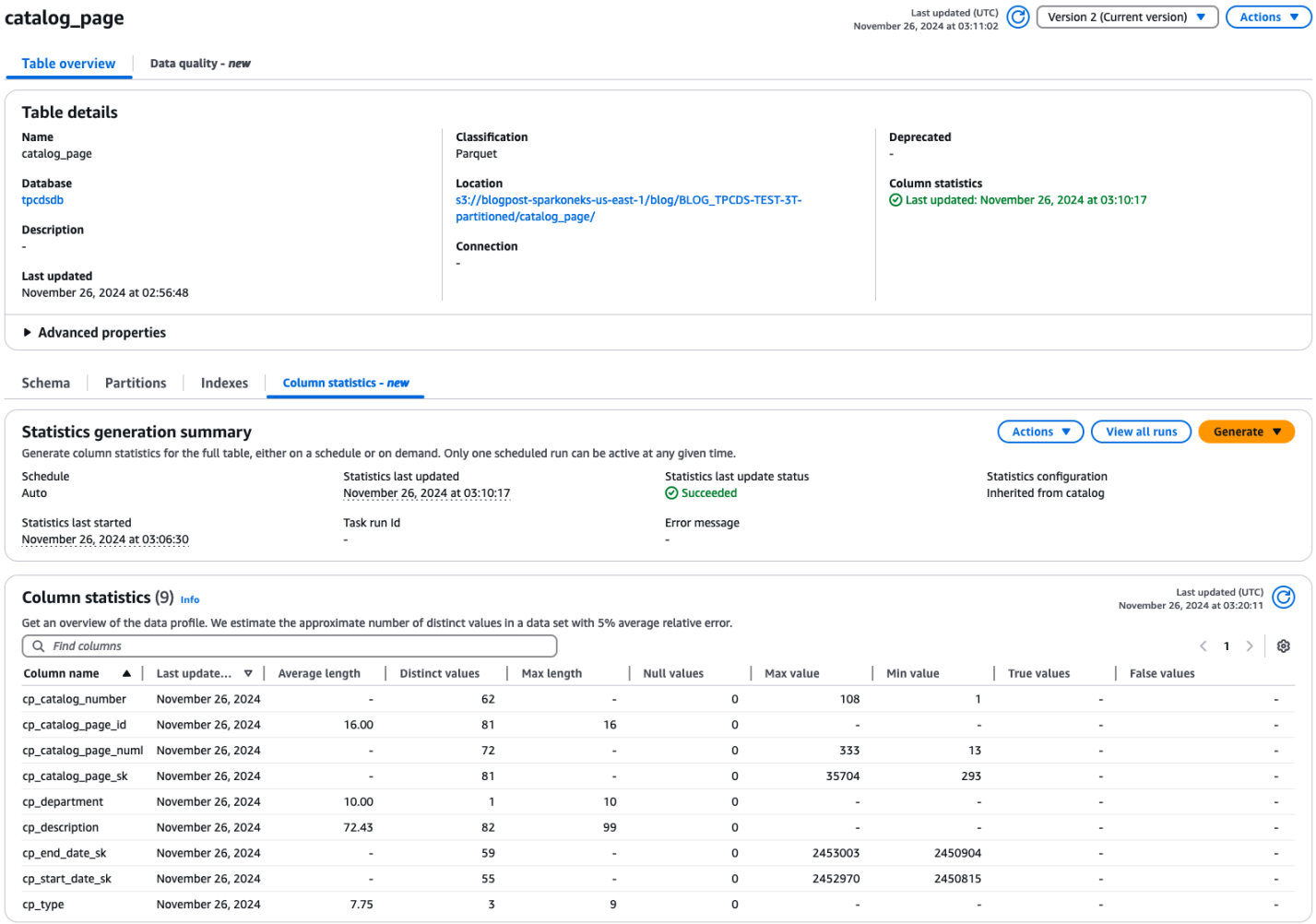
자동 테이블 수준 설정 보기

카탈로그 수준 통계 수집이 활성화되면, AWS Management Console, SDK 또는 AWS Glue 크롤러에서 `CreateTable` 또는 `UpdateTable` API를 통해 Apache Hive 테이블 또는 Apache Iceberg 테이블이 생성되거나 업데이트될 때마다 해당 테이블에 대해 동일한 테이블 수준 설정이 생성됩니다.

자동 통계 생성이 활성화된 테이블은 다음 속성 중 하나를 따라야 합니다.

- org.apache.hadoop으로 시작하고 `TableType`이 `EXTERNAL_TABLE`인 `InputSerdeLibrary` 사용
- `com.amazon.ion`으로 시작하고 `TableType`이 `EXTERNAL_TABLE`인 `InputSerdeLibrary` 사용
- table_type 포함: 파라미터 구조에 'ICEBERG' 포함.

테이블을 생성하거나 업데이트한 후 테이블 세부 정보에서 통계 생성을 확인할 수 있습니다. `Statistics generation summary`의 경우 `Schedule` 속성 세트가 `AUTO`로 설정되어 있고 `Statistics configuration` 값이 `Inherited from catalog`입니다. 다음 설정의 테이블 설정은 Glue에 의해 내부적으로 자동으로 트리거됩니다.



카탈로그 수준 열 통계 생성 비활성화

AWS Lake Formation 콘솔, `glue:UpdateCatalogSettings` API 또는 `glue:DeleteColumnStatisticsTaskSettings` API를 사용하여 새 테이블에 대한 자동 열 통계 생성을 비활성화할 수 있습니다.

계정 수준에서 자동 열 통계 생성을 비활성화하려면

1. Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)을 엽니다.
2. 왼쪽 탐색 모음에서 카탈로그를 선택합니다.
3. 카탈로그 요약 페이지의 최적화 구성에서 편집을 선택합니다.
4. 테이블 최적화 구성 페이지에서 카탈로그의 테이블에 대해 자동 통계 생성 활성화 옵션을 선택 취소합니다.
5. 제출을 선택합니다.

일정에 따른 열 통계 생성

AWS Glue 콘솔, AWS CLI또는 CreateColumnStatisticsTaskSettings 작업을 사용하여 에서 AWS Glue Data Catalog 열 통계를 생성하기 위한 일정을 구성하려면 다음 단계를 따르세요.

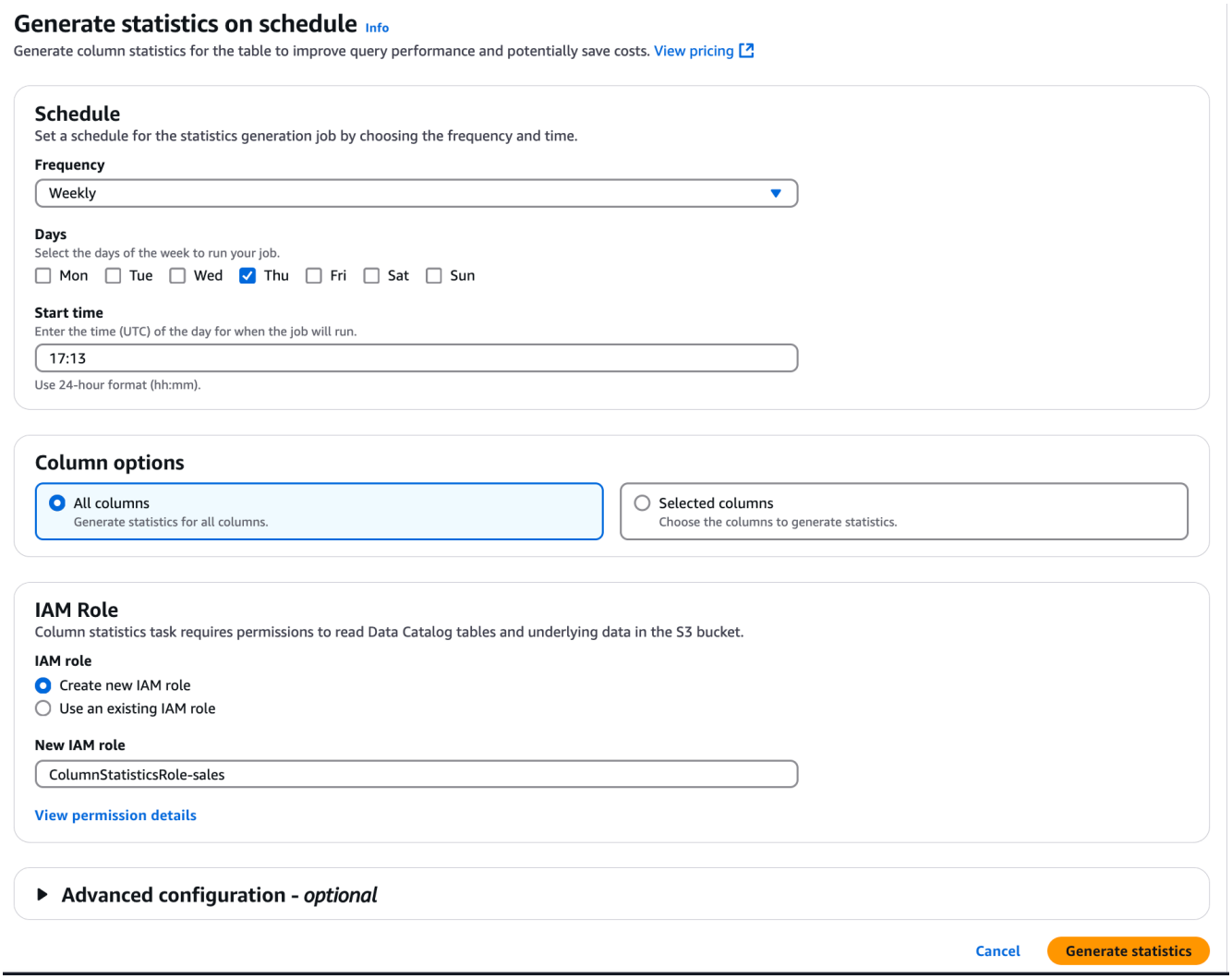
Console

콘솔을 사용하여 열 통계를 생성하는 방법

1. 에서 AWS Glue 콘솔에 로그인합니다https://console.aws.amazon.com/glue/.
2. 데이터 카탈로그 테이블을 선택합니다.
3. 목록에서 테이블을 선택합니다.
4. 테이블 페이지의 하단 섹션에서 열 통계 탭을 선택합니다.
5. 작업 의 열 통계에서 일정에 따라 생성을 선택할 수도 있습니다.
6. 일정에 대한 통계 생성 페이지에서 빈도 및 시작 시간을 선택하여 열 통계 작업을 실행하기 위한 반복 일정을 구성합니다. 시간별, 일별, 주별 빈도를 선택하거나 cron 표현식을 정의하여 일정을 지정할 수 있습니다.

   cron 표현식은 일정 패턴을 나타내는 문자열로, 공백으로 구분된 6개의 필드로 구성됩니다. * * * * * <minute> <hour> <day of month> <day of week> <year> 예를 들어, 매일 자정에 작업을 실행하려면 cron 표현식은 0 0 * * ? *

   자세한 내용은 cron 표현식 섹션을 참조하세요.



7. 그런 다음 열 옵션을 선택하여 통계를 생성합니다.

   - 모든 열 - 표의 모든 열에 대한 통계를 생성하려면 이 옵션을 선택합니다.
   - 선택한 열 - 특정 열에 대한 통계를 생성하려면 이 옵션을 선택합니다. 드롭다운 목록에서 를 선택합니다.

8. IAM 역할을 선택하거나 통계를 생성할 수 있는 권한이 있는 기존 역할을 생성합니다. 이 역할을 AWS Glue 가정하여 열 통계를 생성합니다.

   더 빠른 접근 방식은 AWS Glue 콘솔이 역할을 생성하도록 하는 것입니다. 생성하는 역할은 특히 열 통계를 생성하기 위한 것이며 관리`AWSGlueServiceRole` AWS 형 정책과 지정된 데이터 소스에 필요한 인라인 정책을 포함합니다.

   열 통계를 생성하기 위해 기존 역할을 지정하는 경우 정책 또는 이에 상응하는 정책(또는 이 `AWSGlueServiceRole` 정책의 범위 축소 버전)과 필요한 인라인 정책을 포함해야 합니다.

9. (선택 사항) 다음으로 로그에 대해 저장 중 암호화를 활성화하는 보안 구성을 선택합니다.
10. (선택 사항) 테이블에서 특정 백분율의 행만 표시하여 통계를 생성하여 샘플 크기를 선택할 수 있습니다. 기본값은 모든 행입니다. 위쪽 및 아래쪽 화살표를 사용하여 백분율 값을 늘리거나 줄입니다.

    정확한 통계를 계산하려면 표에 모든 행을 포함하는 것이 좋습니다. 대략적인 값이 허용되는 경우에만 샘플 행을 사용하여 열 통계를 생성하십시오.
11. 통계 생성을 선택하여 열 통계 생성 작업을 실행합니다.

AWS CLI

다음 AWS CLI 예제를 사용하여 열 통계 생성 일정을 생성할 수 있습니다. 데이터베이스 이름, 테이블 이름 및 역할은 필수 파라미터이며 선택적 파라미터는 일정 column-name-list, 카탈로그 ID, 샘플 크기 및 보안 구성입니다.

```
aws glue create-column-statistics-task-settings \
 --database-name 'database_name' \
 --table-name table_name \
 --role 'arn:aws:iam::123456789012:role/stats-role' \
 --schedule 'cron(0 0-5 14 * * ?)' \
 --column-name-list 'col-1' \
 --catalog-id '123456789012' \
 --sample-size '10.0 ' \
 --security-configuration 'test-security'
```

StartColumnStatisticsTaskRun 작업을 호출하여 열 통계를 생성할 수도 있습니다.

열 통계 생성 일정 관리

에서 열 통계 생성에 대한 일정 업데이트, 시작, 중지 및 삭제와 같은 예약 작업을 관리할 수 있습니다 AWS Glue. 콘솔 AWS CLI또는 AWS Glue 열 통계 API 작업을 사용하여 AWS Glue 이러한 작업을 수행할 수 있습니다.

주제

- 열 통계 생성 일정 업데이트
- 열 통계 생성을 위한 일정 중지
- 열 통계 생성 일정 재개

- 열 통계 생성 일정 삭제

열 통계 생성 일정 업데이트

일정이 생성된 후 열 통계 생성 작업을 트리거하도록 일정을 업데이트할 수 있습니다. AWS Glue 콘솔을 사용하거나 UpdateColumnStatisticsTaskSettings 작업을 AWS CLI실행하여 테이블의 일정을 업데이트할 수 있습니다. 일정 유형(온디맨드 또는 예약됨) 및 기타 선택적 파라미터와 같은 기존 일정의 파라미터를 수정할 수 있습니다.

AWS Management Console

열 통계 생성 작업의 설정을 업데이트하려면

1. 에서 AWS Glue 콘솔에 로그인합니다https://console.aws.amazon.com/glue/.
2. 테이블 목록에서 업데이트할 테이블을 선택합니다.
3. 테이블 세부 정보 페이지의 하단 섹션에서 열 통계 를 선택합니다.
4. 작업에서 편집을 선택하여 일정을 업데이트합니다.
5. 일정을 원하는 대로 변경하고 저장을 선택합니다.

AWS CLI

콘솔에서 AWS Glue의 통계 생성 기능을 사용하지 않는 경우 `update-column-statistics-task-settings` 명령을 사용하여 일정을 수동으로 업데이트할 수 있습니다. 다음 예에서는 AWS CLI를 이용하여 열 통계를 업데이트하는 방법을 보여 줍니다.

```
aws glue update-column-statistics-task-settings \
 --database-name 'database_name' \
 --table-name 'table_name' \
 --role arn:aws:iam::123456789012:role/stats_role \
 --schedule 'cron(0 0-5 16 * * ?)' \
 --column-name-list 'col-1' \
 --sample-size '20.0' \
 --catalog-id '123456789012'\
 --security-configuration 'test-security'
```

열 통계 생성을 위한 일정 중지

증분 통계가 더 이상 필요하지 않은 경우 예약된 생성을 중지하여 리소스와 비용을 절감할 수 있습니다. 일정을 일시 중지해도 이전에 생성된 통계에는 영향을 주지 않습니다. 편한 시간에 일정을 재개할 수 있습니다.

AWS Management Console

열 통계 생성 작업의 일정을 중지하려면

1. AWS Glue 콘솔의 데이터 카탈로그에서 테이블을 선택합니다.
2. 열 통계가 있는 테이블을 선택합니다.
3. 테이블 세부정보 페이지에서 열 통계를 선택합니다.
4. 작업에서 예약된 생성 , 일시 중지 를 선택합니다.
5. 일시 중지를 선택하여 확인합니다.

AWS CLI

를 사용하여 열 통계 작업 실행 일정을 중지하려면 다음 명령을 사용할 AWS CLI수 있습니다.

```
aws glue stop-column-statistics-task-run-schedule \
 --database-name ''database_name' \
 --table-name 'table_name'
```

`database_name` 및 를 열 통계 작업 실행 일정을 중지하려는 데이터베이스 및 테이블의 `table_name` 실제 이름으로 바꿉니다.

열 통계 생성 일정 재개

통계 생성 일정을 일시 중지한 경우 에서편리한 시간에 일정을 재개할 수 AWS Glue 있습니다. 콘솔 AWS CLI또는 StartColumnStatisticsTaskRunSchedule 작업을 사용하여 AWS Glue 일정을 재개할 수 있습니다.

AWS Management Console

열 통계 생성 일정을 재개하려면

1. AWS Glue 콘솔의 데이터 카탈로그에서 테이블을 선택합니다.

2. 열 통계가 있는 테이블을 선택합니다.
3. 테이블 세부정보 페이지에서 열 통계를 선택합니다.
4. 작업에서 예약된 생성 을 선택하고 재개 를 선택합니다.
5. 재개를 선택하여 확인합니다.

AWS CLI

`database_name` 및 를 열 통계 작업 실행 일정을 중지하려는 데이터베이스 및 테이블의 `table_name` 실제 이름으로 바꿉니다.

```
aws glue start-column-statistics-task-run-schedule \
 --database-name 'database_name' \
 --table-name 'table_name'
```

열 통계 생성 일정 삭제

최적의 쿼리 성능을 위해 일반적으로 통계를 유지하는 up-to-date 것이 좋지만 자동 생성 일정을 제거하는 것이 도움이 될 수 있는 특정 사용 사례가 있습니다.

- 데이터가 비교적 정적 상태로 유지되는 경우 기존 열 통계는 장기간 동안 정확할 수 있으므로 자주 업데이트할 필요가 줄어듭니다. 일정을 삭제하면 변경되지 않은 데이터에 대한 통계 재생성과 관련된 불필요한 리소스 소비 및 오버헤드를 방지할 수 있습니다.
- 통계 생성을 수동으로 제어하는 것이 선호되는 경우. 자동 일정을 삭제하면 관리자는 특정 간격으로 또는 중요한 데이터 변경 후 열 통계를 선택적으로 업데이트하여 프로세스를 유지 관리 전략 및 리소스 할당 요구 사항에 맞게 조정할 수 있습니다.

AWS Management Console

열 통계 생성 일정을 삭제하려면

1. AWS Glue 콘솔의 데이터 카탈로그에서 테이블을 선택합니다.
2. 열 통계가 있는 테이블을 선택합니다.
3. 테이블 세부정보 페이지에서 열 통계를 선택합니다.
4. 작업에서 예약된 생성 , 삭제 를 선택합니다.
5. 삭제를 선택하여 확인합니다.

AWS CLI

database_name 및 를 열 통계 작업 실행 일정을 중지하려는 데이터베이스 및 테이블의 table_name 실제 이름으로 바꿉니다.

DeleteColumnStatisticsTaskSettings API 작업 또는 를 사용하여 열 통계 일정을 삭제할 수 있습니다 AWS CLI. 다음 예제에서는 AWS Command Line Interface ()를 사용하여 열 통계를 생성하기 위한 일정을 삭제하는 방법을 보여줍니다AWS CLI.

```
aws glue delete-column-statistics-task-settings \
    --database-name 'database_name' \
    --table-name 'table_name'
```

온디맨드 열 통계 생성

일정 설정 없이 온디맨드 방식으로 AWS Glue Data Catalog 테이블 작업에 대한 열 통계 작업을 실행할 수 있습니다. 이 옵션은 임시 분석에 유용하거나 통계를 즉시 계산해야 하는 경우에 유용합니다.

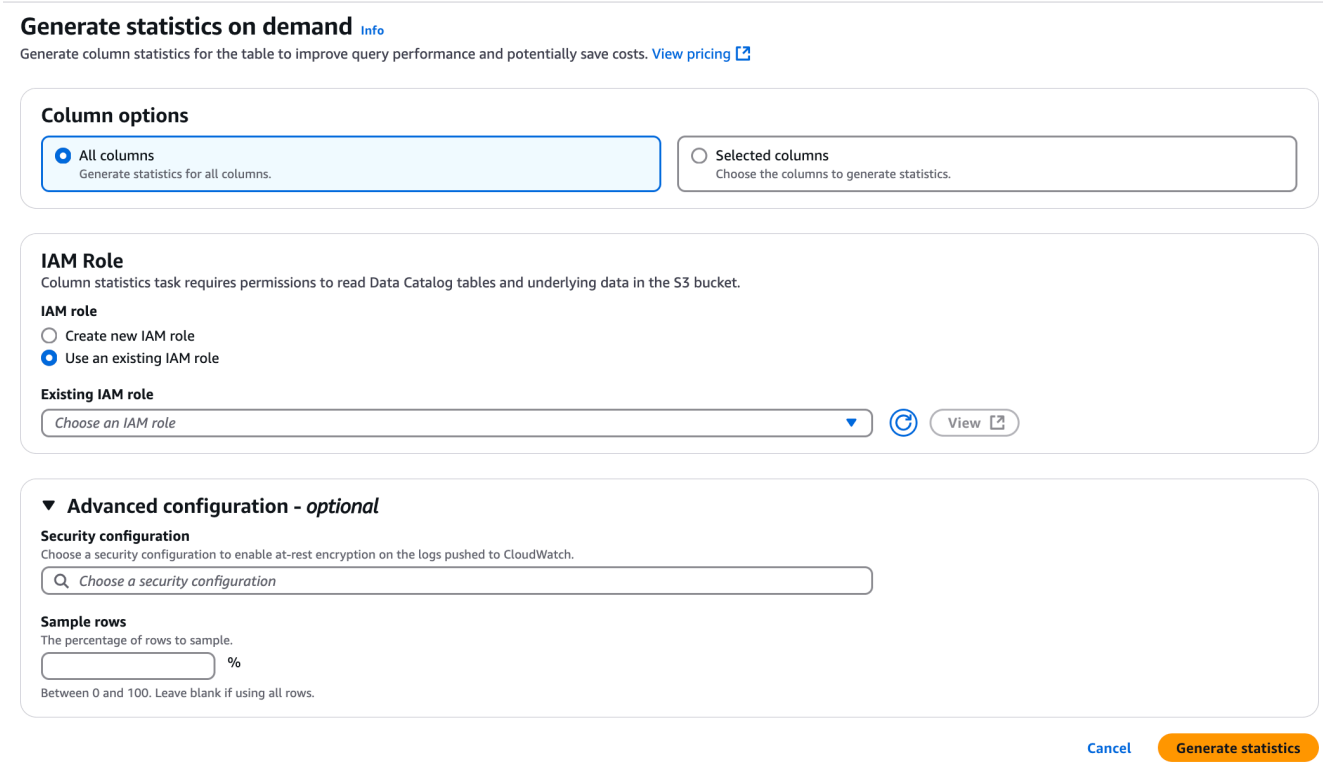
AWS Glue 콘솔 또는 를 사용하여 데이터 카탈로그 테이블에 대한 열 통계를 생성하려면 다음 단계를 따르세요 AWS CLI.

AWS Management Console

콘솔을 사용하여 열 통계를 생성하는 방법

1. 에서 AWS Glue 콘솔에 로그인합니다https://console.aws.amazon.com/glue/.
2. 데이터 카탈로그 테이블을 선택합니다.
3. 목록에서 테이블을 선택합니다.
4. 작업 메뉴에서 통계 생성을 선택합니다.

   테이블 페이지 하단의 열 통계 탭에서 생성, 온디맨드 생성 옵션을 선택할 수도 있습니다.
5. 의 7~11단계를 따라 테이블에 대한 열 통계를 일정에 따른 열 통계 생성 생성합니다.
6. 통계 생성 페이지에서 다음 옵션을 지정합니다.



- 모든 열 - 표의 모든 열에 대한 통계를 생성하려면 이 옵션을 선택합니다.
- 선택한 열 - 특정 열에 대한 통계를 생성하려면 이 옵션을 선택합니다. 드롭다운 목록에서 를 선택합니다.
- IAM 역할 - 열 통계 생성 작업을 실행하는 데 필요한 권한 정책이 있는 새 IAM 역할 생성을 선택합니다. 권한 세부 정보 보기를 선택하여 정책 설명을 검토합니다. 목록에서 IAM 역할을 선택할 수도 있습니다. 필요한 권한에 대한 자세한 정보는 열 통계 생성을 위한 사전 요구 사항 단원을 참조하세요.

  AWS Glue 는 통계를 생성하기 위해 지정하는 역할의 권한을 맡습니다.

  에 대한 역할 제공에 대한 자세한 내용은 에 대한 자격 증명 기반 정책을 AWS Glue참조하세요. AWS Glue
- (선택 사항) 다음으로 로그에 대해 저장 중 암호화를 활성화하는 보안 구성을 선택합니다.
- 샘플 행 - 테이블에서 특정 비율의 행만 선택하여 통계를 생성합니다. 기본값은 모든 행입니다. 위쪽 및 아래쪽 화살표를 사용하여 백분율 값을 늘리거나 줄입니다.

> **Note**
>
> 정확한 통계를 계산하려면 표에 모든 행을 포함하는 것이 좋습니다. 대략적인 값이 허용되는 경우에만 샘플 행을 사용하여 열 통계를 생성하십시오.

통계 생성을 선택하여 작업을 실행합니다.

AWS CLI

이 명령은 지정된 테이블에 대한 열 통계 작업 실행을 트리거합니다. 데이터베이스 이름, 테이블 이름, 통계를 생성할 수 있는 권한이 있는 IAM 역할을 제공하고 선택적으로 통계 계산을 위한 열 이름과 샘플 크기 백분율을 제공해야 합니다.

```
aws glue start-column-statistics-task-run \
    --database-name 'database_name \
    --table-name 'table_name' \
    --role 'arn:aws:iam::123456789012:role/stats-role' \
    --column-name 'col1','col2'  \
    --sample-size 10.0
```

이 명령은 지정된 테이블에 대한 열 통계를 생성하는 작업을 시작합니다.

온디맨드 열 통계 업데이트

쿼리 옵티마이저가 효율적인 실행 계획을 생성하여 쿼리 성능을 개선하고 리소스 소비를 줄이며 전체 시스템 성능을 향상하려면 열 통계를 up-to-date 유지하는 것이 중요합니다. 이 프로세스는 대량 로드 또는 광범위한 수정과 같은 중요한 데이터 변경 후 특히 중요하며, 이로 인해 기존 통계가 더 이상 사용되지 않을 수 있습니다.

열 통계를 새로 고치려면 콘솔에서 통계 생성 작업을 명시적으로 실행해야 합니다. AWS Glue 데이터 카탈로그는 통계를 자동으로 새로 고치지 않습니다.

콘솔에서 AWS Glue의 통계 생성 기능을 사용하지 않는 경우 UpdateColumnStatisticsForTable API 작업 또는 를 사용하여 열 통계를 수동으로 업데이트할 수 있습니다 AWS CLI. 다음 예에서는 AWS CLI 를 이용하여 열 통계를 업데이트하는 방법을 보여 줍니다.

```
aws glue update-column-statistics-for-table --cli-input-json:
```

```
{
    "CatalogId": "111122223333",
    "DatabaseName": "database_name",
    "TableName": "table_name",
    "ColumnStatisticsList": [
        {
            "ColumnName": "col1",
            "ColumnType": "Boolean",
            "AnalyzedTime": "1970-01-01T00:00:00",
            "StatisticsData": {
                "Type": "BOOLEAN",
                "BooleanColumnStatisticsData": {
                    "NumberOfTrues": 5,
                    "NumberOfFalses": 5,
                    "NumberOfNulls": 0
                }
            }
        }
    ]
}
```

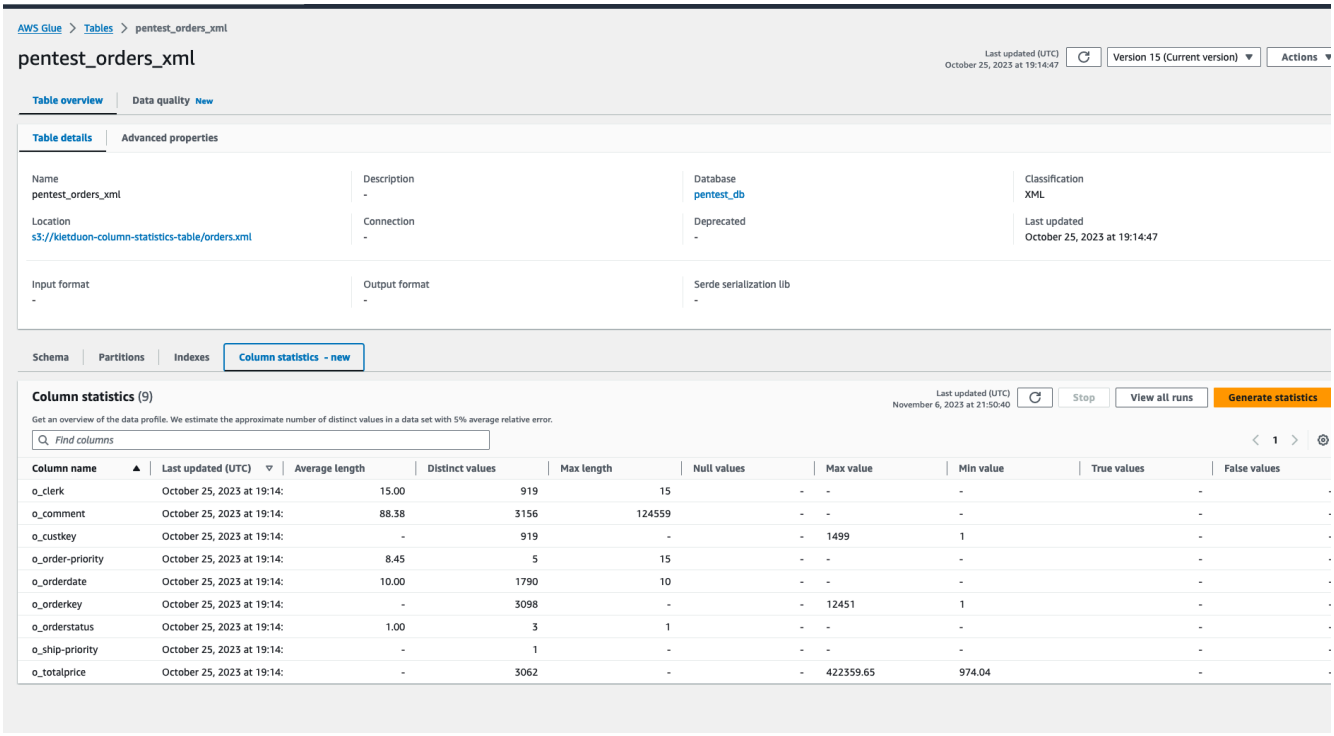
열 통계 보기

통계를 성공적으로 생성한 후 Data Catalog는 쿼리를 실행할 때 최적의 선택을 하기 위해 Amazon Athena 및 Amazon Redshift에 비용 기반 최적화 프로그램에 대한 이 정보를 저장합니다. 통계는 열의 유형에 따라 달라집니다.

AWS Management Console

테이블의 열 통계를 보는 방법

- 열 통계 작업을 실행한 후 테이블 세부정보 페이지의 열 통계 탭에 테이블의 통계가 표시됩니다.



다음과 같은 통계를 사용할 수 있습니다.

- 열 이름: 통계 생성에 사용되는 열 이름
- 최종 업데이트: 통계가 생성된 날짜 및 시각
- 평균 길이: 열에 있는 값의 평균 길이
- 고유 값: 열의 고유 값 총 수입니다. 상대 오차가 5% 인 열의 고유 값 수를 추정합니다.
- 최대값: 열에서 최고 값입니다.
- 최소값: 열에서 최저 값입니다.
- 최대 길이: 열에서 최고 값의 길이입니다.
- 열의 null 값의 수입니다.
- 열의 true 값의 수입니다.
- 열의 false 값의 수입니다.
- numFiles: 테이블의 총 파일 수입니다. 이 값은 고급 속성 탭에 제공됩니다.

AWS CLI

다음 예제에서는 AWS CLI를 사용하여 열 통계를 검색하는 방법을 보여줍니다.

```
aws glue get-column-statistics-for-table \
```

```
    --database-name database_name \
    --table-name table_name \
    --column-names <column_name>
```

GetColumnStatisticsForTable API 작업을 사용하여 열 통계를 볼 수도 있습니다.

컬럼 통계 태스크 실행 보기

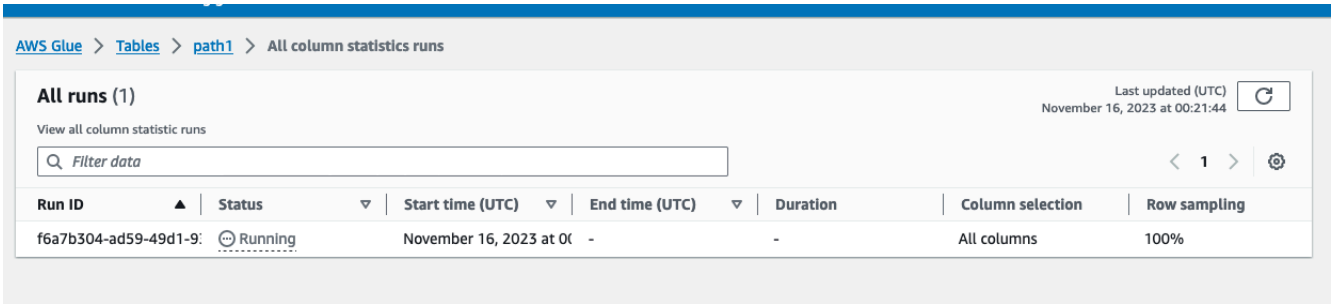
열 통계 작업을 실행한 후 AWS Glue 콘솔 AWS CLI 또는 GetColumnStatisticsTaskRuns 작업을 사용하여 테이블에 대한 작업 실행 세부 정보를 탐색할 수 있습니다.

Console

열 통계 작업 실행 세부 정보를 보는 방법

1. AWS Glue 콘솔의 데이터 카탈로그에서 테이블을 선택합니다.
2. 열 통계가 있는 테이블을 선택합니다.
3. 테이블 세부정보 페이지에서 열 통계를 선택합니다.
4. 실행 보기를 선택합니다.

   지정된 테이블과 관련된 모든 실행에 대한 정보를 볼 수 있습니다.



AWS CLI

다음 예제에서는 DatabaseName 및 TableName의 값을 실제 데이터베이스 및 테이블 이름으로 바꿉니다.

```
aws glue get-column-statistics-task-runs --input-cli-json file://input.json
{
    "DatabaseName": "database_name",
    "TableName": "table_name"
}
```

열 통계 작업 실행 중지

AWS Glue 콘솔 AWS CLI 또는 StopColumnStatisticsTaskRun 작업을 사용하여 테이블에 대한 열 통계 작업 실행을 중지할 수 있습니다.

Console

열 통계 작업을 중지하려면 작업을 실행하십시오

1. AWS Glue 콘솔의 데이터 카탈로그에서 테이블을 선택합니다.
2. 열 통계 작업 실행이 진행 중인 테이블을 선택합니다.
3. 테이블 세부정보 페이지에서 열 통계를 선택합니다.
4. 중지를 선택합니다.

   실행이 완료되기 전에 작업을 중지하면 테이블에 대한 열 통계가 생성되지 않습니다.

AWS CLI

다음 예제에서는 DatabaseName 및 TableName의 값을 실제 데이터베이스 및 테이블 이름으로 바꿉니다.

```
aws glue stop-column-statistics-task-run --input-cli-json file://input.json
{
    "DatabaseName": "database_name",
    "TableName": "table_name"
}
```

컬럼 통계 삭제

DeleteColumnStatisticsForTable API 작업 또는 를 사용하여 열 통계를 삭제할 수 있습니다 AWS CLI. 다음 예제에서는 AWS Command Line Interface ()를 사용하여 열 통계를 삭제하는 방법을 보여줍니다 AWS CLI.

```
aws glue delete-column-statistics-for-table \
    --database-name 'database_name' \
```

```
    --table-name 'table_name' \
    --column-name 'column_name'
```

고려 사항 및 제한

열 통계 생성에는 다음 고려 사항 및 제한 사항이 적용됩니다.

고려 사항

- 샘플링을 사용하여 통계를 생성하면 실행 시간이 줄어들지만 통계가 정확하지 않을 수도 있습니다.
- 데이터 카탈로그에는 통계의 서로 다른 버전이 저장되지 않습니다.
- 테이블당 한 번에 하나의 통계 생성 작업만 실행할 수 있습니다.
- Data Catalog에 등록된 고객 AWS KMS 키를 사용하여 테이블을 암호화하는 경우 는 동일한 키를 AWS Glue 사용하여 통계를 암호화합니다.

열 통계 작업은 통계 생성을 지원합니다.

- IAM 역할에 전체 테이블 권한(IAM 또는 Lake Formation)이 있는 경우.
- Lake Formation 하이브리드 액세스 모드를 사용하여 IAM 역할에 테이블에 대한 권한이 있는 경우.

열 통계 작업은 다음에 대한 통계 생성을 지원하지 않습니다.

- Lake Formation 셀 기반 액세스 제어가 있는 테이블
- 트랜잭션 데이터 레이크 - Linux 파운데이션 Delta Lake, Apache Hudi
- 페더레이션된 데이터베이스의 테이블 - Hive 메타스토어, Amazon Redshift 데이터 공유
- 중첩된 열, 배열, 구조체 데이터 유형.
- 다른 계정에서 공유되는 테이블

데이터 카탈로그 암호화

AWS Key Management Service(AWS KMS)에서 관리하는 암호화 키를 사용하여 AWS Glue Data Catalog에 저장된 메타데이터를 유휴 시에 보호할 수 있습니다. 데이터 카탈로그 설정을 사용하여 새 데이터 카탈로그에 대한 데이터 카탈로그 암호화를 활성화할 수 있습니다. 필요에 따라 기존 데이터 카탈로그의 암호화를 활성화하거나 비활성화할 수 있습니다. 활성화하면 AWS Glue는 카탈로그에 기록되는 모든 새 메타데이터를 암호화하지만 기존 메타데이터는 암호화되지 않은 상태로 유지됩니다.

데이터 카탈로그 암호화에 대한 자세한 내용은 데이터 카탈로그 암호화 섹션을 참조하세요.

Lake Formation을 사용한 데이터 카탈로그 보호

AWS Lake Formation은 AWS에서 안전한 데이터 레이크를 더 쉽게 설정할 수 있게 해주는 서비스입니다. 세분화된 액세스 제어 권한을 정의하여 데이터 레이크를 생성하고 안전하게 관리할 수 있는 중앙 위치를 제공합니다. Lake Formation은 데이터 카탈로그를 사용하여 테이블 정의, 스키마 정보, 데이터 액세스 제어 설정 등과 같은 데이터 레이크에 대한 메타데이터를 저장하고 검색합니다.

Lake Formation에 메타데이터 테이블 또는 데이터베이스의 Amazon S3 데이터 위치를 등록하고 이를 사용하여 데이터 카탈로그 리소스에 대한 메타데이터 수준 권한을 정의할 수 있습니다. 또한 통합 분석 엔진 대신 Lake Formation을 사용하여 Amazon S3에 저장된 기본 데이터에 대한 스토리지 액세스 권한을 관리할 수 있습니다.

자세한 내용은 AWS Lake Formation이란 무엇인가요?를 참조하세요.

AWS Glue에서 AWS Glue Data Catalog 뷰 작업

AWS Glue Data Catalog에서 일반적으로 AWS Glue Data Catalog 뷰라고 하는 뷰를 생성 및 관리할 수 있습니다. 이러한 뷰는 여러 SQL 쿼리 엔진을 지원하므로 Amazon Athena, Amazon Redshift 및 AWS Glue와 같은 다양한 AWS 서비스에서 동일한 뷰에 액세스할 수 있어 유용합니다. Apache Iceberg, Apache Hudi 및 Delta Lake를 기반으로 뷰를 사용할 수 있습니다.

Data Catalog에서 뷰를 생성하면 AWS Lake Formation에서 리소스 권한 부여 및 태그 기반 액세스 제어를 사용하여 이에 대한 액세스 권한을 부여할 수 있습니다. 이 액세스 제어 방법을 사용하면 뷰를 생성할 때 참조한 테이블에 대한 추가 액세스를 구성하지 않아도 됩니다. 이러한 권한 부여 방법을 정의자 시맨틱이라고 하며 이러한 보기를 정의자 보기라고 합니다. AWS Lake Formation에서의 액세스 제어에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 Data Catalog 리소스에서 권한 부여 및 취소를 참조하세요.

Data Catalog 보기는 다음 사용 사례에 유용합니다.

- 세분화된 액세스 제어 - 사용자에게 필요한 권한에 따라 데이터 액세스를 제한하는 뷰를 생성할 수 있습니다. 예를 들어 데이터 카탈로그의 뷰를 사용하여 인사 관리(HR) 부서 소속이 아닌 직원은 개인 식별 정보(PII)를 보지 못하게 할 수 있습니다.
- 완전한 뷰 정의 - Data Catalog의 뷰에 필터를 적용하여 뷰의 데이터 레코드가 항상 완전한지 확인합니다.
- 보안 강화 - 뷰를 생성하는 데 사용되는 쿼리 정의가 완전해야 하므로 Data Catalog 뷰가 악의적인 공격자의 SQL 명령에 덜 취약합니다.

- 간단한 데이터 공유 - AWS Lake Formation에서 교차 계정 데이터 공유를 사용하여 데이터를 이동하지 않고 다른 AWS 계정과 데이터를 공유합니다.

Data Catalog 뷰 생성

Spark SQL을 사용하는 AWS CLI 및 AWS Glue ETL 스크립트를 사용하여 Data Catalog 뷰를 생성할 수 있습니다. Data Catalog 뷰를 생성하는 구문에는 뷰 유형을 `MULTI DIALECT`로 지정하고 `SECURITY` 조건자를 `DEFINER`로 지정하여 정의자 뷰를 나타내는 작업이 포함됩니다.

Data Catalog 뷰를 생성하는 SQL 문 예시:

```
CREATE PROTECTED MULTI DIALECT VIEW database_name.catalog_view SECURITY DEFINER
AS SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY order_date;
```

Data Catalog 뷰를 생성한 후 뷰에 대한 AWS Lake Formation `SELECT` 권한이 있는 IAM 역할을 사용하여 Amazon Athena, Amazon Redshift 또는 AWS Glue ETL 작업과 같은 서비스에서 쿼리할 수 있습니다. 뷰에서 참조된 기본 테이블에 대한 액세스 권한을 부여하지 않아도 됩니다.

Data Catalog 뷰 생성 및 구성에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 AWS Glue Data Catalog 뷰 빌드를 참조하세요.

지원되는 뷰 작업

다음 명령 조각은 Data Catalog 뷰를 사용하는 다양한 방법을 보여줍니다.

CREATE VIEW

Data Catalog 뷰를 생성합니다. 다음은 기존 테이블에서 뷰를 생성하는 방법을 보여주는 샘플입니다.

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER AS SELECT * FROM my_catalog.my_database.source_table
```

ALTER VIEW

사용 가능한 구문:

```
ALTER VIEW view_name [FORCE] ADD DIALECT AS query
```

```
ALTER VIEW view_name [FORCE] UPDATE DIALECT AS query
ALTER VIEW view_name DROP DIALECT
```

FORCE ADD DIALECT 옵션을 사용하여 새 엔진 언어에 따라 스키마 및 하위 객체를 강제 업데이트할 수 있습니다. FORCE를 사용하여 다른 엔진 언어를 업데이트하지 않아도 이로 인해 쿼리 오류가 발생할 수 있습니다. 다음은 샘플을 보여줍니다.

```
ALTER VIEW catalog_view FORCE ADD DIALECTAS
SELECT order_date, sum(totalprice) AS priceFROM source_tableGROUP BY orderdate;
```

다음은 언어를 업데이트하기 위해 뷰를 변경하는 방법을 보여줍니다.

```
ALTER VIEW catalog_view UPDATE DIALECT AS
SELECT count(*) FROM my_catalog.my_database.source_table;
```

DESCRIBE VIEW

뷰를 설명하는 데 사용할 수 있는 구문:

SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name] - 사용자에게 뷰를 설명하는 데 필요한 AWS Glue 및 AWS Lake Formation 권한이 있는 경우 권한으로 열이 나열될 수 있습니다. 다음은 열을 표시하기 위한 몇 가지 샘플 명령입니다.

```
SHOW COLUMNS FROM my_database.source_table;
SHOW COLUMNS IN my_database.source_table;
```

DESCRIBE view_name - 사용자에게 뷰를 설명하는 데 필요한 AWS Glue 및 AWS Lake Formation 권한이 있는 경우 권한으로 메타데이터와 함께 뷰의 열이 나열될 수 있습니다.

DROP VIEW

사용 가능한 구문:

```
DROP VIEW [ IF EXISTS ] view_name
```

다음 샘플은 삭제 전에 뷰가 존재하는지 테스트하는 DROP 문을 보여줍니다.

```
DROP VIEW IF EXISTS catalog_view;
```

`SHOW CREATE VIEW view_name` - 지정된 뷰를 생성하는 SQL 문을 보여줍니다. 다음은 Data Catalog 뷰 생성을 보여주는 샘플입니다.

```
SHOW CREATE TABLE my_database.catalog_view;CREATE PROTECTED MULTI DIALECT VIEW
 my_catalog.my_database.catalog_view (
  net_profit,
  customer_id,
  item_id,
  sold_date)
TBLPROPERTIES (
  'transient_lastDdlTime' = '1736267222')
SECURITY DEFINER AS SELECT * FROM
my_database.store_sales_partitioned_lf WHERE customer_id IN (SELECT customer_id from
 source_table limit 10)
```

SHOW VIEWS

일반 뷰, 다중 언어 뷰(MDV) 및 Spark 언어가 없는 MDV와 같은 카탈로그의 모든 뷰를 나열합니다. 사용 가능한 구문은 다음과 같습니다.

```
SHOW VIEWS [{ FROM | IN } database_name] [LIKE regex_pattern]:
```

다음은 뷰를 표시하는 샘플 명령입니다.

```
SHOW VIEWS IN marketing_analytics LIKE 'catalog_view*';
```

Data Catalog 뷰 생성 및 구성에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 AWS Glue Data Catalog 뷰 빌드를 참조하세요.

Data Catalog 뷰 쿼리

Data Catalog 보기를 생성한 후 뷰를 쿼리할 수 있습니다. AWS Glue 작업에 구성된 IAM 역할에는 Data Catalog 뷰에 대한 Lake Formation SELECT 권한이 있어야 합니다. 뷰에서 참조된 기본 테이블에 대한 액세스 권한을 부여하지 않아도 됩니다.

모든 설정을 마치면 보기를 쿼리할 수 있습니다. 예를 들어 다음 쿼리를 실행하여 뷰에 액세스할 수 있습니다.

```
SELECT * from my_database.catalog_view LIMIT 10;
```

제한 사항

Data Catalog 보기를 사용하는 경우 다음 제한 사항을 고려합니다.

- AWS Glue 5.0 이상에서만 Data Catalog 뷰를 생성할 수 있습니다.
- Data Catalog 뷰 정의자는 뷰에서 액세스하는 기본 테이블에 대해 SELECT 액세스 권한이 있어야 합니다. 특정 기본 테이블의 정의자 역할에 Lake Formation 필터가 적용된 경우 Data Catalog 뷰 생성에 실패합니다.
- 기본 테이블에는 AWS Lake Formation의 IAMAllowedPrincipals 데이터 레이크 권한이 없어야 합니다. 이 권한이 있는 경우 다중 언어 뷰는 IAMAllowedPrincipals 권한이 없는 테이블만 참조라는 오류가 발생할 수 있습니다.
- 테이블의 Amazon S3 위치를 AWS Lake Formation 데이터 레이크 위치로 등록해야 합니다. 테이블이 등록되지 않은 경우 `Multi Dialect views may only reference AWS Lake Formation managed tables` 오류가 발생합니다. AWS Lake Formation에 Amazon S3 위치를 등록하는 자세한 방법은 AWS Lake Formation 개발자 안내서에서 Amazon S3 위치 등록을 참조하세요.
- PROTECTED Data Catalog 보기만 생성할 수 있습니다. UNPROTECTED 보기는 지원되지 않습니다.
- Data Catalog 뷰 정의의 다른 AWS 계정에 있는 테이블을 참조할 수 없습니다. 또한 별도의 리전에 있는 동일한 계정의 테이블을 참조할 수 없습니다.
- 계정 또는 리전 간에 데이터를 공유하려면 AWS Lake Formation 리소스 링크를 사용하여 전체 뷰를 계정 및 리전 간에 공유해야 합니다.
- 사용자 정의 함수(UDF)는 지원되지 않습니다.
- Data Catalog 보기는 다른 보기를 참조할 수 없습니다.

데이터 카탈로그 액세스

AWS Glue Data Catalog(Data Catalog)를 사용하면 데이터를 검색하고 파악할 수 있습니다. 데이터 카탈로그는 스키마 정의, 데이터 유형, 위치 및 기타 메타데이터를 일관되게 유지 관리할 수 있는 방법을 제공합니다. 다음 방법을 사용하여 데이터 카탈로그에 액세스할 수 있습니다.

- AWS Glue 콘솔 - 웹 기반 사용자 인터페이스인 AWS Glue 콘솔을 통해 데이터 카탈로그에 액세스하고 관리할 수 있습니다. 콘솔을 사용하면 데이터베이스, 테이블 및 관련 메타데이터를 찾고 검색할 수 있을 뿐만 아니라 메타데이터 정의를 생성, 업데이트 및 삭제할 수 있습니다.
- AWS Glue 크롤러 - 크롤러는 데이터 소스를 자동으로 스캔하고 데이터 카탈로그를 메타데이터로 채우는 프로그램입니다. 크롤러를 생성하고 실행하여 Amazon S3, Amazon RDS, Amazon

DynamoDB, Amazon CloudWatch와 MySQL, PostgreSQL 등 같은 JDBC 호환 관계형 데이터베이스뿐만 아니라 Snowflake, Google BigQuery 등과 같은 여러 비 AWS 소스에서 데이터를 검색하고 카탈로그화할 수 있습니다.

- AWS Glue API - AWS Glue API를 사용하여 프로그래밍 방식으로 데이터 카탈로그에 액세스할 수 있습니다. 이러한 API를 사용하면 다른 애플리케이션과 서비스에서 프로그래밍 방식으로 데이터 카탈로그와 상호 작용하여 자동화 및 통합을 수행할 수 있습니다.
- AWS Command Line Interface(AWS CLI) - 명령줄에서 AWS CLI를 사용하여 데이터 카탈로그에 액세스하고 관리할 수 있습니다. CLI는 메타데이터 정의를 생성, 업데이트 및 삭제하는 명령과 메타데이터 정보를 쿼리하고 검색하는 명령을 제공합니다.
- 다른 AWS 서비스와의 통합 - 데이터 카탈로그는 다른 많은 AWS 서비스와 통합되므로 이를 통해 카탈로그에 저장된 메타데이터에 액세스하고 활용할 수 있습니다. 예를 들어 Amazon Athena를 사용하면 데이터 카탈로그의 메타데이터를 사용하여 데이터 소스를 쿼리할 수 있으며, AWS Lake Formation을 사용하면 데이터 카탈로그 리소스에 대한 데이터 액세스 및 거버넌스를 관리할 수 있습니다.

주제

- AWS Glue Iceberg REST 엔드포인트를 사용하여 Data Catalog에 연결
- AWS Glue Iceberg REST 확장 엔드포인트를 사용하여 Data Catalog에 연결
- Apache Iceberg용 AWS Glue REST API 사양
- 독립 실행형 Spark 애플리케이션에서 Data Catalog에 연결
- Amazon Redshift와 Apache Iceberg 간의 데이터 매핑
- AWS Glue Iceberg REST 카탈로그 API 사용 시 고려 사항 및 제한 사항

AWS Glue Iceberg REST 엔드포인트를 사용하여 Data Catalog에 연결

AWS Glue의 Iceberg REST 엔드포인트는 Apache Iceberg REST 사양에 명시된 API 작업을 지원합니다. Iceberg REST 클라이언트를 사용하여 분석 엔진에서 실행되는 애플리케이션을 Data Catalog에 호스팅되는 REST 카탈로그에 연결할 수 있습니다.

엔드포인트는 v2로 기본 설정된 v1 및 v2의 Apache Iceberg 테이블 사양을 모두 지원합니다. Iceberg 테이블 v1 사양을 사용하는 경우 API 직접 호출에서 v1을 지정해야 합니다. API 작업을 사용하여 Amazon S3 객체 스토리지와 Amazon S3 Table 스토리지 모두에 저장된 Iceberg 테이블에 액세스할 수 있습니다.

엔드포인트 구성

서비스 엔드포인트를 사용하여 AWS Glue Iceberg REST 카탈로그에 액세스할 수 있습니다. 리전별 엔드포인트는 AWS Glue 서비스 엔드포인트 참조 가이드를 참조하세요. 예를 들어 us-east-1 리전에서 AWS Glue에 연결할 때 다음과 같이 엔드포인트 URI 속성을 구성해야 합니다.

```
Endpoint : https://glue.us-east-1.amazonaws.com/iceberg
```

추가 구성 속성 – Iceberg 클라이언트를 사용하여 Spark와 같은 분석 엔진을 서비스 엔드포인트에 연결하는 경우 다음 애플리케이션 구성 속성을 지정해야 합니다.

```
catalog_name = "mydatacatalog"
aws_account_id = "123456789012"
aws_region = "us-east-1"
spark = SparkSession.builder \
    ... \
    .config("spark.sql.defaultCatalog", catalog_name) \
    .config(f"spark.sql.catalog.{catalog_name}",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config(f"spark.sql.catalog.{catalog_name}.type", "rest") \
    .config(f"spark.sql.catalog.{catalog_name}.uri", "https://glue.
{aws_region}.amazonaws.com/iceberg") \
    .config(f"spark.sql.catalog.{catalog_name}.warehouse", "{aws_account_id}") \
    .config(f"spark.sql.catalog.{catalog_name}.rest.sigv4-enabled", "true") \
    .config(f"spark.sql.catalog.{catalog_name}.rest.signing-name", "glue") \

 .config("spark.sql.extensions","org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensi
 \
    .getOrCreate()
```

AWS Glue Iceberg 엔드포인트 `https://glue.us-east-1.amazonaws.com/iceberg`는 다음과 같은 Iceberg REST API를 지원합니다.

- GetConfig
- ListNamespaces
- CreateNamespace
- LoadNamespaceMetadata
- UpdateNamespaceProperties
- DeleteNamespace
- ListTables

- CreateTable
- LoadTable
- TableExists
- UpdateTable
- DeleteTable

접두사 및 카탈로그 경로 파라미터

Iceberg REST 카탈로그 API는 요청 URL에 자유 형식 접두사를 사용합니다. 예를 들어 `ListNamespaces` API 직접 호출은 `GET/v1/{prefix}/namespaces` URL 형식을 사용합니다. AWS Glue 접두사는 REST 경로가 AWS Glue 다중 카탈로그 계층 구조를 정렬하도록 항상 `/catalogs/{catalog}` 구조를 따릅니다. `{catalog}` 경로 파라미터는 다음 규칙에 따라 파생될 수 있습니다.

액세스 패턴	Glue 카탈로그 ID 스타일	접두사 스타일	기본 카탈로그 ID 예	REST 경로 예
현재 계정의 기본 카탈로그에 액세스	불필요	:	해당 사항 없음	GET /v1/catalogs/:/namespaces
특정 계정의 기본 카탈로그에 액세스	accountID	accountID	111122223333	GET /v1/catalogs/111122223333/namespaces
현재 계정의 중첩 카탈로그에 액세스	catalog1/catalog2	catalog1/catalog2	rmscatalog1:db1	GET /v1/catalogs/rmscatalog1:db1/namespaces
특정 계정의 중첩 카탈로그에 액세스	accountId:catalog1/catalog2	accountId:catalog1/catalog2	123456789012/rmscatalog1:db1	GET /v1/catalogs/123456789012:rmscatalog1:db1/namespaces

이 카탈로그 ID/접두사 매핑은 REST API를 직접적으로 호출하는 경우에만 필요합니다. 엔진을 통해 AWS Glue Iceberg REST 카탈로그 API로 작업하는 경우 Iceberg REST AWS Glue 카탈로그 API 설정의 `warehouse` 파라미터 또는 AWS Glue 익스텐션 API 설정의 `glue.id` 파라미터에 카탈로그 ID를 지정해야 합니다. 예를 들어, Spark와 함께 Iceberg 클러스터 사용에서 EMR Spark와 함께 사용하는 방법을 확인하세요.

네임스페이스 경로 파라미터

Iceberg REST 카탈로그 API 경로의 네임스페이스는 여러 수준을 가질 수 있습니다. 그러나 AWS Glue는 단일 수준 네임스페이스만 지원합니다. 다중 수준 카탈로그 계층 구조의 네임스페이스에 액세스하려면 네임스페이스 위의 다중 수준 카탈로그에 연결하여 네임스페이스를 참조하면 됩니다. 이렇게 하면 다중 수준 네임스페이스를 사용하는 것에 비해 호환성 문제 없이 `catalog.namespace.table`의 3부 표기법을 지원하는 모든 쿼리 엔진이 AWS Glue의 다중 수준 카탈로그 계층 구조의 객체에 액세스할 수 있습니다.

AWS Glue Iceberg REST 확장 엔드포인트를 사용하여 Data Catalog에 연결

AWS Glue Iceberg REST 확장 엔드포인트는 Apache Iceberg REST 사양에 없는 추가 API를 제공하고 서버 측 스캔 계획 기능을 제공합니다. 이러한 추가 API는 Amazon Redshift 관리형 스토리지에 저장된 테이블에 액세스할 때 사용됩니다. 엔드포인트는 애플리케이션에서 Apache Iceberg AWS Glue Data Catalog 확장 프로그램을 사용하여 액세스할 수 있습니다.

엔드포인트 구성 - 서비스 엔드포인트를 사용하여 Redshift 관리형 스토리지에 테이블이 있는 카탈로그에 액세스할 수 있습니다. 리전별 엔드포인트는 AWS Glue 서비스 엔드포인트 참조 가이드를 참조하세요. 예를 들어 us-east-1 리전에서 AWS Glue에 연결할 때 다음과 같이 엔드포인트 URI 속성을 구성해야 합니다.

```
Endpoint : https://glue.us-east-1.amazonaws.com/extensions
```

```
catalog_name = "myredshiftcatalog"
aws_account_id = "123456789012"
aws_region = "us-east-1"
spark = SparkSession.builder \
    .config("spark.sql.defaultCatalog", catalog_name) \
    .config(f"spark.sql.catalog.{catalog_name}",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config(f"spark.sql.catalog.{catalog_name}.type", "glue") \
    .config(f"spark.sql.catalog.{catalog_name}.glue.id",
 "{123456789012}:redshiftnamespacecatalog/redshiftdb") \
```

```
.config("spark.sql.extensions","org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensi
\
   .getOrCreate()
```

Apache Iceberg용 AWS Glue REST API 사양

이 섹션에는 AWS Glue Iceberg REST 카탈로그 및 AWS Glue 익스텐션 API에 대한 사양과 이러한 API를 사용할 때의 고려 사항이 포함되어 있습니다.

AWS Glue Data Catalog 엔드포인트에 대한 API 요청은 AWS Signature Version 4(SigV4)를 사용하여 인증됩니다. AWS SigV4에 대한 자세한 내용은 API 요청용 AWS Signature Version 4 섹션을 참조하세요.

AWS Glue 서비스 엔드포인트 및 AWS Glue 메타데이터에 액세스할 때 애플리케이션은 IAM 작업이 필요한 `glue:getCatalog` IAM 역할을 수임합니다.

Data Catalog 및 해당 객체에 대한 액세스는 IAM, Lake Formation 또는 Lake Formation 하이브리드 모드 권한을 사용하여 관리할 수 있습니다.

Data Catalog의 페더레이션 카탈로그에는 Lake Formation에 등록된 데이터 위치가 있습니다. Lake Formation은 Data Catalog와 연동하며 Data Catalog 객체에 대한 사용자 액세스를 관리할 수 있는 데이터베이스 스타일 권한을 제공합니다.

IAM, AWS Lake Formation 또는 Lake Formation 하이브리드 모드 권한을 사용하여 기본 Data Catalog 및 해당 객체에 대한 액세스를 관리할 수 있습니다.

Lake Formation 관리형 객체에서 데이터를 생성, 삽입 또는 삭제하려면 IAM 사용자 또는 역할에 대한 권한을 설정해야 합니다.

- CREATE_CATALOG – 카탈로그를 생성하는 데 필요합니다.
- CREATE_DATABASE – 데이터베이스를 생성하는 데 필요합니다.
- CREATE_TABLE – 테이블을 생성하는 데 필요합니다.
- DELETE - 테이블에서 데이터를 삭제하는 데 필요합니다.
- DESCRIBE - 메타데이터를 읽는 데 필요합니다.
- DROP - 테이블 또는 데이터베이스를 삭제하는 데 필요합니다.
- INSERT - 위탁자가 테이블에 데이터를 삽입해야 할 때 필요합니다.
- SELECT - 위탁자가 테이블에서 데이터를 선택해야 할 때 필요합니다.

자세한 내용은 AWS Lake Formation 개발자 가이드에서 Lake Formation 권한 참조를 참조하세요.

GetConfig

일반 정보

작업 이름	GetConfig
유형	Iceberg REST 카탈로그 API
REST 경로	GET /iceberg/v1/config
IAM 작업	glue:GetCatalog
Lake Formation 권한	해당 사항 없음
CloudTrail 이벤트	glue:GetCatalog
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L67

고려 사항 및 제한 사항

- `warehouse` 쿼리 파라미터는 AWS Glue 카탈로그 ID로 설정해야 합니다. 설정하지 않을 경우 현재 계정의 루트 카탈로그가 응답을 반환하는 데 사용됩니다. 자세한 내용은 the section called “접두사 및 카탈로그 경로 파라미터” 단원을 참조하십시오.

GetCatalog

일반 정보

작업 이름	GetCatalog
유형	AWS Glue 익스텐션 API
REST 경로	GET/extensions/v1/catalogs/{catalog}
IAM 작업	glue:GetCatalog
Lake Formation 권한	DESCRIBE

CloudTrail 이벤트	glue:GetCatalog
API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L40

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.

ListNamespaces

일반 정보

작업 이름	ListNamespaces
유형	Iceberg REST 카탈로그 API
REST 경로	GET/iceberg/v1/catalogs/{catalog}/namespaces
IAM 작업	glue:GetDatabase
Lake Formation 권한	ALL, DESCRIBE, SELECT
CloudTrail 이벤트	glue:GetDatabase
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L205

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- 그 다음 수준의 네임스페이스만 표시됩니다. 네임스페이스를 더 깊은 수준에서 나열하려면 카탈로그 경로 파라미터에 중첩 카탈로그 ID를 지정합니다.

CreateNamespace

일반 정보

작업 이름	CreateNamespace
유형	Iceberg REST 카탈로그 API
REST 경로	POST/iceberg/v1/catalogs/{catalog}/namespaces
IAM 작업	glue:CreateDatabase
Lake Formation 권한	ALL, DESCRIBE, SELECT
CloudTrail 이벤트	glue:CreateDatabase
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L256

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- 단일 수준 네임스페이스만 생성할 수 있습니다. 다중 수준 네임스페이스를 생성하려면 각 수준을 반복적으로 생성하고 카탈로그 경로 파라미터를 사용하여 해당 수준에 연결해야 합니다.

StartCreateNamespaceTransaction

일반 정보

작업 이름	StartCreateNamespaceTransaction
유형	AWS Glue 익스텐션 API
REST 경로	POST/extensions/v1/catalogs/{catalog}/namespaces
IAM 작업	glue:CreateDatabase
Lake Formation 권한	ALL, DESCRIBE, SELECT

CloudTrail 이벤트	glue:CreateDatabase
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L256

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- 단일 수준 네임스페이스만 생성할 수 있습니다. 다중 수준 네임스페이스를 생성하려면 각 수준을 반복적으로 생성하고 카탈로그 경로 파라미터를 사용하여 해당 수준에 연결해야 합니다.
- 이 API는 비동기식이며 `CheckTransactionStatus` API 직접 호출을 사용하여 추적하는 데 사용할 수 있는 트랜잭션 ID를 반환합니다.
- `GetCatalog` API 직접 호출의 응답에 `use-extensions=true` 파라미터가 포함된 경우에만 이 API를 직접적으로 호출할 수 있습니다.

LoadNamespaceMetadata

일반 정보

작업 이름	LoadNamespaceMetadata
유형	Iceberg REST 카탈로그 API
REST 경로	GET/iceberg/v1/catalogs/{catalog}/namespaces/{ns}
IAM 작업	glue:GetDatabase
Lake Formation 권한	ALL, DESCRIBE, SELECT
CloudTrail 이벤트	glue:GetDatabase
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L302

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.

UpdateNamespaceProperties

일반 정보

작업 이름	UpdateNamespaceProperties
유형	Iceberg REST 카탈로그 API
REST 경로	POST /iceberg/v1/catalogs/{catalog}/namespaces/{ns}/properties
IAM 작업	glue:UpdateDatabase
Lake Formation 권한	ALL, ALTER
CloudTrail 이벤트	glue:UpdateDatabase
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L400

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.

DeleteNamespace

일반 정보

작업 이름	DeleteNamespace

유형	Iceberg REST 카탈로그 API
REST 경로	DELETE/iceberg/v1/catalogs/{catalog}/namespces/{ns}
IAM 작업	glue:DeleteDatabase
Lake Formation 권한	ALL, DROP
CloudTrail 이벤트	glue:DeleteDatabase
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L365

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 데이터베이스에 객체가 존재할 경우, 해당 작업은 실패합니다.
- 이 API는 비동기식이며 `CheckTransactionStatus` API 직접 호출을 사용하여 추적하는 데 사용할 수 있는 트랜잭션 ID를 반환합니다.
- `GetCatalog` API 직접 호출이 응답으로 `use-extensions=true`를 나타내는 경우에만 이 API를 사용할 수 있습니다.

StartDeleteNamespaceTransaction

일반 정보

작업 이름	StartDeleteNamespaceTransaction
유형	AWS Glue 익스텐션 API
REST 경로	DELETE /extensions/v1/catalogs/{catalog}/namespces/{ns}
IAM 작업	glue:DeleteDatabase
Lake Formation 권한	ALL, DROP

CloudTrail 이벤트	glue:DeleteDatabase
API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L85

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 데이터베이스에 객체가 존재할 경우, 해당 작업은 실패합니다.
- 이 API는 비동기식이며 `CheckTransactionStatus` API 직접 호출을 사용하여 추적하는 데 사용할 수 있는 트랜잭션 ID를 반환합니다.
- `GetCatalog` API 직접 호출이 응답으로 `use-extensions=true`를 나타내는 경우에만 이 API를 사용할 수 있습니다.

ListTables

일반 정보

작업 이름	ListTables
유형	Iceberg REST 카탈로그 API
REST 경로	GET /iceberg/v1/catalogs/{catalog}/namespaces/{ns}/tables
IAM 작업	glue:GetTables
Lake Formation 권한	ALL, SELECT, DESCRIBE
CloudTrail 이벤트	glue:GetTables
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L463

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- Iceberg가 아닌 테이블을 포함한 모든 테이블이 나열됩니다. 테이블을 Iceberg 테이블로 로드할 수 있는지 여부를 확인하려면 `LoadTable` 작업을 직접적으로 호출합니다.

CreateTable

일반 정보

작업 이름	CreateTable
유형	Iceberg REST 카탈로그 API
REST 경로	GET /iceberg/v1/catalogs/{catalog}/namespaces/{ns}/tables
IAM 작업	glue:CreateTable
Lake Formation 권한	ALL, CREATE_TABLE
CloudTrail 이벤트	glue:CreateTable
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L497

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 스테이징이 포함된 `CreateTable`은 지원되지 않습니다. `stageCreate` 쿼리 파라미터를 지정하면 작업이 실패합니다. 즉, `CREATE TABLE AS SELECT`와 같은 작업은 지원되지 않으며, 해결 방법으로 `CREATE TABLE` 및 `INSERT INTO`의 조합을 사용할 수 있습니다.
- `CreateTable` API 작업은 `state-create = TRUE` 옵션을 지원하지 않습니다.

StartCreateTableTransaction

일반 정보

작업 이름	CreateTable
유형	AWS Glue 익스텐션 API
REST 경로	POST/extensions/v1/catalogs/{catalog}/namespaces/{ns}/tables
IAM 작업	glue:CreateTable
Lake Formation 권한	ALL, CREATE_TABLE
CloudTrail 이벤트	glue:CreateTable
API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L107

고려 사항 및 제한 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 스테이징이 포함된 `CreateTable`은 지원되지 않습니다. `stageCreate` 쿼리 파라미터를 지정하면 작업이 실패합니다. 즉, `CREATE TABLE AS SELECT`와 같은 작업은 지원되지 않으며, 해결 방법으로 `CREATE TABLE` 및 `INSERT INTO`의 조합을 사용해야 합니다.
- 이 API는 비동기식이며 `CheckTransactionStatus` API 직접 호출을 사용하여 추적하는 데 사용할 수 있는 트랜잭션 ID를 반환합니다.
- `GetCatalog` API 직접 호출이 응답으로 `use-extensions=true`를 나타내는 경우에만 이 API를 사용할 수 있습니다.

LoadTable

일반 정보

작업 이름	LoadTable
유형	Iceberg REST 카탈로그 API
REST 경로	GET /iceberg/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}
IAM 작업	glue:GeTTable
Lake Formation 권한	ALL, SELECT, DESCRIBE
CloudTrail 이벤트	glue:GetTable
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L616

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 스테이징이 포함된 `CreateTable`은 지원되지 않습니다. `stageCreate` 쿼리 파라미터를 지정하면 작업이 실패합니다. 즉, `CREATE TABLE AS SELECT`와 같은 작업은 지원되지 않으며, 해결 방법으로 `CREATE TABLE` 및 `INSERT INTO`의 조합을 사용해야 합니다.
- 이 API는 비동기식이며 `CheckTransactionStatus` API 직접 호출을 사용하여 추적하는 데 사용할 수 있는 트랜잭션 ID를 반환합니다.
- `GetCatalog` API 직접 호출이 응답으로 `use-extensions=true`를 나타내는 경우에만 이 API를 사용할 수 있습니다.

ExtendedLoadTable

일반 정보

작업 이름	LoadTable
유형	AWS Glue 익스텐션 API
REST 경로	GET /extensions/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}
IAM 작업	glue:GetTable
Lake Formation 권한	ALL, SELECT, DESCRIBE
CloudTrail 이벤트	glue:GetTable
API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L134

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 스냅샷 쿼리 파라미터에는 `all` 모드만 지원됩니다.
- `LoadTable` API와 비교하여 `ExtendedLoadTable` API는 다음과 같은 측면에서 다릅니다.
  - 모든 필드를 사용할 수 있도록 엄격하게 강제하지는 않습니다.
  - 응답의 구성 필드에 다음과 같은 추가 파라미터를 제공합니다.

    추가 파라미터

Config 키	설명
aws.server-side-capabilities.scan-planning	PreparePlan 테이블 및 PlanTable API를 사용하여 테이블을 스캔할 수 있는지 여부를 나타냅니다.

Config 키	설명
aws.server-side-capabilities.data-commit	StartUpdateTable 트랜잭션을 사용하여 테이블을 커밋할 수 있는지 여부를 나타냅니다.
aws.glue.staging.location	엔진이 임시 데이터 파일을 작성하는 데 사용할 수 있는 서비스 관리형 스테이징 위치이며 서버 측 스캔 계획 또는 데이터 커밋에 사용됩니다.
aws.glue.staging.access-key-id	서비스 관리형 스테이징 위치에 액세스하기 위한 임시 AWS 자격 증명의 일부이며 서버 측 스캔 계획 또는 데이터 커밋에 사용됩니다.
aws.glue.staging.secret-access-key	서비스 관리형 스테이징 위치에 액세스하기 위한 임시 AWS 자격 증명의 일부이며 서버 측 스캔 계획 또는 데이터 커밋에 사용됩니다.
aws.glue.staging.session-token	서비스 관리형 스테이징 위치에 액세스하기 위한 임시 AWS 자격 증명의 일부이며 서버 측 스캔 계획 또는 데이터 커밋에 사용됩니다.
aws.glue.staging.expiration-ms	서비스 관리형 스테이징 위치에 액세스하기 위한 임시 자격 증명의 만료 시간이며 서버 측 스캔 계획 또는 데이터 커밋에 사용됩니다.
aws.glue.staging.data-transfer-role-arn	서비스 관리형 스테이징 위치에 액세스하기 위해 수임할 수 있는 IAM 역할이며 서버 측 스캔 계획 또는 데이터 커밋에 사용됩니다.

PreplanTable

일반 정보

작업 이름	PreplanTable
유형	AWS Glue 익스텐션 API
REST 경로	POST /extensions/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}/preplan

IAM 작업	glue:GetTable
Lake Formation 권한	ALL, SELECT, DESCRIBE
CloudTrail 이벤트	glue:GetTable
API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L211

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 이 API의 호출자는 항상 페이지 토큰을 기반으로 가져올 나머지 결과가 있는지 확인해야 합니다. 서버 측이 아직 처리 중이지만 지정된 응답 시간 내에 결과를 생성할 수 없는 경우 빈 페이지 항목이 있지만 페이지 매김 토큰이 있는 응답이 가능합니다.
- `ExtendedLoadTable` API 응답에 `aws.server-side-capabilities.scan-planning=true`가 포함된 경우에만 이 API를 사용할 수 있습니다.

PlanTable

일반 정보

작업 이름	PlanTable
유형	AWS Glue 익스텐션 API
REST 경로	POST /extensions/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}/plan
IAM 작업	glue:GetTable
Lake Formation 권한	ALL, SELECT, DESCRIBE
CloudTrail 이벤트	glue:GetTable

API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L243

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 이 API의 호출자는 항상 페이지 토큰을 기반으로 가져올 나머지 결과가 있는지 확인해야 합니다. 서버 측이 아직 처리 중이지만 지정된 응답 시간 내에 결과를 생성할 수 없는 경우 빈 페이지 항목이 있지만 페이지 매김 토큰이 있는 응답이 가능합니다.
- `ExtendedLoadTable` API 응답에 `aws.server-side-capabilities.scan-planning=true`가 포함된 경우에만 이 API를 사용할 수 있습니다.

TableExists

일반 정보

작업 이름	TableExists
유형	Iceberg REST 카탈로그 API
REST 경로	HEAD/iceberg/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}
IAM 작업	glue:GetTable
Lake Formation 권한	ALL, SELECT, DESCRIBE
CloudTrail 이벤트	glue:GetTable
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L833

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.

UpdateTable

일반 정보

작업 이름	UpdateTable
유형	Iceberg REST 카탈로그 API
REST 경로	POST /iceberg/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}
IAM 작업	glue:UpdateTable
Lake Formation 권한	ALL, ALTER
CloudTrail 이벤트	glue:UpdateTable
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L677

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.

StartUpdateTableTransaction

일반 정보

작업 이름	StartUpdateTableTransaction

유형	AWS Glue 익스텐션 API
REST 경로	POST/extensions/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}
IAM 작업	glue:UpdateTable
Lake Formation 권한	ALL, ALTER
CloudTrail 이벤트	glue:UpdateTable
API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L154

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- 이 API는 비동기식이며 `CheckTransactionStatus` API 직접 호출을 사용하여 추적하는 데 사용할 수 있는 트랜잭션 ID를 반환합니다.
- 이 API를 통해 `RenamTable` 작업을 수행할 수도 있습니다. 이 경우 호출자는 이름을 바꿀 테이블에 대한 glue:CreateTable 또는 LakeFormation CREATE_TABLE 권한도 얻어야 합니다.
- `ExtendedLoadTable` API 응답에 `aws.server-side-capabilities.scan-planning=true`가 포함된 경우에만 이 API를 사용할 수 있습니다.

DeleteTable

일반 정보

작업 이름	DeleteTable
유형	Iceberg REST 카탈로그 API
REST 경로	DELETE/iceberg/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}

IAM 작업	glue:DeleteTable
Lake Formation 권한	ALL, DROP
CloudTrail 이벤트	glue:DeleteTable
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L793

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- `DeleteTable` API 작업은 제거 옵션을 지원합니다. 제거를 `true`로 설정하면 테이블 데이터가 삭제되고, 그렇지 않으면 데이터가 삭제되지 않습니다. Amazon S3의 테이블에서는 이 작업을 실행해도 테이블 데이터가 삭제되지 않습니다. 테이블이 Amazon S3 및 `purge = TRUE,`에 저장되어 있는 경우 이 작업이 실패합니다.

  Amazon Redshift 관리형 스토리지에 저장된 테이블의 경우 이 작업은 Amazon Redshift의 `DROP TABLE` 동작과 마찬가지로 테이블 데이터를 삭제합니다. 테이블이 Amazon Redshift 및 `purge = FALSE`에 저장되어 있는 경우 이 작업이 실패합니다.
- `purgeRequest=true`는 지원되지 않습니다.

StartDeleteTableTransaction

일반 정보

작업 이름	StartDeleteTableTransaction
유형	AWS Glue 익스텐션 API
REST 경로	DELETE /extensions/v1/catalogs/{catalog}/namespaces/{ns}/tables/{table}
IAM 작업	glue:DeleteTable

Lake Formation 권한	ALL, DROP
CloudTrail 이벤트	glue:DeleteTable
API 정의 열기	https://github.com/apache/iceberg/blob/apache-iceberg-1.6.1/open-api/rest-catalog-open-api.yaml#L793

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.
- REST 경로 파라미터에는 단일 수준 네임스페이스만 지정할 수 있습니다. 자세한 내용은 the section called “네임스페이스 경로 파라미터” 섹션을 참조하세요.
- `purgeRequest=false`는 지원되지 않습니다.
- 이 API는 비동기식이며 `CheckTransactionStatus`를 통해 추적할 수 있는 트랜잭션 ID를 반환합니다.

CheckTransactionStatus

일반 정보

작업 이름	CheckTransactionStatus
유형	AWS Glue 익스텐션 API
REST 경로	POST/extensions/v1/transactions/status
IAM 작업	트랜잭션을 시작하는 작업과 동일한 권한
Lake Formation 권한	트랜잭션을 시작하는 작업과 동일한 권한
API 정의 열기	https://github.com/awslabs/glue-extensions-for-iceberg/blob/main/glue-extensions-api.yaml#L273

고려 사항

- 카탈로그 경로 파라미터는 the section called “접두사 및 카탈로그 경로 파라미터” 섹션에 설명된 스타일을 따라야 합니다.

독립 실행형 Spark 애플리케이션에서 Data Catalog에 연결

독립 실행형 애플리케이션에서 Apache Iceberg 커넥터를 사용하여 Data Catalog에 연결할 수 있습니다.

1. Spark 애플리케이션에 대한 IAM 역할을 생성합니다.
2. AWS Glue Iceberg 커넥터를 사용하여 Iceberg Rest 엔드포인트에 연결합니다.

```
# configure your application. Refer to https://docs.aws.amazon.com/cli/latest/
userguide/cli-configure-envvars.html for best practices on configuring environment
 variables.
export AWS_ACCESS_KEY_ID=$(aws configure get appUser.aws_access_key_id)
export AWS_SECRET_ACCESS_KEY=$(aws configure get appUser.aws_secret_access_key)
export AWS_SESSION_TOKEN=$(aws configure get appUser.aws_secret_token)

export AWS_REGION=us-east-1
export REGION=us-east-1
export AWS_ACCOUNT_ID = {specify your aws account id here}

~/spark-3.5.3-bin-hadoop3/bin/spark-shell \
    --packages org.apache.iceberg:iceberg-spark-runtime-3.4_2.12:1.6.0 \
    --conf
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
 \
    --conf "spark.sql.defaultCatalog=spark_catalog" \
    --conf "spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkCatalog" \
    --conf "spark.sql.catalog.spark_catalog.type=rest" \
    --conf "spark.sql.catalog.spark_catalog.uri=https://glue.us-east-1.amazonaws.com/
iceberg" \
    --conf "spark.sql.catalog.spark_catalog.warehouse = {AWS_ACCOUNT_ID}" \
    --conf "spark.sql.catalog.spark_catalog.rest.sigv4-enabled=true" \
    --conf "spark.sql.catalog.spark_catalog.rest.signing-name=glue" \
    --conf "spark.sql.catalog.spark_catalog.rest.signing-region=us-east-1" \
    --conf "spark.sql.catalog.spark_catalog.io-
impl=org.apache.iceberg.aws.s3.S3FileIO" \
    --conf
 "spark.hadoop.fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.SimpleAWSCredentialPr
```

3. Data Catalog의 데이터를 쿼리합니다.

```
spark.sql("create database myicebergdb").show()
spark.sql("""CREATE TABLE myicebergdb.mytbl (name string) USING iceberg location
 's3://bucket_name/mytbl'""")
spark.sql("insert into myicebergdb.mytbl values('demo') ").show()
```

Amazon Redshift와 Apache Iceberg 간의 데이터 매핑

Redshift와 Iceberg는 다양한 데이터 유형을 지원합니다. 다음 호환성 매트릭스는 이러한 두 데이터 시스템 간에 데이터를 매핑할 때 지원되는 항목과 제한 사항을 간략하게 설명합니다. 각 데이터 시스템에서 지원되는 데이터 유형에 대한 자세한 내용은 Amazon Redshift 데이터 유형과 Apache Iceberg 테이블 사양을 참조하세요.

Redshift 데이터 유형	에일리어스	Iceberg 데이터 유형
SMALLINT	INT2	int
INTEGER	INT, INT4	int
BIGINT	INT8	long
DECIMAL	NUMERIC	decimal
REAL	FLOAT4	float
REAL	FLOAT4	float
DOUBLE PRECISION	FLOAT8, FLOAT	double
CHAR	CHARACTER, NCHAR	문자열
VARCHAR	CHARACTER VARYING, NVARCHAR	문자열
BPCHAR		문자열

Redshift 데이터 유형	에일리어스	Iceberg 데이터 유형
TEXT		문자열
날짜		date
TIME	TIME WITHOUT TIMEZONE	시간
TIME	TIME WITH TIMEZONE	지원되지 않음
TIMESTAMP	TIMESTAMP WITHOUT TIMEZONE	TIMESTAMP
TIMESTAMPZ	TIMESTAMP WITH TIMEZONE	TIMESTAMPZ
INTERVAL YEAR TO MONTH		지원되지 않음
INTERVAL DAY TO SECOND		지원되지 않음
BOOLEAN	BOOL	bool
HLLSKETCH		지원되지 않음
SUPER		지원되지 않음
VARBYTE	VARBINARY, BINARY VARYING	이진수
GEOMETRY		지원되지 않음
GEOGRAPHY		지원되지 않음

AWS Glue Iceberg REST 카탈로그 API 사용 시 고려 사항 및 제한 사항

다음은 Apache Iceberg REST 카탈로그 데이터 정의 언어(DDL) 작업 동작을 사용할 때 고려해야 할 사항과 제한 사항입니다.

고려 사항

- **`RenameTable`** API 동작 - `RenameTable` 작업은 Amazon Redshift의 테이블에서는 지원되지만 Amazon S3에서는 지원되지 않습니다.
- Amazon Redshift의 네임스페이스 및 테이블에 대한 DDL 작업 - Amazon Redshift의 네임스페이스 및 테이블에 대한 생성, 업데이트, 삭제 작업은 Amazon Redshift 관리형 작업 그룹을 언제 사용할 수 있는지 여부, 그리고 서로 충돌하는 DDL 및 DML 트랜잭션이 진행 중이고 작업이 잠금을 기다린 다음 변경 사항을 커밋해야 하는지 여부에 따라 달라지기 때문에 비동기 작업입니다.

제한 사항

- Apache Iceberg REST 사양의 보기 API는 AWS Glue Iceberg REST 카탈로그에서 지원되지 않습니다.

AWS Glue 데이터 카탈로그 모범 사례

이 섹션에서는 AWS Glue Data Catalog를 효과적으로 관리하고 활용하기 위한 모범 사례에 대해 다룹니다. 특히 효율적인 크롤러 사용법, 메타데이터 구성, 보안, 성능 최적화, 자동화, 데이터 거버넌스, 다른 AWS 서비스와의 통합 등의 사례를 강조합니다.

- 효과적인 크롤러 사용 - 정기적으로 크롤러를 실행하여 데이터 소스의 변경 사항을 반영하고 데이터 카탈로그를 최신 상태로 유지합니다. 자주 변경되는 데이터 소스에는 증분 크롤링을 사용하여 성능을 개선할 수 있습니다. 변경 사항이 감지되면 자동으로 새 파티션을 추가하거나 스키마를 업데이트하도록 크롤러를 구성하세요.
- 메타데이터 테이블 구성 및 이름 지정 - 데이터 카탈로그에서 데이터베이스 및 테이블에 대해 일관된 이름 지정 규칙을 설정합니다. 더 나은 구성을 위해 관련 데이터 소스를 논리적 데이터베이스 또는 폴더로 그룹화합니다. 각 테이블의 목적과 내용을 알 수 있는 설명이 포함된 이름을 사용합니다.
- 효과적인 스키마 관리 - AWS Glue 크롤러의 스키마 추론 기능을 활용합니다. 스키마 변경 사항을 적용하기 전에 검토하고 업데이트하여 다운스트림 애플리케이션이 중단되지 않도록 하세요. 스키마 진화 기능을 사용하면 스키마 변경 사항을 원활하게 처리할 수 있습니다.
- 데이터 카탈로그 보호 - 데이터 카탈로그에서 저장 및 전송 중인 데이터를 암호화할 수 있습니다. 세분화된 액세스 제어 정책을 구현하여 민감한 데이터에 대한 액세스를 제한하세요. 데이터 카탈로그 사용 권한 및 활동 로그를 정기적으로 감사하고 검토하세요.
- 다른 AWS 서비스와의 통합 - 데이터 카탈로그를 Amazon Athena, Redshift Spectrum, AWS Lake Formation 등과 같은 서비스를 위한 중앙 집중식 메타데이터 계층으로 사용하세요. AWS Glue ETL

작업을 활용하면 데이터를 변환하고 다양한 데이터 스토어로 로드하는 동시에 데이터 카탈로그에 메타데이터를 유지 관리할 수 있습니다.

- 성능 모니터링 및 최적화 - Amazon CloudWatch 메트릭을 사용하여 크롤러 및 ETL 작업의 성능을 모니터링하세요. 데이터 카탈로그에서 대규모 데이터세트를 분할하여 쿼리 성능을 개선하세요. 자주 액세스하는 메타데이터에 대한 성능 최적화를 구현하세요.
- AWS Glue 설명서 및 모범 사례에 대한 최신 정보 확인 - AWS Glue 설명서 및 AWS Glue 리소스에 대한 최신 업데이트, 모범 사례 및 권장 사항을 정기적으로 확인하세요. AWS Glue 웨비나, 워크숍 및 기타 이벤트에 참석하여 전문가로부터 배우고 새로운 특징과 기능에 대한 최신 정보를 받아보세요.

Amazon CloudWatch에서 Data Catalog 사용량 지표 모니터링

Amazon CloudWatch에서 AWS Glue Data Catalog 사용량 지표가 이제 제공되므로 Data Catalog의 리소스 사용률 모니터링 및 이해가 간소화됩니다. 이제 카탈로그, 데이터베이스, 테이블, 파티션 및 연결의 Glue Catalog API 사용량을 즉시 확인할 수 있으므로 Data Catalog를 더욱 쉽게 감독할 수 있습니다.

Data Catalog 지표의 개요

AWS Glue Data Catalog는 사용량 지표를 Amazon CloudWatch에 자동으로 게시합니다. CloudWatch 지표 통합을 사용하면 다음을 포함하여 1분마다 중요한 성능 지표를 추적할 수 있습니다.

- 테이블 요청
- 생성된 파티션 인덱스
- 업데이트된 연결
- 업데이트된 통계

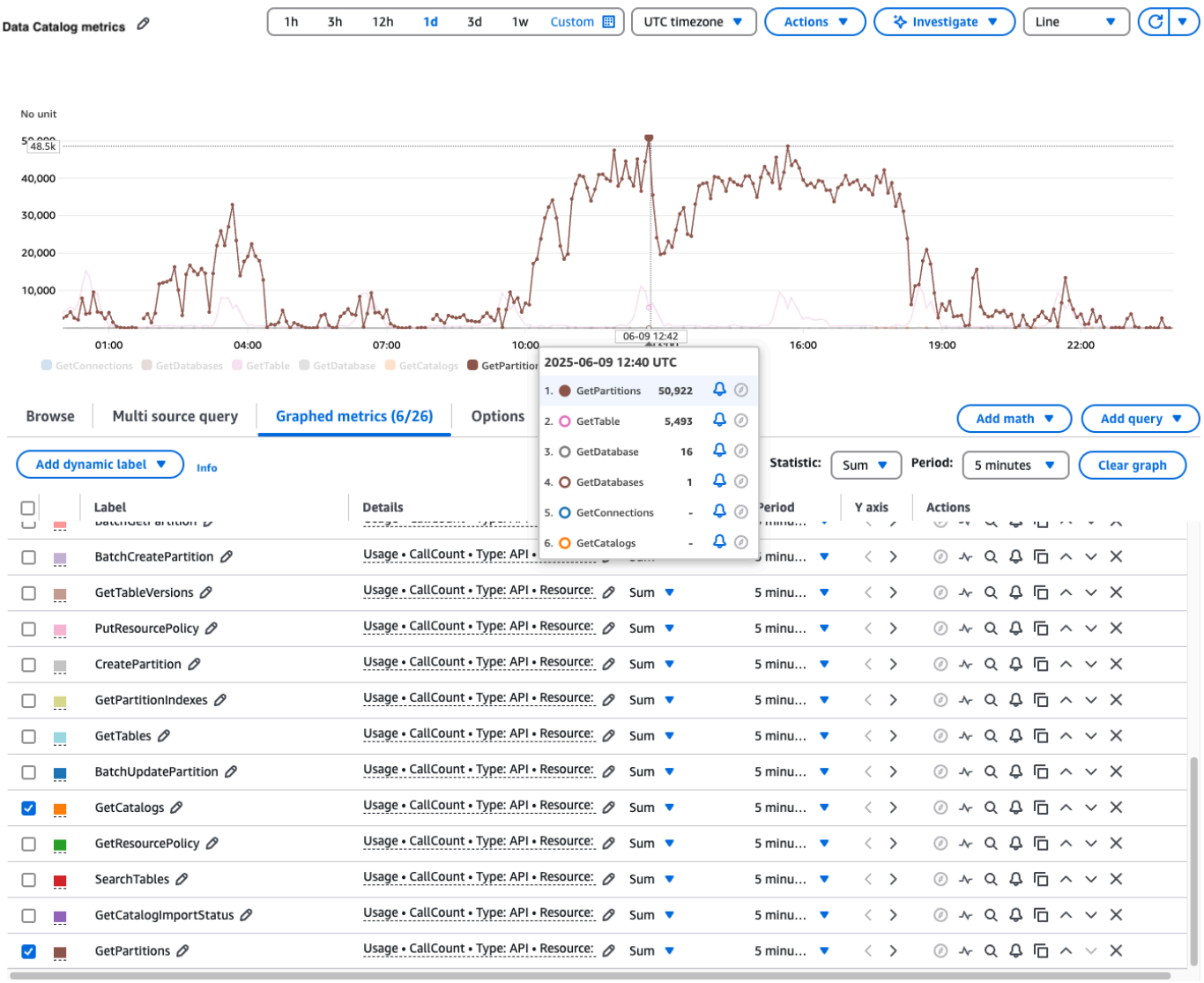
이러한 지표는 병목 현상을 식별하고, 이상을 감지하고, 데이터에 기반한 결정을 내려 전체 데이터 카탈로그의 신뢰성을 개선하는 데 도움이 됩니다. 지표가 지정된 임계값을 초과할 때 알림을 수신하도록 CloudWatch 경보를 설정하여 배포를 선제적으로 관리할 수도 있습니다.

CloudWatch 대시보드에 지표 추가

사용자 지정 대시보드를 생성하여 AWS Glue Data Catalog 리소스를 모니터링하고 경보를 설정하여 비정상적인 활동에 대한 알림을 받을 수 있습니다.

다음 단계를 따라 CloudWatch 대시보드에 Data Catalog 지표를 추가할 수 있습니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 모든 지표를 선택합니다.
4. 사용량>AWS 리소스 기준을 선택합니다.
5. Glue를 기준으로 필터링하여 사용 가능한 지표를 확인합니다.
6. 대시보드에 추가하려는 지표를 선택합니다.
7. 카탈로그, 데이터베이스, 테이블, 파티션 및 연결에 대한 지표를 CloudWatch 그래프에 추가합니다.



API 사용량이 정의된 임계값을 초과할 때 자동으로 트리거되는 사용자 지정 경보를 구성하여 데이터 카탈로그 사용량의 이상 징후를 식별할 수 있습니다.

경보 설정에 대한 구체적인 지침은 Metrics Insights CloudWatch 경보 생성을 참조하세요.

AWS Glue 스키마 레지스트리

> **Note**
>
> AWS Glue 콘솔의 아시아 태평양(자카르타) 및 중동(UAE) 리전에서는 AWS Glue 스키마 레지스트리가 지원되지 않습니다.

AWS Glue 스키마 레지스트리를 사용하면 데이터 스트림 스키마를 중앙에서 검색, 제어 및 발전시킬 수 있습니다. 스키마는 데이터 레코드의 구조와 포맷을 정의합니다. AWS Glue 스키마 레지스트리를 사용하면 Apache Kafka, Amazon Managed Streaming for Apache Kafka, Amazon Kinesis Data Streams, Apache Flink용 Amazon Managed Service for Apache Flink, AWS Lambda와의 편리한 통합을 사용하여 데이터 스트리밍 애플리케이션에서 스키마를 관리하고 적용할 수 있습니다.

스키마 레지스트리는 AVRO(v1.11.4) 데이터 형식, Everit 라이브러리를 사용한 JSON 스키마 검증이 포함된 스키마에 대한 JSON 스키마 포맷을 사용하는 JSON 데이터 형식(사양 Draft-04, Draft-06 및 Draft-07), `extensions` 또는 `groups`에 대한 지원이 없는 프로토콜 버퍼(Protobuf) 버전 proto2 및 proto3, 기타 데이터 형식과 언어를 제공할 예정인 Java 언어 지원 등을 지원합니다. 지원되는 기능에는 호환성, 메타데이터를 통한 스키마 소싱, 스키마 자동 등록, IAM 호환성, 저장 및 데이터 전송을 줄이기 위한 선택적 ZLIB 압축이 포함됩니다. 스키마 레지스트리는 서버리스이며 무료입니다.

생산자와 소비자 간의 데이터 포맷 계약으로 스키마를 사용하면 데이터 거버넌스가 개선되고 데이터 품질이 향상되며 데이터 소비자가 호환 가능한 업스트림 변경 사항에 탄력적으로 대처할 수 있습니다.

스키마 레지스트리를 사용하면 직렬화 및 역직렬화를 위해 서로 다른 시스템이 스키마를 공유할 수 있습니다. 예를 들어 데이터 생산자와 소비자가 있다고 가정합니다. 생산자는 데이터를 게시할 때 스키마를 알고 있습니다. Schema Registry는 Amazon MSK 또는 Apache Kafka와 같은 특정 시스템용 serializer 및 deserializer를 제공합니다.

자세한 내용은 스키마 레지스트리 작동 방식 섹션을 참조하세요.

주제

- 스키마
- 레지스트리
- 스키마 버전 관리 및 호환성

- 오픈 소스 Serde 라이브러리
- Schema Registry의 할당량
- 스키마 레지스트리 작동 방식
- 스키마 레지스트리 시작하기
- AWS Glue Schema Registry와 통합
- 서드 파티 Schema Registry에서 AWS Glue Schema Registry로 마이그레이션

스키마

스키마는 데이터 레코드의 구조와 포맷을 정의합니다. 스키마는 신뢰할 수 있는 데이터 게시, 소비 또는 저장을 위한 버전 지정 사양입니다.

Avro에 대한 이 예제 스키마에서 포맷과 구조는 레이아웃 및 필드 이름으로 정의되고 필드 이름 포맷은 데이터 유형(예: `string`, `int`)으로 정의됩니다.

```
{
    "type": "record",
    "namespace": "ABC_Organization",
    "name": "Employee",
    "fields": [
        {
            "name": "Name",
            "type": "string"
        },
        {
            "name": "Age",
            "type": "int"
        },
        {
            "name": "address",
            "type": {
                "type": "record",
                "name": "addressRecord",
                "fields": [
                    {
                        "name": "street",
                        "type": "string"
                    },
                    {
```

```
                        "name": "zipcode",
                        "type": "int"
                    }
                ]
            }
        }
    ]
}
```

이 JSON용 JSON 스키마 초안-07 예에서 포맷은 JSON 스키마 조직에서 정의합니다.

```
{
 "$id": "https://example.com/person.schema.json",
 "$schema": "http://json-schema.org/draft-07/schema#",
 "title": "Person",
 "type": "object",
 "properties": {
  "firstName": {
   "type": "string",
   "description": "The person's first name."
  },
  "lastName": {
   "type": "string",
   "description": "The person's last name."
  },
  "age": {
   "description": "Age in years which must be equal to or greater than zero.",
   "type": "integer",
   "minimum": 0
  }
 }
}
```

Protobuf에 대한 이 예제에서는 프로토콜 버퍼 언어 버전 2(proto2)로 형식을 정의합니다.

```
syntax = "proto2";

package tutorial;

option java_multiple_files = true;
option java_package = "com.example.tutorial.protos";
option java_outer_classname = "AddressBookProtos";
```

```
message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    optional string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

레지스트리

레지스트리는 스키마의 논리적 컨테이너입니다. 레지스트리를 사용하면 스키마를 구성하고 애플리케이션에 대한 액세스 제어를 관리할 수 있습니다. 레지스트리에는 레지스트리 내의 스키마 작업에 대한 다양한 액세스 권한을 구성하고 설정할 수 있는 Amazon 리소스 이름(ARN)이 있습니다.

기본 레지스트리를 사용하거나 필요한 만큼 새 레지스트리를 생성할 수 있습니다.

AWS Glue Schema Registry 계층 구조

- RegistryName: [string]
  - RegistryArn: [AWS ARN]
  - CreatedTime: [timestamp]
  - UpdatedTime: [timestamp]
- SchemaName: [string]
  - SchemaArn: [AWS ARN]

  - DataFormat: [Avro, Json 또는 Protobuf]
  - Compatibility: [예: BACKWARD, BACKWARD_ALL, FORWARD, FORWARD_ALL, FULL, FULL_ALL, NONE, DISABLED]
  - Status: [예: PENDING, AVAILABLE, DELETING]
  - SchemaCheckpoint: [integer]
  - CreatedTime: [timestamp]
  - UpdatedTime: [timestamp]
- SchemaVersion: [string]
  - SchemaVersionNumber: [integer]
  - Status: [예: PENDING, AVAILABLE, DELETING]
  - SchemaDefinition: [string, Value: JSON]
  - CreatedTime: [timestamp]
- SchemaVersionMetadata: [list]
  - MetadataKey: [string]
  - MetadataInfo
  - MetadataValue: [string]
  - CreatedTime: [timestamp]

스키마 버전 관리 및 호환성

각 스키마에는 여러 버전이 있을 수 있습니다. 버전 관리는 스키마에 적용되는 호환성 규칙에 의해 관리됩니다. 새 스키마 버전 등록 요청은 성공하기 전에 Schema Registry에서 이 규칙에 대해 검사합니다.

체크포인트로 표시된 스키마 버전은 스키마의 새 버전 등록 호환성을 결정하는 데 사용됩니다. 스키마가 처음 생성되면 기본 체크포인트가 첫 번째 버전이 됩니다. 스키마가 더 많은 버전으로 발전함에 따라 CLI/SDK를 사용하여 일련의 제약 조건을 준수하는 `UpdateSchema` API를 사용하는 스키마 버전으로 체크포인트를 변경할 수 있습니다. 콘솔에서 스키마 정의 또는 호환성 모드를 편집하면 기본적으로 체크포인트가 최신 버전으로 변경됩니다.

호환성 모드를 사용하면 시간이 지남에 따라 스키마가 어떻게 발전할 수 있는지 여부를 제어할 수 있습니다. 이러한 모드는 데이터를 생성하고 소비하는 애플리케이션 간의 계약을 형성합니다. 새 버전의 스키마가 레지스트리에 제출되면 스키마 이름에 적용된 호환성 규칙을 사용하여 새 버전을 수락

할 수 있는지 여부를 결정합니다. NONE, DISABLED, BACKWARD, BACKWARD_ALL, FORWARD, FORWARD_ALL, FULL, FULL_ALL의 8가지 호환 모드가 있습니다.

Avro 데이터 포맷에서 필드는 선택 사항이거나 필수일 수 있습니다. 선택적 필드는 Type이 null을 포함하는 필드입니다. 필수 필드에는 Type으로 null이 없습니다.

Protobuf 데이터 형식에서 필드가 proto2 구문에서는 선택 사항(반복 포함)이거나 필수일 수 있지만 proto3 구문에서는 모든 필드가 선택 사항입니다(반복 포함). 모든 호환성 규칙은 프로토콜 버퍼 사양에 대한 이해와 Google 프로토콜 버퍼 설명서의 지침에 따라 결정됩니다.

- NONE: 호환 모드가 적용되지 않습니다. 개발 시나리오에서 또는 스키마에 적용할 호환성 모드를 모르는 경우 이 선택 사항을 사용할 수 있습니다. 추가된 모든 새 버전은 호환성 검사를 거치지 않고 수락됩니다.
- DISABLED: 이 호환성 선택 항목은 특정 스키마에 대한 버전 관리를 방지합니다. 새 버전을 추가할 수 없습니다.
- BACKWARD: 이 호환성 선택 항목은 소비자가 현재 및 이전 스키마 버전을 모두 읽을 수 있도록 하므로 권장됩니다. 이 선택 항목을 사용하여 필드를 삭제하거나 선택적 필드를 추가할 때 이전 스키마 버전과의 호환성을 확인할 수 있습니다. BACKWARD의 일반적인 사용 사례는 애플리케이션이 가장 최근 스키마에 대해 생성된 경우입니다.

  AVRO

  예를 들어 이름(필수), 성(필수), 이메일(필수) 및 전화번호(선택 사항)로 정의된 스키마가 있다고 가정합니다.

  다음 스키마 버전에서 필수 이메일 필드를 제거하면 성공적으로 등록됩니다. BACKWARD 호환성을 위해서는 소비자가 현재 및 이전 스키마 버전을 읽을 수 있어야 합니다. 이전 메시지의 추가 이메일 필드가 무시되므로 소비자는 새 스키마를 읽을 수 있습니다.

  필수 필드를 추가하는 제안된 새 스키마 버전이 있는 경우(예: 우편 번호) 이는 BACKWARD 호환성으로 성공적으로 등록되지 않습니다. 새 버전의 소비자는 필수 우편 번호 필드가 누락되어 스키마 변경 전에 이전 메시지를 읽을 수 없습니다. 그러나 새 스키마에서 우편 번호 필드가 선택 사항으로 설정된 경우 소비자가 선택적 우편 번호 필드 없이 이전 스키마를 읽을 수 있으므로 제안된 버전이 성공적으로 등록됩니다.

  JSON

  예를 들어 이름(선택 사항), 성(선택 사항), 이메일(선택 사항) 및 전화 번호(선택 사항)로 정의된 스키마 버전이 있다고 가정합니다.

다음 스키마 버전이 선택적 전화번호 속성을 추가하는 경우 원래 스키마 버전이 `additionalProperties` 필드를 false로 설정하여 추가 속성을 허용하지 않는 한 성공적으로 등록됩니다. BACKWARD 호환성을 위해서는 소비자가 현재 및 이전 스키마 버전을 읽을 수 있어야 합니다. 소비자는 전화번호 속성이 존재하지 않는 원래 스키마로 생성된 데이터를 읽을 수 있습니다.

선택적 전화 번호 속성을 추가하는 제안된 새 스키마 버전이 있는 경우 원래 스키마 버전이 `additionalProperties` 필드를 true로 설정하면(즉, 추가 속성을 허용할 때) BACKWARD 호환성으로 성공적으로 등록되지 않습니다. 새 버전의 소비자는 다른 유형(예: 숫자 대신 문자열)의 전화번호 속성이 있는 데이터를 읽을 수 없으므로 스키마가 변경되기 전에 이전 메시지를 읽을 수 없습니다.

PROTOBUF

예를 들어 proto2 구문에서 `first name`(필수), `last name`(필수), `email`(필수), phone number(선택) 필드를 포함하는 Message `Person`으로 정의된 스키마 버전이 있다고 가정합니다.

AVRO 시나리오에서처럼 다음 스키마 버전에서 필수 `email` 필드를 제거하면 성공적으로 등록됩니다. BACKWARD 호환성을 위해서는 소비자가 현재 및 이전 스키마 버전을 읽을 수 있어야 합니다. 이전 메시지의 추가 `email` 필드가 무시되므로 소비자는 새 스키마를 읽을 수 있습니다.

필수 항목(예: `zip code`)을 추가하는 제안된 새 스키마 버전이 있는 경우, 이는 이전 버전과의 호환성으로 성공적으로 등록되지 않습니다. 새 버전의 소비자는 필수 `zip code` 필드가 누락되어 스키마 변경 전에 이전 메시지를 읽을 수 없습니다. 그러나 새 스키마에서 `zip code` 필드가 선택 사항으로 설정된 경우 소비자가 선택적 `zip code` 필드 없이 이전 스키마를 읽을 수 있으므로 제안된 버전이 성공적으로 등록됩니다.

gRPC 사용 사례의 경우 새 RPC 서비스 또는 RPC 메서드 추가는 이전 버전과 호환되는 변경 사항입니다. 예를 들어 RPC 서비스 `MyService`에서 두 개의 RPC 메서드 `Foo` 및 `Bar`를 사용하여 정의한 스키마 버전이 있다고 가정합니다.

다음 스키마 버전에서 `Baz`라는 새 RPC 메서드를 추가하는 경우 성공적으로 등록됩니다. 새로 추가된 RPC 메서드 `Baz`가 선택 사항이므로 소비자는 이전 버전과의 호환성에 따라 원래 스키마로 생성된 데이터를 읽을 수 있습니다.

기존 RPC 메서드 `Foo`를 제거하는 제안된 새 스키마 버전이 있는 경우, 이는 이전 버전과의 호환성으로 성공적으로 등록되지 않습니다. 새 버전의 소비자는 gRPC 애플리케이션에서 RPC 메서드 `Foo`가 없는 데이터를 이해하거나 읽을 수 없으므로 스키마 변경 전에 이전 메시지를 읽을 수 없습니다.

- BACKWARD_ALL: 이 호환성 선택을 통해 소비자는 현재 및 모든 이전 스키마 버전을 모두 읽을 수 있습니다. 이 선택 항목을 사용하여 필드를 삭제하거나 선택적 필드를 추가할 때 모든 이전 스키마 버전과의 호환성을 확인할 수 있습니다.
- FORWARD: 이 호환성 선택을 통해 소비자는 현재 및 후속 스키마 버전을 모두 읽을 수 있지만 반드시 이후 버전은 아닙니다. 이 선택 항목 사용하여 필드를 추가하거나 선택적 필드를 삭제할 때 마지막 스키마 버전과의 호환성을 확인할 수 있습니다. FORWARD의 일반적인 사용 사례는 애플리케이션이 이전 스키마에 대해 생성되었고 더 최신 스키마를 처리할 수 있어야 하는 경우입니다.

  AVRO

  예를 들어 이름(필수), 성(필수), 이메일(선택 사항)로 정의된 스키마 버전이 있다고 가정합니다.

  필수 항목(예: 전화번호)을 추가하는 새 스키마 버전이 있는 경우 성공적으로 등록됩니다. FORWARD 호환성을 위해서는 소비자가 이전 버전을 사용하여 새 스키마로 생성된 데이터를 읽을 수 있어야 합니다.

  필수 이름 필드를 삭제하는 제안된 스키마 버전이 있는 경우 FORWARD 호환성으로 성공적으로 등록되지 않습니다. 이전 버전의 소비자는 필수 이름 필드가 누락되어 제안된 스키마를 읽을 수 없습니다. 그러나 이름 필드가 원래 선택 사항인 경우 제안된 새 스키마는 선택 사항인 이름 필드가 없는 새 스키마를 기반으로 소비자가 데이터를 읽을 수 있으므로 성공적으로 등록됩니다.

  JSON

  예를 들어 이름(선택 사항), 성(선택 사항), 이메일(선택 사항) 및 전화 번호(선택 사항)로 정의된 스키마 버전이 있다고 가정합니다.

  선택적 전화 번호 속성을 제거하는 새 스키마 버전이 있는 경우 새 스키마 버전에서 `additionalProperties` 필드를 false로 설정하여 추가 속성을 허용하지 않는 한 성공적으로 등록됩니다. FORWARD 호환성을 위해서는 소비자가 이전 버전을 사용하여 새 스키마로 생성된 데이터를 읽을 수 있어야 합니다.

  선택적 전화번호 속성을 삭제하는 제안된 스키마 버전이 있는 경우 새 스키마 버전이 `additionalProperties` 필드를 true로 설정하면 즉, 추가 속성을 허용할 때 FORWARD 호환성으로 성공적으로 등록되지 않습니다. 이전 버전의 소비자는 다른 유형의 전화번호 속성(예: 숫자 대신 문자열)을 가질 수 있으므로 제안된 스키마를 읽을 수 없습니다.

  PROTOBUF

  예를 들어 proto2 구문에서 `first name`(필수), `last name`(필수), `email`(선택 사항) 필드를 포함하는 Message `Person`으로 정의된 스키마 버전이 있다고 가정합니다.

AVRO 시나리오처럼 필수 항목(예: phone number)을 추가하는 새 스키마 버전이 있는 경우 성공적으로 등록됩니다. FORWARD 호환성을 위해서는 소비자가 이전 버전을 사용하여 새 스키마로 생성된 데이터를 읽을 수 있어야 합니다.

필수 first name 필드를 삭제하는 제안된 스키마 버전이 있는 경우 FORWARD 호환성으로 성공적으로 등록되지 않습니다. 이전 버전의 소비자는 필수 first name 필드가 누락되어 제안된 스키마를 읽을 수 없습니다. 그러나 first name 필드가 원래 선택 사항인 경우 제안된 새 스키마는 선택 사항인 first name 필드가 없는 새 스키마를 기반으로 소비자가 데이터를 읽을 수 있으므로 성공적으로 등록됩니다.

gRPC 사용 사례의 경우 RPC 서비스 또는 RPC 메서드 제거는 다음 버전과 호환되는 변경 사항입니다. 예를 들어 RPC 서비스 MyService에서 두 개의 RPC 메서드 Foo 및 Bar를 사용하여 정의한 스키마 버전이 있다고 가정합니다.

다음 스키마 버전에서 Foo라는 기존 RPC 메서드를 삭제하는 경우 소비자가 이전 버전을 사용하여 새 스키마로 생성된 데이터를 읽을 수 있으므로 FORWARD 호환성에 따라 성공적으로 등록됩니다. Baz라는 RPC 메서드를 추가하는 제안된 새 스키마 버전이 있는 경우 FORWARD 호환성으로 성공적으로 등록되지 않습니다. 이전 버전의 소비자는 Baz라는 RPC 메서드가 누락되어 제안된 스키마를 읽을 수 없습니다.

- FORWARD_ALL: 이 호환성 선택을 통해 소비자는 새로 등록된 스키마의 생산자가 작성한 데이터를 읽을 수 있습니다. 필드를 추가하거나 선택적 필드를 삭제하고 모든 이전 스키마 버전과의 호환성을 확인해야 할 때 이 선택 사항을 사용할 수 있습니다.
- FULL: 이 호환성 선택을 통해 소비자는 이전 또는 다음 버전의 스키마를 사용하여 생산자가 작성한 데이터를 읽을 수 있지만 이전 또는 이후 버전은 아닙니다. 이 선택 항목 사용하여 선택적 필드를 추가하거나 제거할 때 마지막 스키마 버전과의 호환성을 확인할 수 있습니다.
- FULL_ALL: 이 호환성 선택을 통해 소비자는 모든 이전 스키마 버전을 사용하여 생산자가 작성한 데이터를 읽을 수 있습니다. 이 선택 항목을 사용하여 선택적 필드를 추가하거나 제거할 때 모든 이전 스키마 버전과의 호환성을 확인할 수 있습니다.

오픈 소스 Serde 라이브러리

AWS는 데이터 직렬화 및 역직렬화를 위한 프레임워크로 오픈 소스 Serde 라이브러리를 제공합니다. 이러한 라이브러리의 오픈 소스 설계를 통해 일반적인 오픈 소스 애플리케이션 및 프레임워크가 프로젝트에서 이러한 라이브러리를 지원할 수 있습니다.

Serde 라이브러리의 작동 방식에 대한 자세한 내용은 스키마 레지스트리 작동 방식 섹션을 참조하세요.

Schema Registry의 할당량

AWS에서 제한이라고도 하는 할당량은 AWS 계정의 리소스, 작업, 항목의 최댓값입니다. 다음은 AWS Glue의 Schema Registry에 대한 소프트 제한입니다.

스키마 버전 메타데이터 키-값 페어

AWS 리전별로 SchemaVersion당 최대 10개의 키-값 페어가 있을 수 있습니다.

QuerySchemaVersionMetadata 작업(Python: query_schema_version_metadata) 또는 PutSchemaVersionMetadata 작업(Python: put_schema_version_metadata) API를 사용하여 키-값 메타데이터 페어를 보거나 설정할 수 있습니다.

다음은 AWS Glue의 Schema Registry에 대한 하드 제한입니다.

레지스트리

이 계정의 경우 AWS 리전당 최대 100개의 레지스트리를 보유할 수 있습니다.

SchemaVersion

이 계정에 대해 AWS 리전당 최대 10,000개의 스키마 버전을 보유할 수 있습니다.

각 새 스키마는 새 스키마 버전을 생성하므로 각 스키마에 버전이 하나만 있는 경우 이론상 지역별로 계정당 최대 10,000개의 스키마를 보유할 수 있습니다.

스키마 페이로드

스키마 페이로드의 크기 제한은 170KB입니다.

스키마 레지스트리 작동 방식

이 섹션에서는 스키마 레지스트리의 직렬화 및 역직렬화 프로세스의 작동 방식에 대해 설명합니다.

1. 스키마 등록: 스키마가 레지스트리에 이미 존재하지 않는 경우 대상 이름과 동일한 스키마 이름(예: test_topic, test_stream, prod_firehose)으로 스키마를 등록하거나 생산자가 스키마에 대한 사용자 정의 이름을 제공할 수 있습니다. 생산자는 또한 소스: msk_kafka_topic_A와 같은 메타데이터로 스키마에 키-값 페어를 추가하거나 스키마 생성 시 스키마에 AWS 태그를 적용할 수 있습니다. 스키마가 등록되면 Schema Registry는 스키마 버전 ID를 serializer에 반환합니다. 스키마가 존재하지만

serializer가 존재하지 않는 새 버전을 사용하는 경우 Schema Registry는 스키마 참조를 호환성 규칙으로 확인하여 새 버전을 새 버전으로 등록하기 전에 새 버전이 호환되는지 확인합니다.

스키마를 등록하는 방법에는 수동 등록과 자동 등록의 두 가지가 있습니다. AWS Glue 콘솔이나 CLI/SDK를 통해 수동으로 스키마를 등록할 수 있습니다.

serializer 설정에서 자동 등록이 켜져 있으면 스키마의 자동 등록이 수행됩니다. 생산자 구성에 `REGISTRY_NAME`이 제공되지 않으면 자동 등록은 기본 레지스트리(default-registry) 아래에 새 스키마 버전을 등록합니다. 자동 등록 속성 지정에 대한 자세한 내용은 SerDe 라이브러리 설치 섹션을 참조하세요.

2. Serializer는 스키마에 대해 데이터 레코드를 검증합니다. 데이터를 생성하는 애플리케이션이 해당 스키마를 등록한 경우 Schema Registry serializer는 애플리케이션이 생성하는 레코드가 등록된 스키마와 일치하는 필드 및 데이터 유형으로 구성되었는지 확인합니다. 레코드의 스키마가 등록된 스키마와 일치하지 않는 경우 serializer는 예외를 반환하고 애플리케이션은 레코드를 대상으로 전달하지 못합니다.

   스키마가 없고 생산자 구성을 통해 스키마 이름이 제공되지 않은 경우 스키마는 주제 이름(Apache Kafka 또는 Amazon MSK의 경우) 또는 스트림 이름(Kinesis Data Streams의 경우)과 동일한 이름으로 생성됩니다.

   모든 레코드에는 스키마 정의와 데이터가 있습니다. 스키마 정의는 Schema Registry의 기존 스키마 및 버전에 대해 쿼리됩니다.

   기본적으로 생산자는 등록된 스키마의 스키마 정의 및 스키마 버전 ID를 캐시합니다. 레코드의 스키마 버전 정의가 캐시에서 사용 가능한 것과 일치하지 않는 경우 생산자는 Schema Registry를 사용하여 스키마를 검증하려고 시도합니다. 스키마 버전이 유효하면 해당 버전 ID와 정의가 생산자에서 로컬로 캐시됩니다.

   SerDe 라이브러리 설치의 3단계에서 선택적 생산자 속성 내에서 기본 캐시 기간(24시간)을 조정할 수 있습니다.

3. 레코드 직렬화 및 전달: 레코드가 스키마를 준수하는 경우 serializer는 각 레코드를 스키마 버전 ID로 장식하고 선택한 데이터 포맷(AVRO, JSON, Protobuf 또는 다른 포맷 곧 제공 예정)을 기반으로 레코드를 직렬화하고 레코드를 압축(선택적 생산자 구성), 목적지로 전달합니다.
4. 소비자는 데이터를 역직렬화합니다. 이 데이터를 읽는 소비자는 레코드 페이로드에서 스키마 버전 ID를 구문 분석하는 Schema Registry deserializer 라이브러리를 사용합니다.
5. Deserializer는 Schema Registry에서 스키마를 요청할 수 있습니다. Deserializer가 특정 스키마 버전 ID를 가진 레코드를 처음 본 경우에는 스키마 버전 ID를 사용하여 deserializer가 Schema Registry에서 스키마를 요청하고 소비자에서 로컬로 스키마를 캐시합니다. Schema Registry가 레코

드를 역직렬화할 수 없는 경우 소비자는 레코드에서 데이터를 기록하고 계속 진행하거나 애플리케이션을 중지할 수 있습니다.

6. Deserializer는 스키마를 사용하여 레코드를 역직렬화합니다. Deserializer가 Schema Registry에서 스키마 버전 ID를 검색할 때 deserializer는 레코드를 압축 해제하고(생산자가 보낸 레코드가 압축된 경우) 스키마를 사용하여 레코드를 역직렬화합니다. 이제 애플리케이션이 레코드를 처리합니다.

> **Note**
>
> 암호화: 클라이언트는 HTTPS를 통한 TLS 암호화를 사용하여 전송 중 데이터를 암호화하는 API 호출을 통해 Schema Registry와 통신합니다. Schema Registry에 저장된 스키마는 항상 서비스 관리형 AWS Key Management Service(AWS KMS) 키를 사용하여 저장 시 암호화됩니다.

> **Note**
>
> 사용자 권한 부여: 스키마 레지스트리는 자격 증명 기반 IAM 정책을 지원합니다.

스키마 레지스트리 시작하기

다음 섹션은 Schema Registry를 설정하고 사용하는 방법에 대한 개요 및 설명입니다. 스키마 레지스트리 개념 및 구성 요소에 대한 자세한 내용은 AWS Glue 스키마 레지스트리 섹션을 참조하세요.

주제

- SerDe 라이브러리 설치
- 레지스트리 생성
- 스키마 생성
- 스키마 또는 레지스트리 업데이트
- 스키마 또는 레지스트리 삭제
- Serializer에 대한 IAM 예
- Deserializer에 대한 IAM 예
- AWS PrivateLink를 사용한 프라이빗 연결
- Amazon CloudWatch 지표에 액세스
- 스키마 레지스트리용 샘플 AWS CloudFormation 템플릿

SerDe 라이브러리 설치

> **Note**
>
> 전제 조건: 다음 단계를 완료하기 전에 Amazon Managed Streaming for Apache Kafka(Amazon MSK) 또는 Apache Kafka 클러스터가 실행 중이어야 합니다. 생산자와 소비자는 Java 8 이상에서 실행 중이어야 합니다.

SerDe 라이브러리는 데이터 직렬화 및 역직렬화를 위한 프레임워크를 제공합니다.

데이터를 생성하는 애플리케이션("serializer"로 통칭)을 위한 오픈 소스 serializer를 설치합니다. Serializer는 직렬화, 압축 및 Schema Registry와의 상호 작용을 처리합니다. Serializer는 Amazon MSK와 같은 Schema Registry 호환 대상에 기록되는 레코드에서 스키마를 자동으로 추출합니다. 마찬가지로 데이터를 사용하는 애플리케이션에 오픈 소스 deserializer를 설치합니다.

생산자와 소비자에 라이브러리를 설치하려면

1. 생산자와 소비자의 pom.xml 파일 내에서 아래 코드를 통해 이 종속성을 추가합니다.

```
<dependency>
    <groupId>software.amazon.glue</groupId>
    <artifactId>schema-registry-serde</artifactId>
    <version>1.1.5</version>
</dependency>
```

   또는 AWS Glue Schema Registry Github 리포지토리를 복제할 수 있습니다.

2. 다음 필수 속성으로 생산자를 설정합니다.

```
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
 StringSerializer.class.getName()); // Can replace StringSerializer.class.getName())
 with any other key serializer that you may use
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
 GlueSchemaRegistryKafkaSerializer.class.getName());
props.put(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2");
properties.put(AWSSchemaRegistryConstants.DATA_FORMAT, "JSON"); // OR "AVRO"
```

   기존 스키마가 없으면 자동 등록을 설정해야 합니다(다음 단계). 적용하려는 스키마가 있는 경우 "my-schema"를 스키마 이름으로 바꿉니다. 또한 스키마 자동 등록이 꺼져 있는 경우 "registry-

name"을 제공해야 합니다. 스키마가 "default-registry" 아래에 생성된 경우 레지스트리 이름을 생략할 수 있습니다.

3. (선택 사항) 이러한 선택적 생산자 속성을 설정합니다. 자세한 속성 설명은 ReadMe 파일을 참조하세요.

```
props.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, "true"); // If
 not passed, uses "false"
props.put(AWSSchemaRegistryConstants.SCHEMA_NAME, "my-schema"); // If not passed,
 uses transport name (topic name in case of Kafka, or stream name in case of Kinesis
 Data Streams)
props.put(AWSSchemaRegistryConstants.REGISTRY_NAME, "my-registry"); // If not passed,
 uses "default-registry"
props.put(AWSSchemaRegistryConstants.CACHE_TIME_TO_LIVE_MILLIS, "86400000"); // If
 not passed, uses 86400000 (24 Hours)
props.put(AWSSchemaRegistryConstants.CACHE_SIZE, "10"); // default value is 200
props.put(AWSSchemaRegistryConstants.COMPATIBILITY_SETTING, Compatibility.FULL); //
 Pass a compatibility mode. If not passed, uses Compatibility.BACKWARD
props.put(AWSSchemaRegistryConstants.DESCRIPTION, "This registry is used for several
 purposes."); // If not passed, constructs a description
props.put(AWSSchemaRegistryConstants.COMPRESSION_TYPE,
 AWSSchemaRegistryConstants.COMPRESSION.ZLIB); // If not passed, records are sent
 uncompressed
```

자동 등록은 기본 레지스트리("default-registry") 아래에 스키마 버전을 등록합니다. 이전 단계에서 SCHEMA_NAME을 지정하지 않으면 주제 이름이 SCHEMA_NAME으로 유추됩니다.

호환성 모드에 대한 자세한 내용은 스키마 버전 관리 및 호환성 섹션을 참조하세요.

4. 다음 필수 속성으로 소비자를 설정합니다.

```
props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 StringDeserializer.class.getName());
props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 GlueSchemaRegistryKafkaDeserializer.class.getName());
props.put(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2"); // Pass an AWS ##
props.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName()); // Only required for AVRO data format
```

5. (선택 사항) 이러한 선택적 소비자 속성을 설정합니다. 자세한 속성 설명은 ReadMe 파일을 참조하세요.

```
properties.put(AWSSchemaRegistryConstants.CACHE_TIME_TO_LIVE_MILLIS, "86400000"); //
 If not passed, uses 86400000
props.put(AWSSchemaRegistryConstants.CACHE_SIZE, "10"); // default value is 200
props.put(AWSSchemaRegistryConstants.SECONDARY_DESERIALIZER,
 "com.amazonaws.services.schemaregistry.deserializers.external.ThirdPartyDeserializer"); //
 For migration fall back scenario
```

레지스트리 생성

기본 레지스트리를 사용하거나 AWS Glue API 또는 AWS Glue 콘솔을 사용하여 필요한 만큼 새 레지스트리를 생성할 수 있습니다.

AWS Glue API

다음 단계를 사용하여 AWS Glue API로 이 태스크를 수행할 수 있습니다.

AWS Glue Schema Registry API에 AWS CLI를 사용하려면 AWS CLI를 최신 버전으로 업데이트해야 합니다.

새 레지스트리를 추가하려면 CreateRegistry 작업(Python: create_registry) API를 사용합니다. 생성할 레지스트리의 이름으로 `RegistryName`을 지정합니다. 최대 길이는 255이며 문자, 숫자, 하이픈, 밑줄, 달러 기호 또는 해시 표시만 포함합니다.

URI 주소 여러 줄 문자열 패턴과 일치하는 2048바이트 이하의 문자열로 `Description`을 지정합니다.

선택적으로 키-값 페어의 맵 배열로 레지스트리에 대해 하나 이상의 `Tags`를 지정합니다.

```
aws glue create-registry --registry-name registryName1 --description description
```

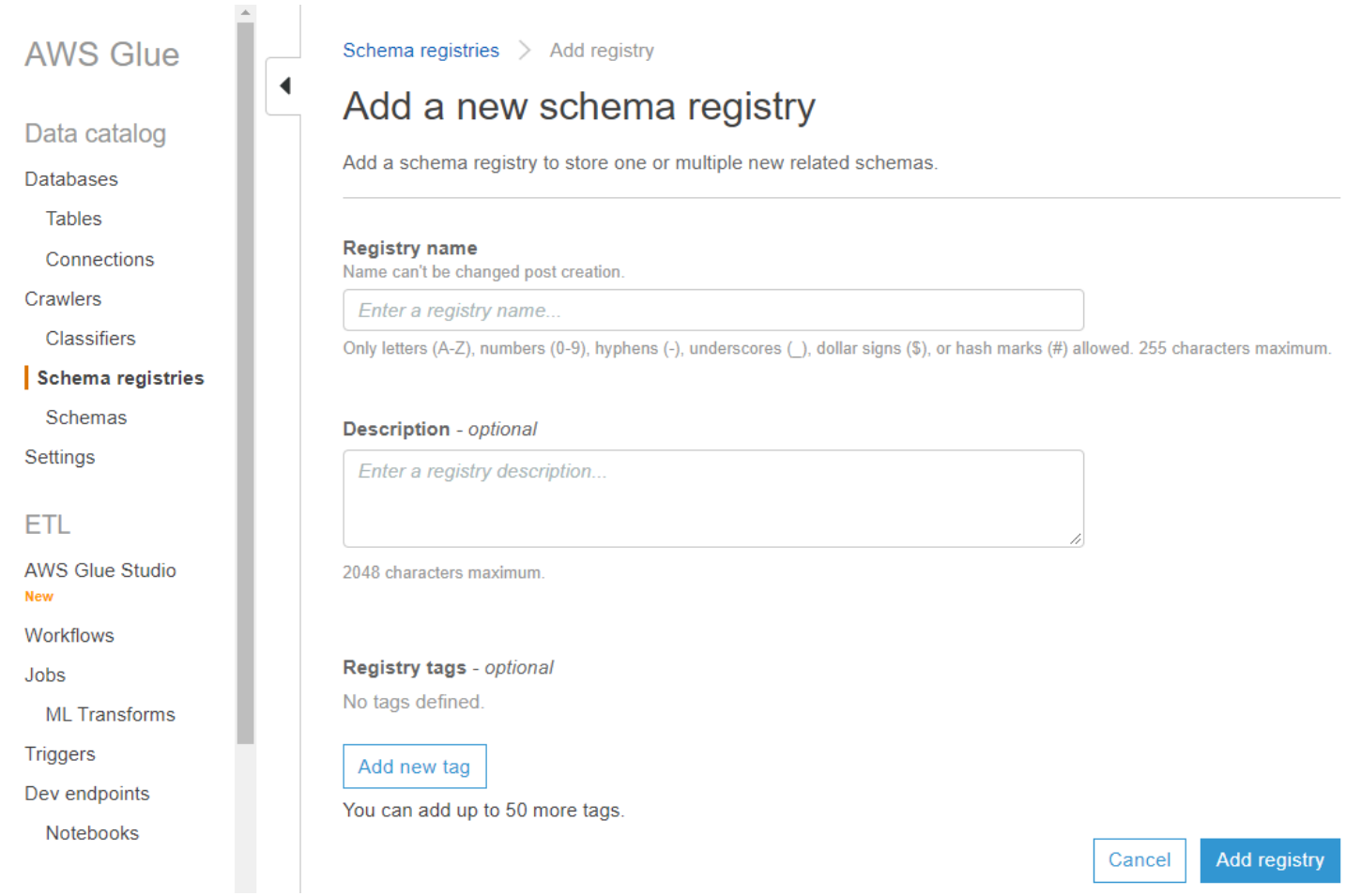
레지스트리가 생성되면 API 응답의 `RegistryArn`에서 볼 수 있는 Amazon 리소스 이름(ARN)이 할당됩니다. 이제 레지스트리를 생성했으므로 해당 레지스트리에 대해 하나 이상의 스키마를 생성합니다.

AWS Glue 콘솔

AWS Glue 콘솔에서 새 레지스트리를 추가하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [스키마 레지스트리(Schema registries)]를 선택합니다.
3. [레지스트리 추가(Add registry)]를 선택합니다.
4. 문자, 숫자, 하이픈 또는 밑줄로 구성된 레지스트리의 [레지스트리 이름(Registry name)]을 입력합니다. 이 이름은 변경할 수 없습니다.
5. 레지스트리에 대한 [설명(Description)](선택 사항)을 입력합니다.
6. 선택적으로 레지스트리에 하나 이상의 태그를 적용합니다. [새 태그 추가(Add new tag)]를 선택하고 [태그 키(Tag key)]와 [태그 값(Tag value)](선택 사항)을 지정합니다.
7. [레지스트리 추가(Add registry)]를 선택합니다.



레지스트리가 생성되면 [스키마 레지스트리(Schema registries)]의 목록에서 레지스트리를 선택하여 볼 수 있는 Amazon 리소스 이름(ARN)이 할당됩니다. 이제 레지스트리를 생성했으므로 해당 레지스트리에 대해 하나 이상의 스키마를 생성합니다.

JSON에 대한 특정 레코드(JAVA POJO) 처리

POJO(Plain Old Java Object)를 사용하고 객체를 레코드로 전달할 수 있습니다. 이는 AVRO에서 특정 레코드의 개념과 유사합니다. mbknor-jackson-jsonschema는 전달된 POJO에 대한 JSON 스키마를 생성할 수 있습니다. 이 라이브러리는 JSON 스키마에 추가 정보를 삽입할 수도 있습니다.

AWS Glue Schema Registry 라이브러리는 스키마에 삽입된 "className" 필드를 사용하여 완전히 분류된 클래스 이름을 제공합니다. "className" 필드는 deserializer에서 해당 클래스의 객체로 역직렬화하는 데 사용됩니다.

```
 Example class :

@JsonSchemaDescription("This is a car")
@JsonSchemaTitle("Simple Car Schema")
@Builder
@AllArgsConstructor
@EqualsAndHashCode
// Fully qualified class name to be added to an additionally injected property
// called className for deserializer to determine which class to deserialize
// the bytes into
@JsonSchemaInject(
        strings = {@JsonSchemaString(path = "className",
                value =
 "com.amazonaws.services.schemaregistry.integrationtests.generators.Car")}
)
// List of annotations to help infer JSON Schema are defined by https://github.com/
mbknor/mbknor-jackson-jsonSchema
public class Car {
    @JsonProperty(required = true)
    private String make;

    @JsonProperty(required = true)
    private String model;

    @JsonSchemaDefault("true")
    @JsonProperty
    public boolean used;

    @JsonSchemaInject(ints = {@JsonSchemaInt(path = "multipleOf", value = 1000)})
    @Max(200000)
    @JsonProperty
    private int miles;
```

```
    @Min(2000)
    @JsonProperty
    private int year;

    @JsonProperty
    private Date purchaseDate;

    @JsonProperty
    @JsonFormat(shape = JsonFormat.Shape.NUMBER)
    private Date listedDate;

    @JsonProperty
    private String[] owners;

    @JsonProperty
    private Collection<Float> serviceChecks;

    // Empty constructor is required by Jackson to deserialize bytes
    // into an Object of this class
    public Car() {}
}
```

스키마 생성

AWS Glue API 또는 AWS Glue 콘솔을 사용하여 스키마를 생성할 수 있습니다.

AWS Glue API

다음 단계를 사용하여 AWS Glue API로 이 태스크를 수행할 수 있습니다.

새 스키마를 추가하려면 CreateSchema 작업(Python: create_schema) API를 사용합니다.

스키마에 대한 레지스트리를 나타내려면 `RegistryId` 구조를 지정합니다. 또는 기본 레지스트리를 사용하려면 `RegistryId`를 생략합니다.

문자, 숫자, 하이픈, 밑줄로 구성된 `SchemaName`을 지정하고 `DataFormat`은 **`AVRO`** 또는 **`JSON`**으로 지정합니다. `DataFormat`은 스키마에 한 번 설정되면 변경할 수 없습니다.

`Compatibility` 모드를 지정합니다.

- [뒤로(권장)(Backward (recommended))] - 소비자가 현재 버전과 이전 버전을 모두 읽을 수 있습니다.
- [모두 뒤로(Backward all)] - 소비자가 현재 및 모든 이전 버전을 읽을 수 있습니다.

- [앞으로(Forward)] - 소비자가 현재 및 후속 버전을 모두 읽을 수 있습니다.
- [모두 앞으로(Forward all)] - 소비자가 현재 버전과 모든 후속 버전을 읽을 수 있습니다.
- [전체(Full)] - 뒤로 및 앞으로의 조합입니다.
- [모두 전체(Full all)] - 모두 뒤로 및 모두 앞으로의 조합입니다.
- [없음(None)] - 호환성 확인이 수행되지 않습니다.
- [사용 중지됨(Disabled)] - 이 스키마에 대한 버전 관리를 방지합니다.

선택적으로 스키마에 Tags를 지정합니다.

Avro, JSON 또는 Protobuf 데이터 포맷으로 스키마를 정의하려면 SchemaDefinition을(를) 지정합니다. 예제를 참조하세요.

Avro 데이터 포맷의 경우:

```
aws glue create-schema --registry-id RegistryName="registryName1" --schema-name
 testschema --compatibility NONE --data-format AVRO --schema-definition "{\"type\":
 \"record\", \"name\": \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"},
 {\"name\": \"f2\", \"type\": \"string\"} ]}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName1" --schema-name testschema --compatibility
 NONE --data-format AVRO  --schema-definition "{\"type\": \"record\", \"name\": \"r1\",
 \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type\":
 \"string\"} ]}"
```

JSON 데이터 포맷의 경우:

```
aws glue create-schema --registry-id RegistryName="registryName" --schema-name
 testSchemaJson --compatibility NONE --data-format JSON --schema-definition "{\"$schema
\": \"http://json-schema.org/draft-07/schema#\",\"type\":\"object\",\"properties\":
{\"f1\":{\"type\":\"string\"}}}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName" --schema-name testSchemaJson --compatibility
 NONE --data-format JSON --schema-definition "{\"$schema\": \"http://json-schema.org/
draft-07/schema#\",\"type\":\"object\",\"properties\":{\"f1\":{\"type\":\"string\"}}}"
```

Protobuf 데이터 형식의 경우:

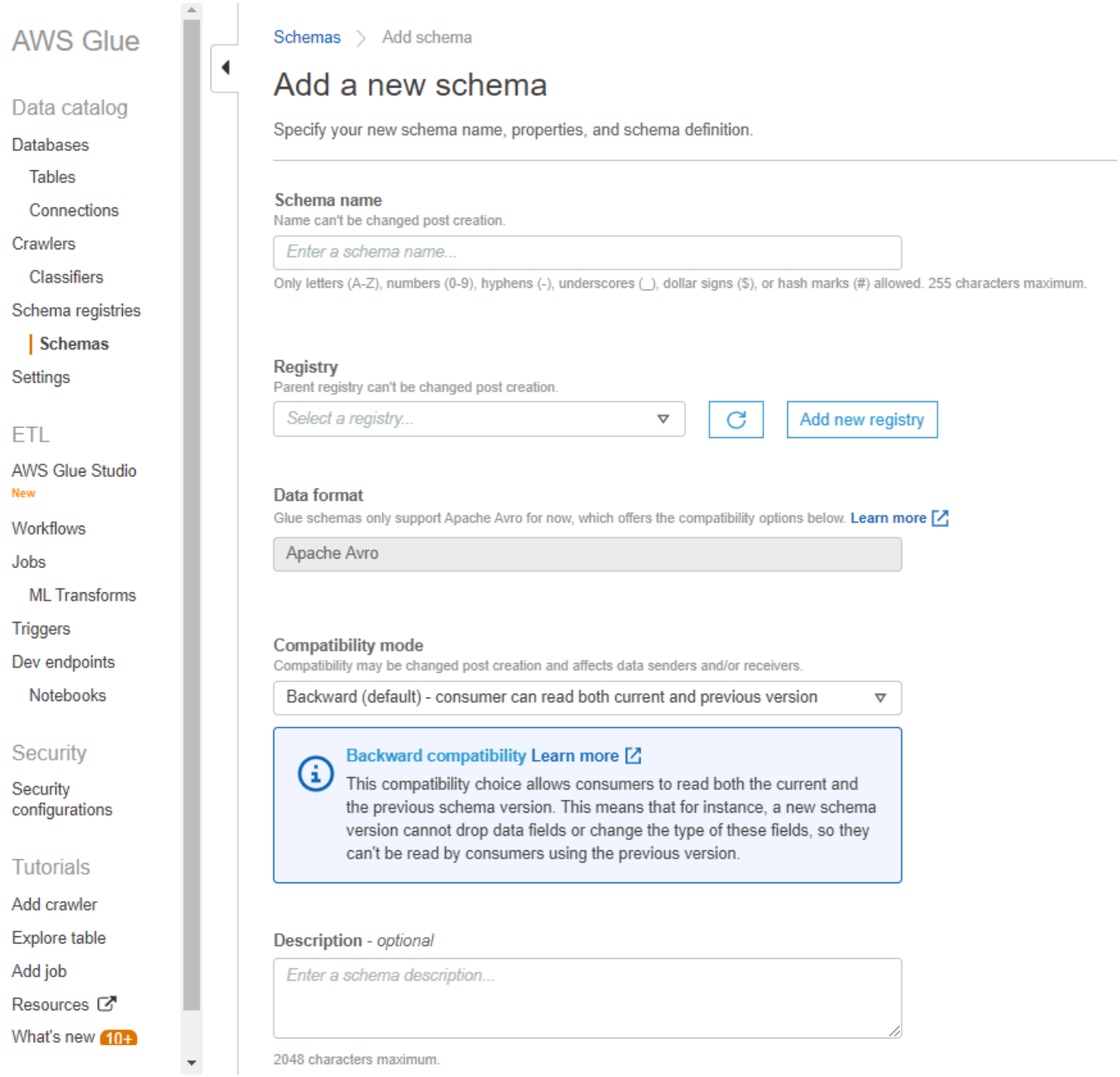
```
aws glue create-schema --registry-id RegistryName="registryName" --schema-name
 testSchemaProtobuf --compatibility NONE --data-format PROTOBUF --schema-definition
 "syntax = \"proto2\";package org.test;message Basic { optional int32 basic = 1;}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName" --schema-name testSchemaProtobuf
 --compatibility NONE --data-format PROTOBUF --schema-definition "syntax =
 \"proto2\";package org.test;message Basic { optional int32 basic = 1;}"
```

AWS Glue 콘솔

AWS Glue 콘솔을 사용하여 새 스키마를 추가하려면

1. AWS 관리 콘솔에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [스키마(Schemas)]를 선택합니다.
3. [스키마 추가(Add schema)]를 선택합니다.
4. 문자, 숫자, 하이픈, 밑줄, 달러 기호 또는 해시마크로 구성된 [스키마 이름(Schema name)]을 입력합니다. 이 이름은 변경할 수 없습니다.
5. 드롭다운 메뉴에서 스키마가 저장될 [레지스트리(Registry)]를 선택합니다. 상위 레지스트리는 생성 후에 변경할 수 없습니다.
6. [데이터 포맷(Data format)]을 [Apache Avro] 또는 [JSON]으로 둡니다. 이 포맷은 이 스키마의 모든 버전에 적용됩니다.
7. [호환성 모드(Compatibility mode)]를 선택합니다.
   - [뒤로(권장)(Backward (recommended))] - 수신자가 현재 버전과 이전 버전을 모두 읽을 수 있습니다.
   - [모두 뒤로(Backward All)] - 수신자가 현재 및 모든 이전 버전을 읽을 수 있습니다.
   - [앞으로(Forward)] - 발신자가 현재 버전과 이전 버전을 모두 쓸 수 있습니다.
   - [모두 앞으로(Forward All)] - 발신자가 현재 버전과 모든 이전 버전을 모두 쓸 수 있습니다.
   - [전체(Full)] - 뒤로 및 앞으로의 조합입니다.
   - [모두 전체(Full All)] - 모두 뒤로 및 모두 앞으로의 조합입니다.
   - [없음(None)] - 호환성 확인이 수행되지 않습니다.
   - [사용 중지됨(Disabled)] - 이 스키마에 대한 버전 관리를 방지합니다.
8. 최대 250자의 레지스트리에 대한 선택적 [설명(Description)]을 입력합니다.



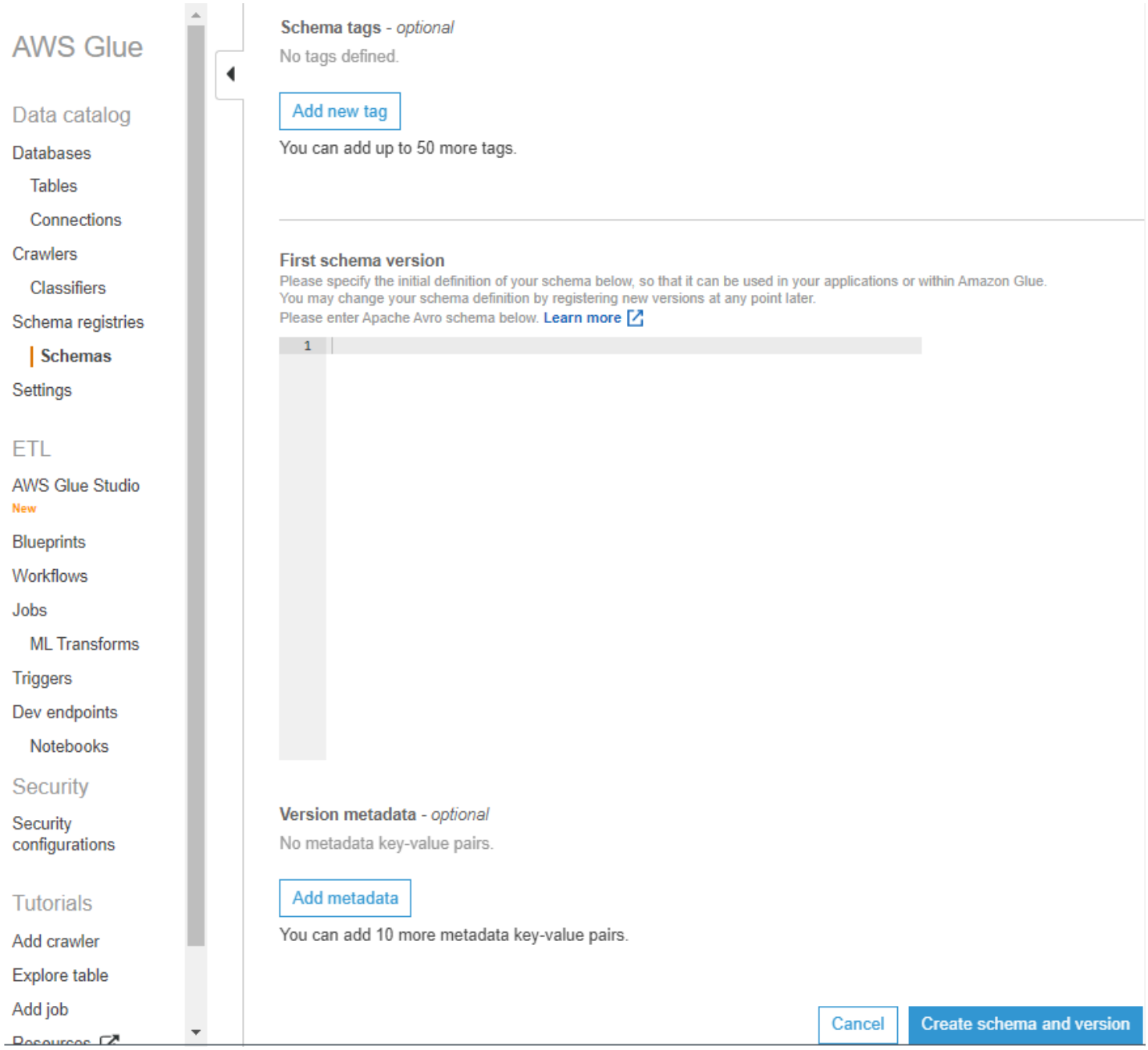
9. 선택적으로 스키마에 하나 이상의 태그를 적용합니다. [새 태그 추가(Add new tag)]를 선택하고 [태그 키(Tag key)]와 [태그 값(Tag value)](선택 사항)을 지정합니다.
10. [첫 번째 스키마 버전(First schema version)] 상자에 초기 스키마를 입력하거나 붙여넣습니다.

    Avro 포맷은 Avro 데이터 포맷 작업 섹션을 참조하세요.

    JSON 포맷은 JSON 데이터 포맷 작업 섹션을 참조하세요.

11.선택적으로 [메타데이터 추가(Add metadata)]를 선택하여 스키마 버전에 주석을 달거나 분류할 버전 메타데이터를 추가합니다.

12[스키마 및 버전 생성(Create schema and version)]을 선택합니다.



스키마가 생성되고 [스키마(Schemas)] 아래의 목록에 나타납니다.

Avro 데이터 포맷 작업

Avro는 데이터 직렬화 및 데이터 교환 서비스를 제공합니다. Avro는 데이터 정의를 JSON 포맷으로 저장하므로 읽고 해석하기 쉽습니다. 데이터 자체는 바이너리 포맷으로 저장됩니다.

Apache Avro 스키마 정의에 대한 자세한 내용은 Apache Avro 사양을 참조하세요.

JSON 데이터 포맷 작업

데이터는 JSON 포맷으로 직렬화 할 수 있습니다. JSON 스키마 포맷은 JSON 스키마 포맷의 표준을 정의합니다.

스키마 또는 레지스트리 업데이트

생성되면 스키마, 스키마 버전 또는 레지스트리를 편집할 수 있습니다.

레지스트리 업데이트

AWS Glue API 또는 AWS Glue 콘솔을 사용하여 레지스트리를 업데이트할 수 있습니다. 기존 레지스트리의 이름은 편집할 수 없습니다. 레지스트리에 대한 설명을 편집할 수 있습니다.

AWS Glue API

기존 레지스트리를 업데이트하려면UpdateRegistry 작업(Python: update_registry) API를 사용합니다.

업데이트할 레지스트리를 나타내는 `RegistryId` 구조를 지정합니다. 레지스트리에 대한 설명을 변경하려면 `Description`을 전달합니다.

```
aws glue update-registry --description updatedDescription --registry-id
 RegistryArn="arn:aws:glue:us-east-2:901234567890:registry/registryName1"
```

AWS Glue 콘솔

AWS Glue 콘솔을 사용하여 레지스트리를 업데이트하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [스키마 레지스트리(Schema registries)]를 선택합니다.
3. 해당 확인란을 선택하여 레지스트리 목록에서 레지스트리를 선택합니다.

4. [작업(Action)] 메뉴에서 [레지스트리 편집(Edit registry)]을 선택합니다.

스키마 업데이트

스키마에 대한 설명 또는 호환성 설정을 업데이트할 수 있습니다.

기존 스키마를 업데이트하려면UpdateSchema 작업(Python: update_schema) API를 사용합니다.

업데이트할 스키마를 나타내는 SchemaId 구조를 지정합니다. VersionNumber 또는 Compatibility 중 하나가 제공되어야 합니다.

코드 예제 11:

```
aws glue update-schema --description testDescription --schema-id
 SchemaName="testSchema1",RegistryName="registryName1" --schema-version-number
 LatestVersion=true --compatibility NONE
```

```
aws glue update-schema --description testDescription --schema-id
 SchemaArn="arn:aws:glue:us-east-2:901234567890:schema/registryName1/testSchema1" --
schema-version-number LatestVersion=true --compatibility NONE
```

스키마 버전 추가

스키마 버전을 추가할 때 버전을 비교하여 새 스키마가 허용되는지 확인해야 합니다.

기존 스키마에 새 버전을 추가하려면 RegisterSchemaVersion 작업(Python: register_schema_version) API를 사용합니다.

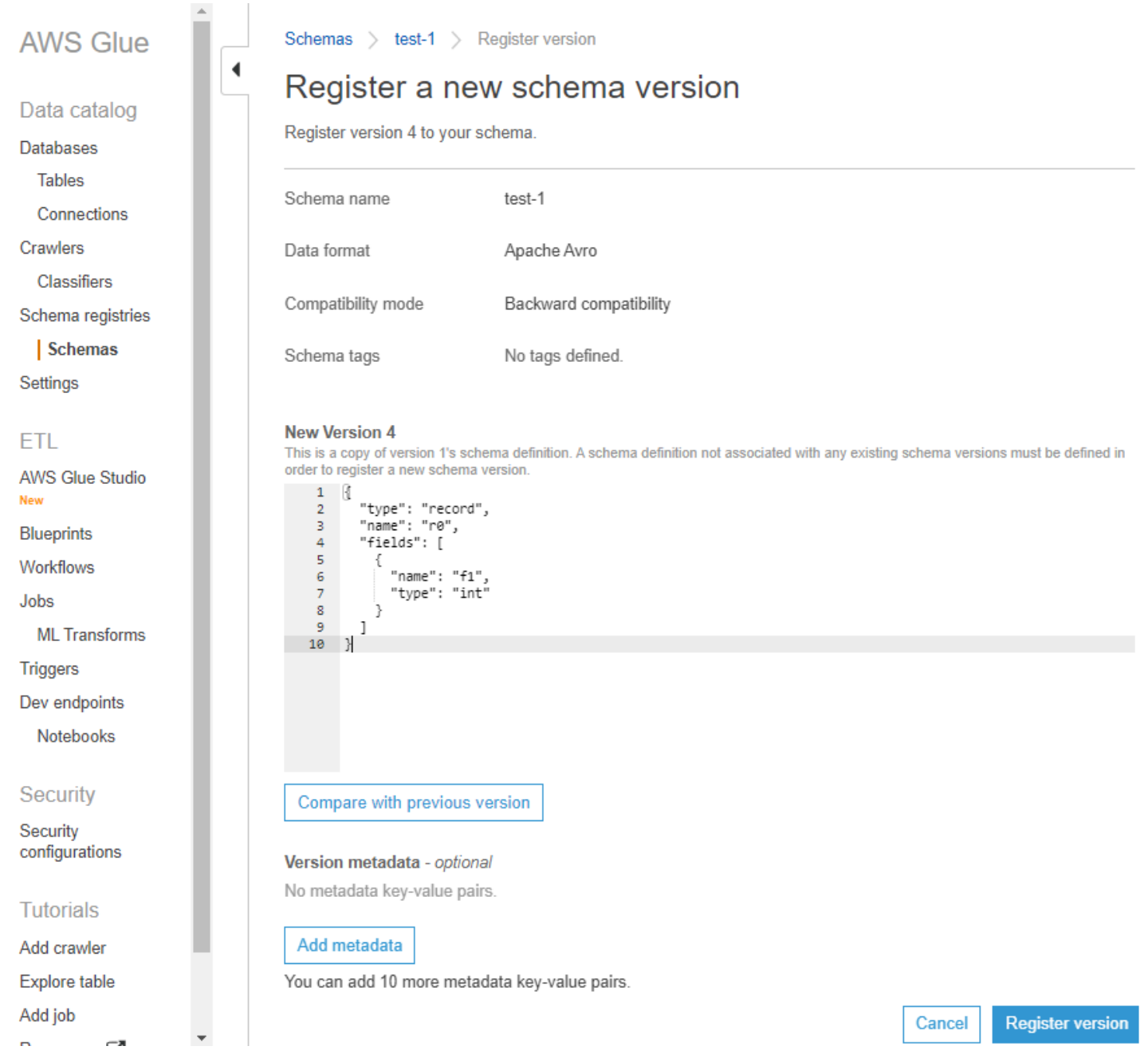
버전을 추가하려는 스키마를 나타내려면 SchemaId 구조를 지정하고 스키마를 정의하려면 SchemaDefinition을 지정합니다.

코드 예제 12:

```
aws glue register-schema-version --schema-definition "{\"type\": \"record\", \"name\":
 \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type
\": \"string\"} ]}" --schema-id SchemaArn="arn:aws:glue:us-east-1:901234567890:schema/
registryName/testschema"
```

```
aws glue register-schema-version --schema-definition "{\"type\": \"record\", \"name\":
 \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type
\": \"string\"} ]}" --schema-id SchemaName="testschema",RegistryName="testregistry"
```

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [스키마(Schemas)]를 선택합니다.
3. 해당 확인란을 선택하여 스키마 목록에서 스키마를 선택합니다.
4. 확인란을 선택하여 목록에서 하나 이상의 스키마를 선택합니다.
5. [작업(Action)] 메뉴에서 [새 버전 등록(Register new version)]을 선택합니다.
6. [새 버전(New version)] 상자에 새 스키마를 입력하거나 붙여 넣습니다.
7. [이전 버전과 비교(Compare with previous version)]를 선택하여 이전 스키마 버전과의 차이점을 확인합니다.
8. 선택적으로 [메타데이터 추가(Add metadata)]를 선택하여 스키마 버전에 주석을 달거나 분류할 버전 메타데이터를 추가합니다. [키(Key)]와 [값(Value)](선택 사항)을 입력합니다.
9. [버전 등록(Register version)]을 선택합니다.



스키마 버전이 버전 목록에 나타납니다. 버전이 호환 모드를 변경한 경우 버전이 체크포인트로 표시됩니다.

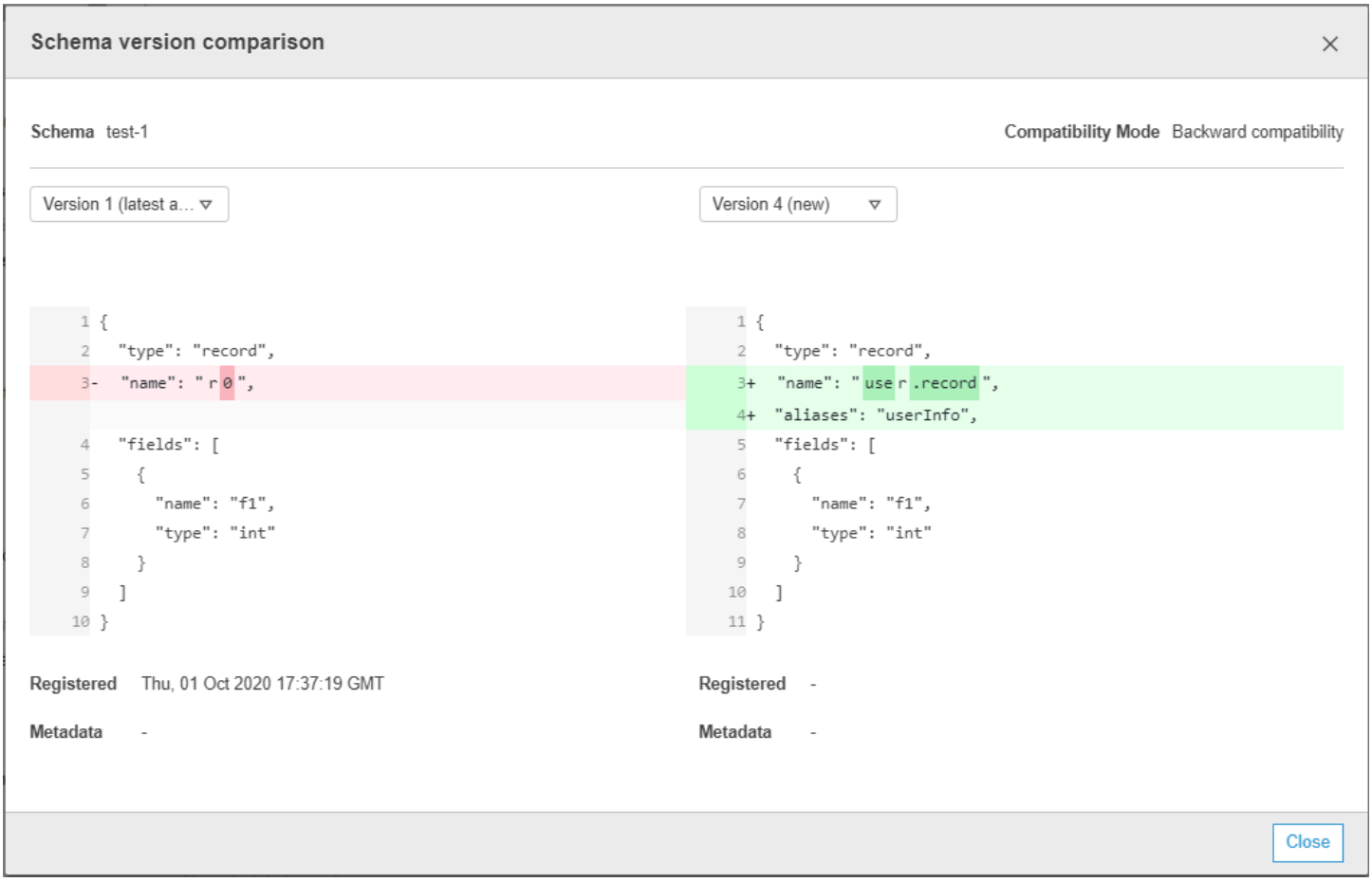
스키마 버전 비교의 예

[이전 버전과 비교(Compare with previous version)]를 선택하면 이전 버전과 새 버전이 함께 표시됩니다. 변경된 정보는 다음과 같이 강조 표시됩니다.

- 노란색: 변경된 정보를 나타냅니다.

- 녹색: 최신 버전에 추가된 내용을 나타냅니다.
- 빨간색: 최신 버전에서 제거된 콘텐츠를 나타냅니다.

이전 버전과 비교할 수도 있습니다.



스키마 또는 레지스트리 삭제

스키마, 스키마 버전 또는 레지스트리 삭제는 취소할 수 없는 영구적인 작업입니다.

스키마 삭제

AWS Management Console 또는 DeleteSchema 작업(Python: delete_schema) API를 사용하여 레지스트리 내에서 더 이상 사용되지 않는 스키마를 삭제할 수 있습니다.

하나 이상의 스키마 삭제는 실행 취소할 수 없는 영구 작업입니다. 스키마가 이제 필요 없는지 확인합니다.

레지스트리에서 스키마를 삭제하려면 DeleteSchema 작업(Python: delete_schema) API를 호출하여 스키마를 식별하는 `SchemaId` 구조를 지정합니다.

예:

```
aws glue delete-schema --schema-id SchemaArn="arn:aws:glue:us-
east-2:901234567890:schema/registryName1/schemaname"
```

```
aws glue delete-schema --schema-id SchemaName="TestSchema6-
deleteschemabyname",RegistryName="default-registry"
```

AWS Glue 콘솔

AWS Glue 콘솔에서 스키마를 삭제하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [스키마 레지스트리(Schema registries)]를 선택합니다.
3. 레지스트리 목록에서 스키마가 포함된 레지스트리를 선택합니다.
4. 확인란을 선택하여 목록에서 하나 이상의 스키마를 선택합니다.
5. [작업(Action)] 메뉴에서 [스키마 삭제(Delete schema)]를 선택합니다.
6. 필드에 **`Delete`** 텍스트를 입력하여 삭제를 확인합니다.
7. Delete(삭제)를 선택합니다.

지정한 스키마가 레지스트리에서 삭제됩니다.

스키마 버전 삭제

스키마가 레지스트리에 누적되면 AWS Management Console 또는 DeleteSchemaVersions 작업 (Python: delete_schema_versions) API를 사용하여 원치 않는 스키마 버전을 삭제할 수 있습니다. 하나 이상의 스키마 버전 삭제는 실행 취소할 수 없는 영구 작업입니다. 스키마 버전이 이제 필요 없는지 확인합니다.

스키마 버전을 삭제할 때 다음 제약 조건에 유의합니다.

- 체크포인트된 버전은 삭제할 수 없습니다.
- 연속 버전의 범위는 25개를 초과할 수 없습니다.
- 최신 스키마 버전은 보류 상태가 아니어야 합니다.

SchemaId 구조를 지정하여 스키마를 식별하고 Versions를 삭제할 버전 범위로 지정합니다. 버전 또는 버전 범위 지정에 대한 자세한 내용은 DeleteRegistry 작업(Python: delete_registry) 섹션을 참조하세요. 지정한 스키마 버전이 레지스트리에서 삭제됩니다.

이 호출 후 ListSchemaVersions 작업(Python: list_schema_versions) API를 호출하면 삭제된 버전의 상태가 나열됩니다.

예:

```
aws glue delete-schema-versions --schema-id
 SchemaName="TestSchema6",RegistryName="default-registry" --versions "1-1"
```

```
aws glue delete-schema-versions --schema-id SchemaArn="arn:aws:glue:us-
east-2:901234567890:schema/default-registry/TestSchema6-NON-Existent" --versions "1-1"
```

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [스키마 레지스트리(Schema registries)]를 선택합니다.
3. 레지스트리 목록에서 스키마가 포함된 레지스트리를 선택합니다.
4. 확인란을 선택하여 목록에서 하나 이상의 스키마를 선택합니다.
5. [작업(Action)] 메뉴에서 [스키마 삭제(Delete schema)]를 선택합니다.
6. 필드에 **`Delete`** 텍스트를 입력하여 삭제를 확인합니다.
7. Delete(삭제)를 선택합니다.

지정한 스키마 버전이 레지스트리에서 삭제됩니다.

레지스트리 삭제

포함된 스키마가 해당 레지스트리 아래에 더 이상 구성되지 않아야 하는 경우 레지스트리를 삭제할 수 있습니다. 이러한 스키마를 다른 레지스트리에 재할당해야 합니다.

하나 이상의 레지스트리 삭제는 실행 취소할 수 없는 영구 작업입니다. 레지스트리가 이제 필요 없는지 확인합니다.

AWS CLI를 사용하여 기본 레지스트리를 삭제할 수 있습니다.

AWS Glue API

스키마와 모든 해당 버전을 포함한 전체 레지스트리를 삭제하려면 DeleteRegistry 작업(Python: delete_registry) API를 호출합니다. `RegistryId` 구조를 지정하여 레지스트리를 식별합니다.

예:

```
aws glue delete-registry --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName1"
```

```
aws glue delete-registry --registry-id RegistryName="TestRegistry-deletebyname"
```

삭제 작업의 상태를 얻으려면 비동기 호출 후 `GetRegistry` API를 호출할 수 있습니다.

AWS Glue 콘솔

AWS Glue 콘솔에서 레지스트리를 삭제하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [스키마 레지스트리(Schema registries)]를 선택합니다.
3. 확인란을 선택하여 목록에서 레지스트리를 선택합니다.
4. [작업(Action)] 메뉴에서 [레지스트리 삭제(Delete registry)]를 선택합니다.
5. 필드에 **`Delete`** 텍스트를 입력하여 삭제를 확인합니다.
6. Delete(삭제)를 선택합니다.

선택한 레지스트리가 AWS Glue에서 삭제됩니다.

Serializer에 대한 IAM 예

> **Note**
>
> AWS 관리형 정책은 일반 사용 사례에서 필요한 권한을 부여합니다. 관리형 정책으로 스키마 레지스트리 관리에 대한 자세한 내용은 AWS Glue에 대한 AWS 관리형(미리 정의된) 정책 섹션을 참조하세요.

Serializer의 경우 지정된 스키마 정의에 대해 schemaVersionId를 찾는 기능을 제공하려면 아래와 유사한 최소 정책을 만들어야 합니다. 레지스트리의 스키마를 읽으려면 레지스트리에 대한 읽기 권한이 있어야 합니다. Resource 절을 사용하여 읽을 수 있는 레지스트리를 제한할 수 있습니다.

코드 예제 13:

```
{
    "Sid" : "GetSchemaByDefinition",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaByDefinition"
    ],
        "Resource" : ["arn:aws:glue:us-east-2:012345678:registry/registryname-1",
                      "arn:aws:glue:us-east-2:012345678:schema/registryname-1/
schemaname-1",
                      "arn:aws:glue:us-east-2:012345678:schema/registryname-1/
schemaname-2"
                     ]
}
```

또한 다음과 같은 추가 메서드를 포함하여 생산자가 새 스키마 및 버전을 생성하도록 허용할 수도 있습니다. 참고로 레지스트리 내부의 스키마를 추가/제거/발전시키기 위해 레지스트리를 검사할 수 있어야 합니다. Resource 절을 사용하여 검사할 수 있는 레지스트리를 제한할 수 있습니다.

코드 예제 14:

```
{
    "Sid" : "RegisterSchemaWithMetadata",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaByDefinition",
        "glue:CreateSchema",
        "glue:RegisterSchemaVersion",
        "glue:PutSchemaVersionMetadata",
    ],
    "Resource" : ["arn:aws:glue:aws-region:123456789012:registry/registryname-1",
                  "arn:aws:glue:aws-region:123456789012:schema/registryname-1/
schemaname-1",
                  "arn:aws:glue:aws-region:123456789012:schema/registryname-1/
schemaname-2"
                 ]
```

```
}
```

Deserializer에 대한 IAM 예

Deserializer(소비자 측)의 경우 deserializer가 역직렬화를 위해 Schema Registry에서 스키마를 가져올 수 있도록 아래와 유사한 정책을 생성해야 합니다. 참고로 레지스트리 내부의 스키마를 가져오기 위해 레지스트리를 검사할 수 있어야 합니다.

코드 예제 15:

```
{
    "Sid" : "GetSchemaVersion",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaVersion"
    ],
    "Resource" : ["*"]
}
```

AWS PrivateLink를 사용한 프라이빗 연결

AWS PrivateLink를 사용하여 AWS Glue에 대한 인터페이스 VPC 엔드포인트를 정의하여 데이터 생산자의 VPC를 AWS Glue에 연결할 수 있습니다. VPC 엔드포인트를 사용하는 경우 VPC와 AWS Glue 사이의 통신은 모두 AWS 네트워크에서 수행됩니다. 자세한 내용은 VPC 엔드포인트와 함께 AWS Glue 사용을 참조하세요.

Amazon CloudWatch 지표에 액세스

Amazon CloudWatch 지표는 CloudWatch 프리 티어의 일부로 제공됩니다. CloudWatch 콘솔에서 이러한 지표에 액세스할 수 있습니다. API 수준 지표에는 CreateSchema(성공 및 대기 시간), GetSchemaByDefinition, (성공 및 대기 시간), GetSchemaVersion(성공 및 대기 시간), RegisterSchemaVersion(성공 및 대기 시간), PutSchemaVersionMetadata(성공 및 대기 시간)가 포함됩니다. 리소스 수준 지표에는 Registry.ThrottledByLimit, SchemaVersion.ThrottledByLimit, SchemaVersion.Size가 포함됩니다.

스키마 레지스트리용 샘플 AWS CloudFormation 템플릿

다음은 AWS CloudFormation에서 Schema Registry 리소스를 생성하기 위한 샘플 템플릿입니다. 계정에 이 스택을 생성하려면 위 템플릿을 파일 `SampleTemplate.yaml`에 복사하고 다음 명령을 실행합니다.

```
aws cloudformation create-stack --stack-name ABCSchemaRegistryStack --template-body
 "'cat SampleTemplate.yaml'"
```

이 예에서는 AWS::Glue::Registry를 사용하여 레지스트리를 생성하고, AWS::Glue::Schema를 사용하여 스키마를 생성하고, AWS::Glue::SchemaVersion을 사용하여 스키마 버전을 생성하고, AWS::Glue::SchemaVersionMetadata를 사용하여 스키마 버전 메타데이터를 채웁니다.

```
Description: "A sample CloudFormation template for creating Schema Registry resources."
Resources:
  ABCRegistry:
    Type: "AWS::Glue::Registry"
    Properties:
      Name: "ABCSchemaRegistry"
      Description: "ABC Corp. Schema Registry"
      Tags:
        - Key: "Project"
          Value: "Foo"
  ABCSchema:
    Type: "AWS::Glue::Schema"
    Properties:
      Registry:
        Arn: !Ref ABCRegistry
      Name: "TestSchema"
      Compatibility: "NONE"
      DataFormat: "AVRO"
      SchemaDefinition: >
        {"namespace":"foo.avro","type":"record","name":"user","fields":
[{"name":"name","type":"string"},{"name":"favorite_number","type":"int"}]}
      Tags:
        - Key: "Project"
          Value: "Foo"
  SecondSchemaVersion:
    Type: "AWS::Glue::SchemaVersion"
    Properties:
      Schema:
        SchemaArn: !Ref ABCSchema
      SchemaDefinition: >
        {"namespace":"foo.avro","type":"record","name":"user","fields":
[{"name":"status","type":"string", "default":"ON"}, {"name":"name","type":"string"},
{"name":"favorite_number","type":"int"}]}
  FirstSchemaVersionMetadata:
    Type: "AWS::Glue::SchemaVersionMetadata"
```

```
    Properties:
      SchemaVersionId: !GetAtt ABCSchema.InitialSchemaVersionId
      Key: "Application"
      Value: "Kinesis"
  SecondSchemaVersionMetadata:
    Type: "AWS::Glue::SchemaVersionMetadata"
    Properties:
      SchemaVersionId: !Ref SecondSchemaVersion
      Key: "Application"
      Value: "Kinesis"
```

AWS Glue Schema Registry와 통합

이 섹션에서는 AWS Glue 스키마 레지스트리와의 통합에 대해 설명합니다. 이 섹션의 예에서는 AVRO 데이터 포맷의 스키마를 보여줍니다. JSON 데이터 포맷의 스키마를 포함한 더 많은 예제는 AWS Glue Schema Registry 오픈 소스 리포지토리의 통합 테스트 및 ReadMe 정보를 참조하세요.

주제

- 사용 사례: Amazon MSK 또는 Apache Kafka에 Schema Registry 연결
- 사용 사례: AWS Glue Schema Registry와 Amazon Kinesis Data Streams 통합
- 사용 사례: Amazon Managed Service for Apache Flink
- 사용 사례: AWS Lambda와 통합
- 사용 사례: AWS Glue Data Catalog
- 사용 사례: AWS Glue 스트리밍
- 사용 사례: Apache Kafka Streams

사용 사례: Amazon MSK 또는 Apache Kafka에 Schema Registry 연결

Apache Kafka 주제에 데이터를 쓰고 있다고 가정하고 다음 단계에 따라 시작할 수 있습니다.

1. 하나 이상의 주제로 Amazon Managed Streaming for Apache Kafka(Amazon MSK) 또는 Apache Kafka 클러스터를 생성합니다. Amazon MSK 클러스터를 생성하는 경우 AWS Management Console을 사용할 수 있습니다. Amazon Managed Streaming for Apache Kafka Developer Guide의 Getting Started Using Amazon MSK에서 설명하는 지침을 따릅니다.
2. 위의 SerDe 라이브러리 설치 단계를 따릅니다.

3. 스키마 레지스트리, 스키마 또는 스키마 버전을 생성하려면 이 문서의 스키마 레지스트리 시작하기 섹션에 있는 지침을 따르세요.
4. 생산자와 소비자가 Schema Registry를 사용하여 Amazon MSK 또는 Apache Kafka 주제에서 레코드를 쓰고 읽도록 시작합니다. 예제 생산자 및 소비자 코드는 Serde 라이브러리의 ReadMe 파일에서 찾을 수 있습니다. 생산자의 Schema Registry 라이브러리는 자동으로 레코드를 직렬화하고 스키마 버전 ID로 레코드를 장식합니다.
5. 이 레코드의 스키마가 입력되었거나 자동 등록이 설정되어 있으면 Schema Registry에 스키마가 등록됩니다.
6. AWS Glue Schema Registry 라이브러리를 사용하여 Amazon MSK 또는 Apache Kafka 주제에서 읽는 소비자는 Schema Registry에서 스키마를 자동으로 조회합니다.

사용 사례: AWS Glue Schema Registry와 Amazon Kinesis Data Streams 통합

이 통합을 위해서는 기존 Amazon Kinesis 데이터 스트림이 있어야 합니다. 자세한 내용은 Amazon Kinesis Data Streams Developer Guide의 Getting Started with Amazon Kinesis Data Streams를 참조하세요.

Kinesis 데이터 스트림의 데이터와 상호 작용할 수 있는 두 가지 방법이 있습니다.

- Java의 Kinesis Producer Library(KPL) 및 Kinesis Client Library(KCL) 라이브러리를 통해 다국어 지원은 제공되지 않습니다.
- AWS SDK for Java에서 사용 가능한 `PutRecords`, `PutRecord` 및 `GetRecords` Kinesis Data Streams API를 통해.

현재 KPL/KCL 라이브러리를 사용하는 경우 해당 방법을 계속 사용하는 것이 좋습니다. 예제와 같이 Schema Registry가 통합된 업데이트된 KCL 및 KPL 버전이 있습니다. 그렇지 않으면 AWS Glue KDS API를 직접 사용하는 경우 샘플 코드를 사용하여 Schema Registry를 활용할 수 있습니다.

Schema Registry 통합은 KPL v0.14.2 이상 및 KCL v2.3 이상에서만 사용할 수 있습니다. JSON 데이터 포맷과 Schema Registry 통합은 KPL v0.14.8 이상 및 KCL v2.3.6 이상에서 사용할 수 있습니다.

Kinesis SDK V2를 사용하여 데이터와 상호 작용

이 섹션에서는 Kinesis SDK V2를 사용한 Kinesis와의 상호 작용에 대해 설명합니다.

```
// Example JSON Record, you can construct a AVRO record also
private static final JsonDataWithSchema record =
 JsonDataWithSchema.builder(schemaString, payloadString);
```

```
private static final DataFormat dataFormat = DataFormat.JSON;

//Configurations for Schema Registry
GlueSchemaRegistryConfiguration gsrConfig = new GlueSchemaRegistryConfiguration("us-
east-1");

GlueSchemaRegistrySerializer glueSchemaRegistrySerializer =
        new GlueSchemaRegistrySerializerImpl(awsCredentialsProvider, gsrConfig);
GlueSchemaRegistryDataFormatSerializer dataFormatSerializer =
        new GlueSchemaRegistrySerializerFactory().getInstance(dataFormat, gsrConfig);

Schema gsrSchema =
        new Schema(dataFormatSerializer.getSchemaDefinition(record), dataFormat.name(),
 "MySchema");

byte[] serializedBytes = dataFormatSerializer.serialize(record);

byte[] gsrEncodedBytes = glueSchemaRegistrySerializer.encode(streamName, gsrSchema,
 serializedBytes);

PutRecordRequest putRecordRequest = PutRecordRequest.builder()
        .streamName(streamName)
        .partitionKey("partitionKey")
        .data(SdkBytes.fromByteArray(gsrEncodedBytes))
        .build();
shardId = kinesisClient.putRecord(putRecordRequest)
        .get()
        .shardId();

GlueSchemaRegistryDeserializer glueSchemaRegistryDeserializer = new
 GlueSchemaRegistryDeserializerImpl(awsCredentialsProvider, gsrConfig);

GlueSchemaRegistryDataFormatDeserializer gsrDataFormatDeserializer =
        glueSchemaRegistryDeserializerFactory.getInstance(dataFormat, gsrConfig);

GetShardIteratorRequest getShardIteratorRequest = GetShardIteratorRequest.builder()
        .streamName(streamName)
        .shardId(shardId)
        .shardIteratorType(ShardIteratorType.TRIM_HORIZON)
        .build();

String shardIterator = kinesisClient.getShardIterator(getShardIteratorRequest)
        .get()
        .shardIterator();
```

```
GetRecordsRequest getRecordRequest = GetRecordsRequest.builder()
        .shardIterator(shardIterator)
        .build();
GetRecordsResponse recordsResponse = kinesisClient.getRecords(getRecordRequest)
        .get();

List<Object> consumerRecords = new ArrayList<>();
List<Record> recordsFromKinesis = recordsResponse.records();

for (int i = 0; i < recordsFromKinesis.size(); i++) {
    byte[] consumedBytes = recordsFromKinesis.get(i)
            .data()
            .asByteArray();

    Schema gsrSchema = glueSchemaRegistryDeserializer.getSchema(consumedBytes);
    Object decodedRecord =
 gsrDataFormatDeserializer.deserialize(ByteBuffer.wrap(consumedBytes),

 gsrSchema.getSchemaDefinition());
    consumerRecords.add(decodedRecord);
}
```

KPL/KCL 라이브러리를 사용하여 데이터와 상호 작용

이 섹션에서는 KPL/KCL 라이브러리를 사용하여 Kinesis Data Streams를 Schema Registry와 통합하는 방법을 설명합니다. KPL/KCL 사용에 대한 자세한 내용은 Amazon Kinesis Data Streams Developer Guide의 Developing Producers Using the Amazon Kinesis Producer Library를 참조하세요.

KPL에서 Schema Registry 설정

1. AWS Glue Schema Registry에서 작성된 데이터, 데이터 포맷 및 스키마 이름에 대한 스키마 정의를 정의합니다.
2. 필요에 따라 `GlueSchemaRegistryConfiguration` 객체를 구성합니다.
3. `addUserRecord` API로 스키마 객체를 전달합니다.

```
private static final String SCHEMA_DEFINITION = "{"namespace": "example.avro",\n"
+ " "type": "record",\n"
+ " "name": "User",\n"
+ " "fields": [\n"
+ " {"name": "name", "type": "string"},\n"
+ " {"name": "favorite_number", "type": ["int", "null"]},\n"
```

```
+ " {"name": "favorite_color", "type": ["string", "null"]}\n"
+ " ]\n"
+ "}";

KinesisProducerConfiguration config = new KinesisProducerConfiguration();
config.setRegion("us-west-1")

//[Optional] configuration for Schema Registry.

GlueSchemaRegistryConfiguration schemaRegistryConfig =
new GlueSchemaRegistryConfiguration("us-west-1");

schemaRegistryConfig.setCompression(true);

config.setGlueSchemaRegistryConfiguration(schemaRegistryConfig);

///Optional configuration ends.

final KinesisProducer producer =
      new KinesisProducer(config);

final ByteBuffer data = getDataToSend();

com.amazonaws.services.schemaregistry.common.Schema gsrSchema =
    new Schema(SCHEMA_DEFINITION, DataFormat.AVRO.toString(), "demoSchema");

ListenableFuture<UserRecordResult> f = producer.addUserRecord(
config.getStreamName(), TIMESTAMP, Utils.randomExplicitHashKey(), data, gsrSchema);

private static ByteBuffer getDataToSend() {
      org.apache.avro.Schema avroSchema =
        new org.apache.avro.Schema.Parser().parse(SCHEMA_DEFINITION);

      GenericRecord user = new GenericData.Record(avroSchema);
      user.put("name", "Emily");
      user.put("favorite_number", 32);
      user.put("favorite_color", "green");

      ByteArrayOutputStream outBytes = new ByteArrayOutputStream();
      Encoder encoder = EncoderFactory.get().directBinaryEncoder(outBytes, null);
      new GenericDatumWriter<>(avroSchema).write(user, encoder);
      encoder.flush();
      return ByteBuffer.wrap(outBytes.toByteArray());
```

```
}
```

Kinesis 클라이언트 라이브러리 설정

Java로 Kinesis Client Library 소비자를 개발합니다. 자세한 내용은 Amazon Kinesis Data Streams Developer Guide의 Developing a Kinesis Client Library Consumer in Java를 참조하세요.

1. `GlueSchemaRegistryConfiguration` 객체를 전달하여 `GlueSchemaRegistryDeserializer`의 인스턴스를 생성합니다.
2. `retrievalConfig.glueSchemaRegistryDeserializer`에 `GlueSchemaRegistryDeserializer`를 전달합니다.
3. `kinesisClientRecord.getSchema()`를 호출하여 수신 메시지의 스키마에 액세스합니다.

```
GlueSchemaRegistryConfiguration schemaRegistryConfig =
    new GlueSchemaRegistryConfiguration(this.region.toString());

 GlueSchemaRegistryDeserializer glueSchemaRegistryDeserializer =
    new
 GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(),
 schemaRegistryConfig);

 RetrievalConfig retrievalConfig =
 configsBuilder.retrievalConfig().retrievalSpecificConfig(new
 PollingConfig(streamName, kinesisClient));
 retrievalConfig.glueSchemaRegistryDeserializer(glueSchemaRegistryDeserializer);

  Scheduler scheduler = new Scheduler(
            configsBuilder.checkpointConfig(),
            configsBuilder.coordinatorConfig(),
            configsBuilder.leaseManagementConfig(),
            configsBuilder.lifecycleConfig(),
            configsBuilder.metricsConfig(),
            configsBuilder.processorConfig(),
            retrievalConfig
        );

 public void processRecords(ProcessRecordsInput processRecordsInput) {
            MDC.put(SHARD_ID_MDC_KEY, shardId);
            try {
                log.info("Processing {} record(s)",
                processRecordsInput.records().size());
```

```
                processRecordsInput.records()
                .forEach(
                    r ->
                        log.info("Processed record pk: {} -- Seq: {} : data {} with
schema: {}",
                        r.partitionKey(),
r.sequenceNumber(), recordToAvroObj(r).toString(), r.getSchema()));
            } catch (Throwable t) {
                log.error("Caught throwable while processing records. Aborting.");
                Runtime.getRuntime().halt(1);
            } finally {
                MDC.remove(SHARD_ID_MDC_KEY);
            }
}

private GenericRecord recordToAvroObj(KinesisClientRecord r) {
   byte[] data = new byte[r.data().remaining()];
   r.data().get(data, 0, data.length);
   org.apache.avro.Schema schema = new
org.apache.avro.Schema.Parser().parse(r.schema().getSchemaDefinition());
   DatumReader datumReader = new GenericDatumReader<>(schema);

   BinaryDecoder binaryDecoder = DecoderFactory.get().binaryDecoder(data, 0,
data.length, null);
   return (GenericRecord) datumReader.read(null, binaryDecoder);
}
```

Kinesis Data Streams API를 사용하여 데이터와 상호 작용

이 섹션에서는 Kinesis Data Streams API를 사용하여 Kinesis Data Streams를 Schema Registry와 통합하는 방법을 설명합니다.

1. 다음 Maven 종속성을 업데이트합니다.

```
<dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>com.amazonaws</groupId>
                <artifactId>aws-java-sdk-bom</artifactId>
                <version>1.11.884</version>
                <type>pom</type>
                <scope>import</scope>
```

```
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-kinesis</artifactId>
        </dependency>

        <dependency>
            <groupId>software.amazon.glue</groupId>
            <artifactId>schema-registry-serde</artifactId>
            <version>1.1.5</version>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-cbor</artifactId>
            <version>2.11.3</version>
        </dependency>
    </dependencies>
```

2. 생산자에서 Kinesis Data Streams의 PutRecords 또는 PutRecord API를 사용하여 스키마 헤더 정보를 추가합니다.

```
//The following lines add a Schema Header to the record
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            new com.amazonaws.services.schemaregistry.common.Schema(schemaDefinition,
 DataFormat.AVRO.name(),
                schemaName);
        GlueSchemaRegistrySerializerImpl glueSchemaRegistrySerializer =
            new
 GlueSchemaRegistrySerializerImpl(DefaultCredentialsProvider.builder().build(), new
 GlueSchemaRegistryConfiguration(getConfigs()));
        byte[] recordWithSchemaHeader =
            glueSchemaRegistrySerializer.encode(streamName, awsSchema,
 recordAsBytes);
```

3. 생산자에서 PutRecords 또는 PutRecord API를 사용하여 레코드를 데이터 스트림에 넣습니다.
4. 소비자의 헤더에서 스키마 레코드를 제거하고 Avro 스키마 레코드를 직렬화합니다.

```
//The following lines remove Schema Header from record
```

```
        GlueSchemaRegistryDeserializerImpl glueSchemaRegistryDeserializer =
            new
    GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(),
    getConfigs());
        byte[] recordWithSchemaHeaderBytes = new
    byte[recordWithSchemaHeader.remaining()];
        recordWithSchemaHeader.get(recordWithSchemaHeaderBytes, 0,
    recordWithSchemaHeaderBytes.length);
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            glueSchemaRegistryDeserializer.getSchema(recordWithSchemaHeaderBytes);
        byte[] record =
    glueSchemaRegistryDeserializer.getData(recordWithSchemaHeaderBytes);

        //The following lines serialize an AVRO schema record
        if (DataFormat.AVRO.name().equals(awsSchema.getDataFormat())) {
            Schema avroSchema = new
    org.apache.avro.Schema.Parser().parse(awsSchema.getSchemaDefinition());
            Object genericRecord = convertBytesToRecord(avroSchema, record);
            System.out.println(genericRecord);
        }
```

Kinesis Data Streams API를 사용하여 데이터와 상호 작용

다음은 PutRecords 및 GetRecords API 사용을 위한 예제 코드입니다.

```
//Full sample code
import
 com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializerImpl;
import
 com.amazonaws.services.schemaregistry.serializers.GlueSchemaRegistrySerializerImpl;
import com.amazonaws.services.schemaregistry.utils.AVROUtils;
import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;
import org.apache.avro.Schema;
import org.apache.avro.generic.GenericData;
import org.apache.avro.generic.GenericDatumReader;
import org.apache.avro.generic.GenericDatumWriter;
import org.apache.avro.generic.GenericRecord;
import org.apache.avro.io.Decoder;
import org.apache.avro.io.DecoderFactory;
import org.apache.avro.io.Encoder;
import org.apache.avro.io.EncoderFactory;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.services.glue.model.DataFormat;
```

```
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;


public class PutAndGetExampleWithEncodedData {
    static final String regionName = "us-east-2";
    static final String streamName = "testStream1";
    static final String schemaName = "User-Topic";
    static final String AVRO_USER_SCHEMA_FILE = "src/main/resources/user.avsc";
    KinesisApi kinesisApi = new KinesisApi();

    void runSampleForPutRecord() throws IOException {
        Object testRecord = getTestRecord();
        byte[] recordAsBytes = convertRecordToBytes(testRecord);
        String schemaDefinition =
 AVROUtils.getInstance().getSchemaDefinition(testRecord);

        //The following lines add a Schema Header to a record
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            new com.amazonaws.services.schemaregistry.common.Schema(schemaDefinition,
 DataFormat.AVRO.name(),
                schemaName);
        GlueSchemaRegistrySerializerImpl glueSchemaRegistrySerializer =
            new
 GlueSchemaRegistrySerializerImpl(DefaultCredentialsProvider.builder().build(), new
 GlueSchemaRegistryConfiguration(regionName));
        byte[] recordWithSchemaHeader =
            glueSchemaRegistrySerializer.encode(streamName, awsSchema, recordAsBytes);

        //Use PutRecords api to pass a list of records
        kinesisApi.putRecords(Collections.singletonList(recordWithSchemaHeader),
 streamName, regionName);

        //OR
        //Use PutRecord api to pass single record
        //kinesisApi.putRecord(recordWithSchemaHeader, streamName, regionName);
    }
```

```
   byte[] runSampleForGetRecord() throws IOException {
       ByteBuffer recordWithSchemaHeader = kinesisApi.getRecords(streamName,
regionName);

       //The following lines remove the schema registry header
       GlueSchemaRegistryDeserializerImpl glueSchemaRegistryDeserializer =
           new
GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(), new
GlueSchemaRegistryConfiguration(regionName));
       byte[] recordWithSchemaHeaderBytes = new
byte[recordWithSchemaHeader.remaining()];
       recordWithSchemaHeader.get(recordWithSchemaHeaderBytes, 0,
recordWithSchemaHeaderBytes.length);

       com.amazonaws.services.schemaregistry.common.Schema awsSchema =
           glueSchemaRegistryDeserializer.getSchema(recordWithSchemaHeaderBytes);

       byte[] record =
glueSchemaRegistryDeserializer.getData(recordWithSchemaHeaderBytes);

       //The following lines serialize an AVRO schema record
       if (DataFormat.AVRO.name().equals(awsSchema.getDataFormat())) {
           Schema avroSchema = new
org.apache.avro.Schema.Parser().parse(awsSchema.getSchemaDefinition());
           Object genericRecord = convertBytesToRecord(avroSchema, record);
           System.out.println(genericRecord);
       }

       return record;
   }

   private byte[] convertRecordToBytes(final Object record) throws IOException {
       ByteArrayOutputStream recordAsBytes = new ByteArrayOutputStream();
       Encoder encoder = EncoderFactory.get().directBinaryEncoder(recordAsBytes,
null);
       GenericDatumWriter datumWriter = new
GenericDatumWriter<>(AVROUtils.getInstance().getSchema(record));
       datumWriter.write(record, encoder);
       encoder.flush();
       return recordAsBytes.toByteArray();
   }

   private GenericRecord convertBytesToRecord(Schema avroSchema, byte[] record) throws
IOException {
```

```
        final GenericDatumReader<GenericRecord> datumReader = new
 GenericDatumReader<>(avroSchema);
        Decoder decoder = DecoderFactory.get().binaryDecoder(record, null);
        GenericRecord genericRecord = datumReader.read(null, decoder);
        return genericRecord;
    }

    private Map<String, String> getMetadata() {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("event-source-1", "topic1");
        metadata.put("event-source-2", "topic2");
        metadata.put("event-source-3", "topic3");
        metadata.put("event-source-4", "topic4");
        metadata.put("event-source-5", "topic5");
        return metadata;
    }

    private GlueSchemaRegistryConfiguration getConfigs() {
        GlueSchemaRegistryConfiguration configs = new
 GlueSchemaRegistryConfiguration(regionName);
        configs.setSchemaName(schemaName);
        configs.setAutoRegistration(true);
        configs.setMetadata(getMetadata());
        return configs;
    }

    private Object getTestRecord() throws IOException {
        GenericRecord genericRecord;
        Schema.Parser parser = new Schema.Parser();
        Schema avroSchema = parser.parse(new File(AVRO_USER_SCHEMA_FILE));

        genericRecord = new GenericData.Record(avroSchema);
        genericRecord.put("name", "testName");
        genericRecord.put("favorite_number", 99);
        genericRecord.put("favorite_color", "red");

        return genericRecord;
    }
}
```

사용 사례: Amazon Managed Service for Apache Flink

Apache Flink는 무제한 및 제한 데이터 스트림에 대한 상태 저장 계산에 널리 사용되는 오픈 소스 프레임워크 및 분산 처리 엔진입니다. Amazon Managed Service for Apache Flink는 스트리밍 데이터를 처리하는 Apache Flink 애플리케이션을 구축 및 관리할 수 있는 완전관리형 AWS 서비스입니다.

오픈 소스 Apache Flink는 다양한 소스와 싱크를 제공합니다. 예를 들어 사전 정의된 데이터 원본에는 파일, 디렉터리 및 소켓에서 읽기와 컬렉션 및 반복기에서 데이터 수집이 포함됩니다. Apache Flink DataStream 커넥터는 Apache Flink가 소스 및/또는 싱크로 Apache Kafka 또는 Kinesis와 같은 다양한 서드 파티 시스템과 인터페이스할 수 있는 코드를 제공합니다.

자세한 내용은 Amazon Kinesis Data Analytics Developer Guide를 참조하세요.

Apache Flink Kafka 커넥터

Apache Flink는 정확히 1회 보장으로 Kafka 주제에서 데이터를 읽고 쓰기 위한 Apache Kafka 데이터 스트림 커넥터를 제공합니다. Flink의 Kafka 소비자인 `FlinkKafkaConsumer`는 하나 이상의 Kafka 주제에서 읽을 수 있는 액세스를 제공합니다. Apache Flink의 Kafka Producer인 `FlinkKafkaProducer`를 사용하면 하나 이상의 Kafka 주제에 대한 레코드 스트림을 작성할 수 있습니다. 자세한 내용은 Apache Kafka 커넥터를 참조하세요.

Apache Flink Kinesis 스트림 커넥터

Kinesis 데이터 스트림 커넥터는 Amazon Kinesis Data Streams에 대한 액세스를 제공합니다. `FlinkKinesisConsumer`는 동일한 AWS 서비스 리전 내에서 여러 Kinesis 스트림을 구독하는 정확히 1회 병렬 스트리밍 데이터 원본이며 작업이 실행되는 동안 스트림의 다시 샤딩을 투명하게 처리할 수 있습니다. 소비자의 각 하위 태스크는 여러 Kinesis 샤드에서 데이터 레코드를 가져오는 일을 담당합니다. 각 하위 태스크에서 가져온 샤드 수는 샤드가 다히고 Kinesis에서 생성됨에 따라 변경됩니다. `FlinkKinesisProducer`는 Kinesis Producer Library(KPL)를 사용하여 Apache Flink 스트림의 데이터를 Kinesis 스트림에 넣습니다. 자세한 내용은 Amazon Kinesis Streams 커넥터를 참조하세요.

자세한 내용은 AWS Glue 스키마 Github 리포지토리를 참조하세요.

Apache Flink와의 통합

Schema Registry와 함께 제공되는 SerDes 라이브러리는 Apache Flink와 통합됩니다. Apache Flink를 사용하려면 Apache Flink 커넥터에 연결할 수 있는 `GlueSchemaRegistryAvroSerializationSchema` 및 `GlueSchemaRegistryAvroDeserializationSchema`라는 `SerializationSchema` 및 `DeserializationSchema` 인터페이스를 구현해야 합니다.

Apache Flink 애플리케이션에 AWS Glue Schema Registry 종속성 추가

Apache Flink 애플리케이션에서 AWS Glue Schema Registry에 대한 통합 종속성을 설정하려면

1. 종속 프로그램을 pom.xml 파일에 추가합니다.

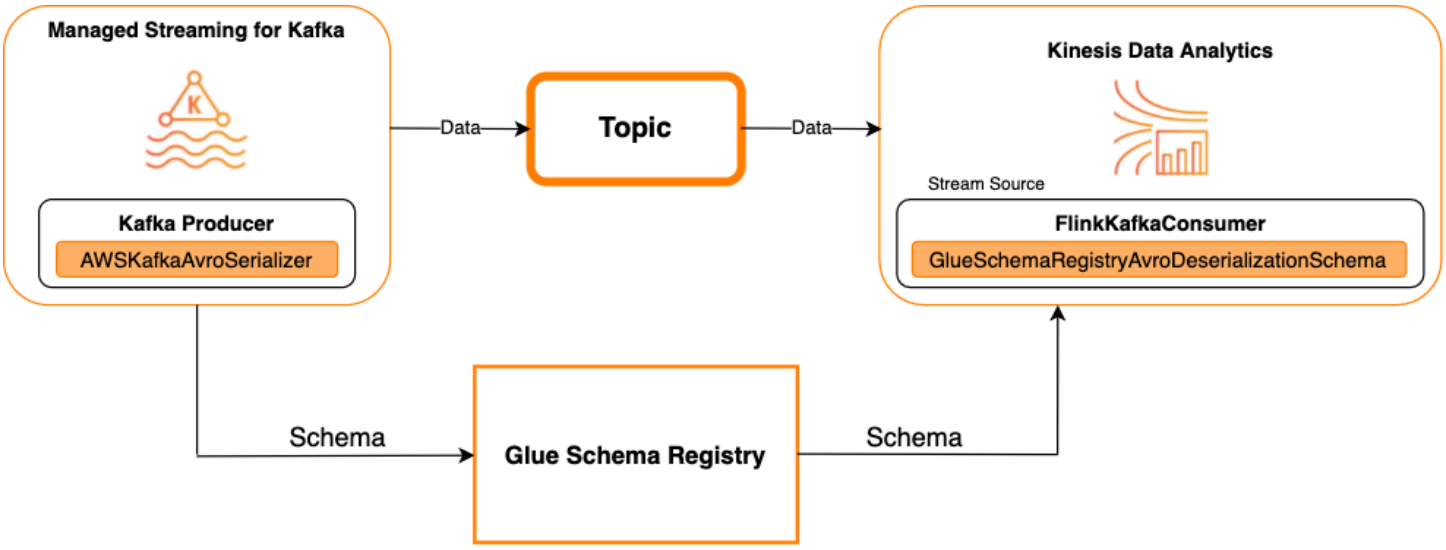
```
<dependency>
    <groupId>software.amazon.glue</groupId>
    <artifactId>schema-registry-flink-serde</artifactId>
    <version>1.0.0</version>
</dependency>
```

pache Flink와 Kafka 또는 Amazon MSK 통합

Kafka를 소스나 싱크로 사용하여 Apache Flink용 Managed Service for Apache Flink를 사용할 수 있습니다.

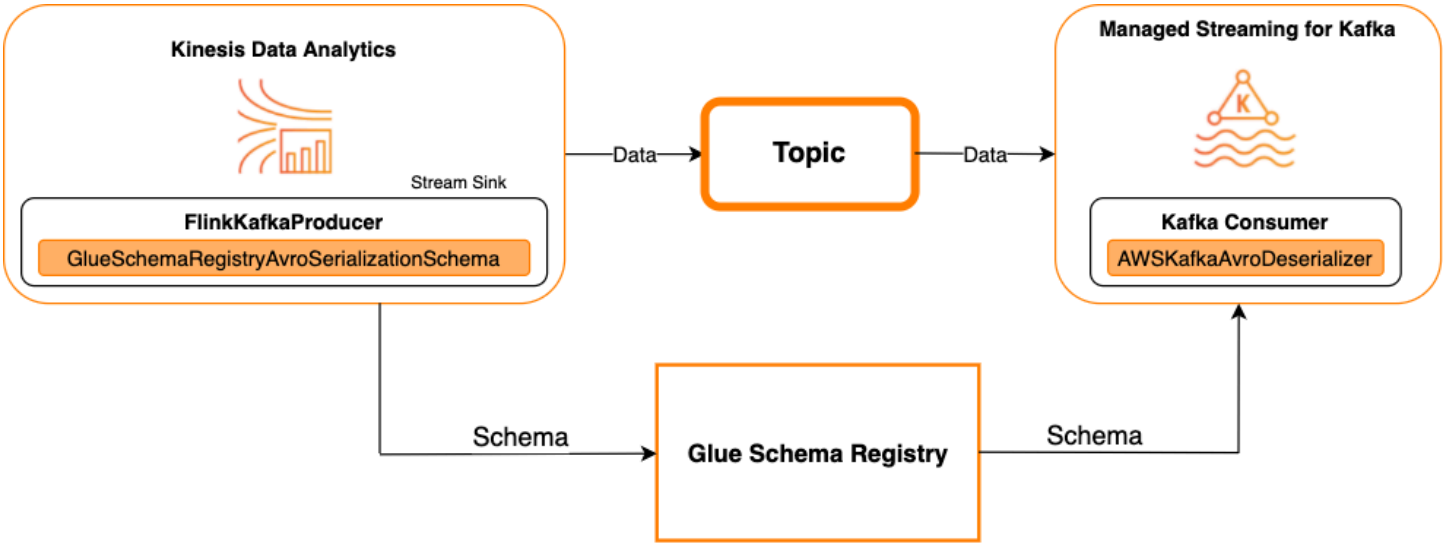
소스형 Kafka

다음 다이어그램에서는 Kafka를 소스로 사용하여 Kinesis Data Streams를 Apache Flink용 Managed Service for Apache Flink와 통합하는 방법을 보여줍니다.



싱크로 Kafka

다음 다이어그램에서는 Kafka를 싱크로 사용하여 Kinesis Data Streams를 Apache Flink용 Managed Service for Apache Flink와 통합하는 방법을 보여줍니다.



Kafka를 소스 또는 싱크로 사용하여 Kafka(또는 Amazon MSK)를 Apache Flink용 Managed Service for Apache Flink와 통합하려면 아래와 같이 코드를 변경합니다. 굵게 표시된 코드 블록을 유사한 섹션의 해당 코드에 추가합니다.

Kafka가 소스인 경우 deserializer 코드를 사용합니다(블록 2). Kafka가 싱크인 경우 serializer 코드(블록 3)를 사용합니다.

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

String topic = "topic";
Properties properties = new Properties();
properties.setProperty("bootstrap.servers", "localhost:9092");
properties.setProperty("group.id", "test");

// block 1
Map<String, Object> configs = new HashMap<>();
configs.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
configs.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
configs.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());

FlinkKafkaConsumer<GenericRecord> consumer = new FlinkKafkaConsumer<>(
    topic,
    // block 2
    GlueSchemaRegistryAvroDeserializationSchema.forGeneric(schema, configs),
    properties);

FlinkKafkaProducer<GenericRecord> producer = new FlinkKafkaProducer<>(
```

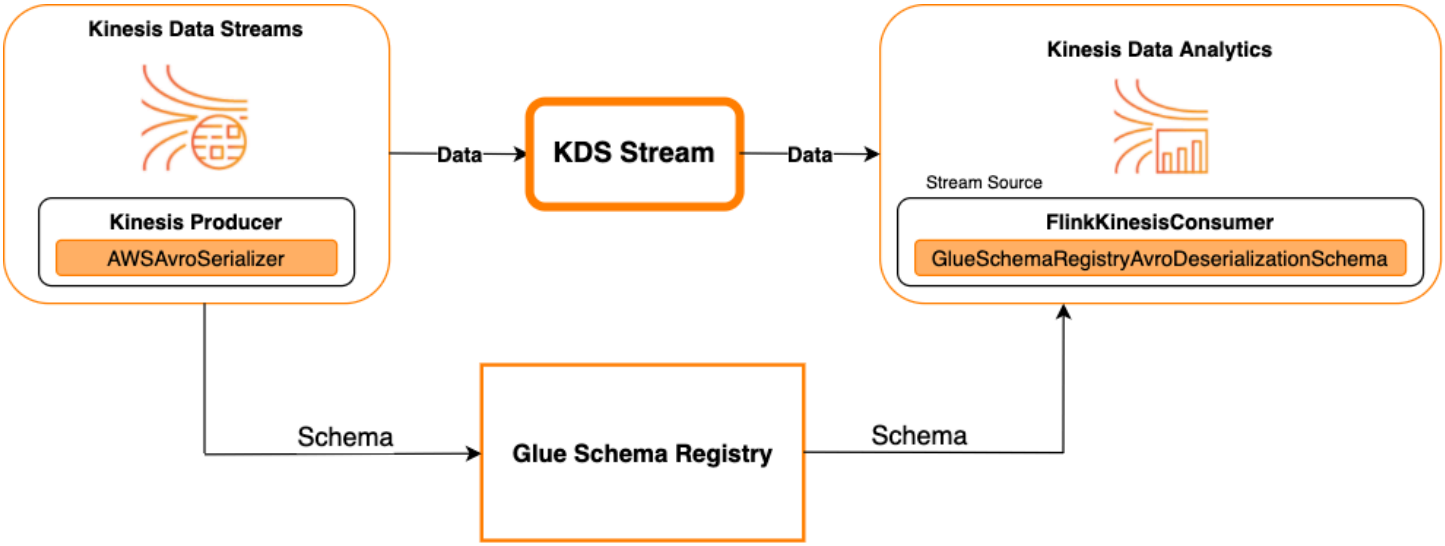
```
    topic,
    // block 3
    GlueSchemaRegistryAvroSerializationSchema.forGeneric(schema, topic, configs),
    properties);

DataStream<GenericRecord> stream = env.addSource(consumer);
stream.addSink(producer);
env.execute();
```

Apache Flink와 Kinesis Data Streams 통합

Kinesis Data Streams를 소스 또는 싱크로 사용하여 Apache Flink용 Managed Service for Apache Flink를 사용할 수 있습니다.

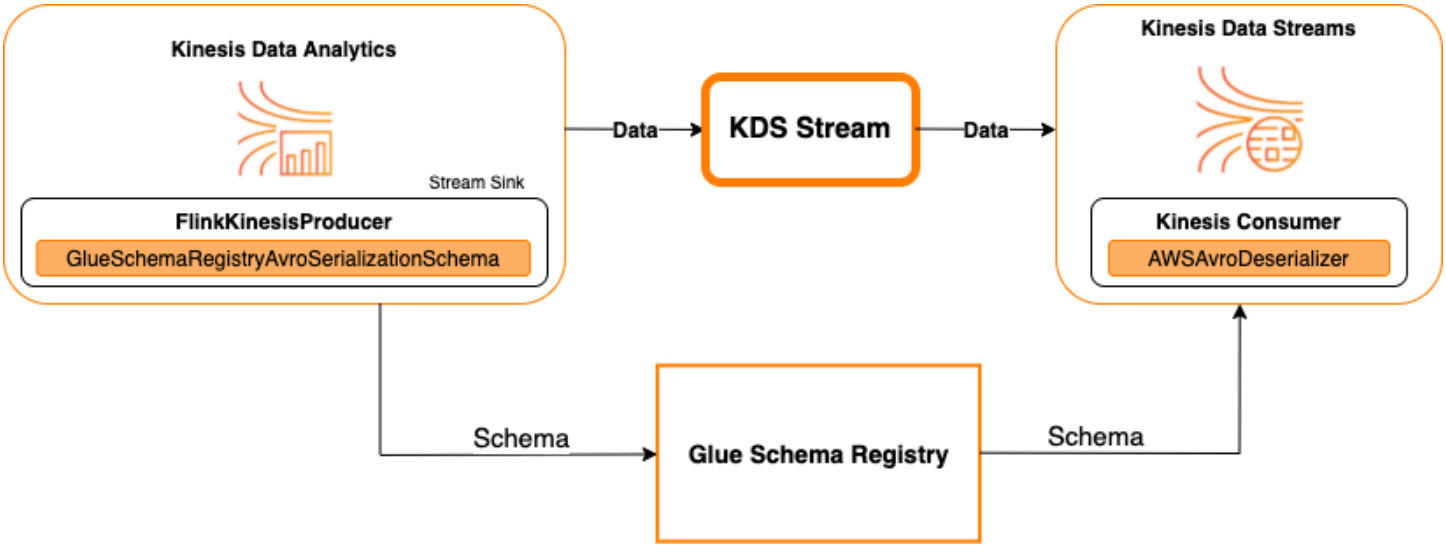
소스로 Kinesis Data Streams

다음 다이어그램에서는 Kinesis Data Streams를 소스로 사용하여 Kinesis Data Streams를 Apache Flink용 Managed Service for Apache Flink와 통합하는 방법을 보여줍니다.



싱크로 Kinesis Data Streams

다음 다이어그램에서는 Kinesis Data Streams를 싱크로 사용하여 Kinesis Data Streams를 Apache Flink용 Managed Service for Apache Flink와 통합하는 방법을 보여줍니다.



Kinesis Data Streams를 소스 또는 싱크로 사용하여 Kinesis Data Streams를 Apache Flink용 Managed Service for Apache Flink와 통합하려면 아래와 같이 코드를 변경합니다. 굵게 표시된 코드 블록을 유사한 섹션의 해당 코드에 추가합니다.

Kinesis Data Streams가 소스인 경우 deserializer 코드(블록 2)를 사용합니다. Kinesis Data Streams가 싱크인 경우 serializer 코드(블록 3)를 사용합니다.

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

String streamName = "stream";
Properties consumerConfig = new Properties();
consumerConfig.put(AWSConfigConstants.AWS_REGION, "aws-region");
consumerConfig.put(AWSConfigConstants.AWS_ACCESS_KEY_ID, "aws_access_key_id");
consumerConfig.put(AWSConfigConstants.AWS_SECRET_ACCESS_KEY, "aws_secret_access_key");
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

// block 1
Map<String, Object> configs = new HashMap<>();
configs.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
configs.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
configs.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());

FlinkKinesisConsumer<GenericRecord> consumer = new FlinkKinesisConsumer<>(
    streamName,
    // block 2
    GlueSchemaRegistryAvroDeserializationSchema.forGeneric(schema, configs),
    properties);
```

```
FlinkKinesisProducer<GenericRecord> producer = new FlinkKinesisProducer<>(
    // block 3
    GlueSchemaRegistryAvroSerializationSchema.forGeneric(schema, topic, configs),
    properties);
producer.setDefaultStream(streamName);
producer.setDefaultPartition("0");

DataStream<GenericRecord> stream = env.addSource(consumer);
stream.addSink(producer);
env.execute();
```

사용 사례: AWS Lambda와 통합

AWS Lambda 함수를 Apache Kafka/Amazon MSK 소비자로 사용하고 AWS Glue Schema Registry를 사용하여 Avro 인코딩 메시지를 역직렬화하려면 MSK Labs 페이지를 방문합니다.

사용 사례: AWS Glue Data Catalog

AWS Glue 테이블은 수동으로 지정하거나 AWS Glue Schema Registry를 참조하여 지정할 수 있는 스키마를 지원합니다. Schema Registry는 데이터 카탈로그와 통합되어 데이터 카탈로그에서 AWS Glue 테이블 또는 파티션을 생성하거나 업데이트할 때 Schema Registry에 저장된 스키마를 선택적으로 사용할 수 있습니다. Schema Registry에서 스키마 정의를 식별하려면 최소한 스키마가 속한 스키마의 ARN을 알아야 합니다. 스키마 정의를 포함하는 스키마의 스키마 버전은 해당 UUID 또는 버전 번호로 참조할 수 있습니다. 버전 번호나 UUID를 몰라도 조회할 수 있는 "최신" 버전이라는 하나의 스키마 버전이 항상 있습니다.

`CreateTable` 또는 `UpdateTable` 작업을 호출할 때 Schema Registry의 기존 스키마에 `SchemaReference`가 있을 수 있는 `StorageDescriptor`를 포함하는 `TableInput` 구조를 전달합니다. 마찬가지로 `GetTable` 또는 `GetPartition` API를 호출할 때 응답에 스키마와 `SchemaReference`가 포함될 수 있습니다. 스키마 참조를 사용하여 테이블 또는 파티션이 생성되면 데이터 카탈로그는 이 스키마 참조에 대한 스키마를 가져오려고 시도합니다. Schema Registry에서 스키마를 찾을 수 없는 경우 `GetTable` 응답에서 빈 스키마를 반환합니다. 그렇지 않으면 응답에 스키마와 스키마 참조가 모두 포함됩니다.

AWS Glue 콘솔에서 작업을 수행할 수도 있습니다.

이러한 작업을 수행하고 스키마 정보를 생성, 업데이트 또는 보려면 `GetSchemaVersion` API에 대한 권한을 제공하는 호출 사용자에게 IAM 역할을 부여해야 합니다.

테이블 추가 또는 테이블에 대한 스키마 업데이트

기존 스키마에서 새 테이블을 추가하면 테이블이 특정 스키마 버전에 바인딩됩니다. 새 스키마 버전이 등록되면 AWS Glue 콘솔의 테이블 보기 페이지에서 또는 UpdateTable 작업(Python: update_table) API를 사용하여 이 테이블 정의를 업데이트할 수 있습니다.

기존 스키마에서 테이블 추가

AWS Glue 콘솔 또는 `CreateTable` API를 사용하여 레지스트리의 스키마 버전에서 AWS Glue 테이블을 생성할 수 있습니다.

AWS Glue API

`CreateTable` API를 호출할 때 Schema Registry의 기존 스키마에 `SchemaReference`가 있고 `StorageDescriptor`가 포함된 `TableInput`을 전달합니다.

AWS Glue 콘솔

AWS Glue 콘솔에서 테이블을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [테이블(Tables)]을 선택합니다.
3. [테이블 추가(Add Tables)] 메뉴에서 [기존 스키마에서 테이블 추가(Add table from existing schema)]를 선택합니다.
4. AWS Glue 개발자 안내서에 따라 테이블 속성 및 데이터 스토어를 구성합니다.
5. [Glue 스키마 선택(Choose a Glue schema)] 페이지에서 스키마가 있는 [레지스트리(Registry)]를 선택합니다.
6. [스키마 이름(Schema name)]을 선택하고 적용할 스키마의 [버전(Version)]을 선택합니다.
7. 스키마 미리 보기를 검토하고 [다음(Next)]을 선택합니다.
8. 테이블을 검토하고 생성합니다.

테이블에 적용된 스키마 및 버전은 테이블 목록의 [Glue 스키마(Glue schema)] 열에 나타납니다. 테이블을 보고 자세한 세부 정보를 확인할 수 있습니다.

테이블에 대한 스키마 업데이트

새 스키마 버전을 사용할 수 있게 되면 UpdateTable 작업(Python: update_table) API 또는 AWS Glue 콘솔을 사용하여 테이블의 스키마를 업데이트할 수 있습니다.

> **Important**
>
> AWS Glue 스키마가 수동으로 지정된 기존 테이블의 스키마를 업데이트할 때 Schema Registry에서 참조하는 새 스키마가 호환되지 않을 수 있습니다. 이로 인해 작업이 실패할 수 있습니다.

AWS Glue API

`UpdateTable` API를 호출할 때 Schema Registry의 기존 스키마에 `SchemaReference`가 있고 `StorageDescriptor`가 포함된 `TableInput`을 전달합니다.

AWS Glue 콘솔

AWS Glue 콘솔에서 테이블에 대한 스키마를 업데이트하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 [데이터 카탈로그(Data catalog)]에서 [테이블(Tables)]을 선택합니다.
3. 테이블 목록에서 테이블을 봅니다.
4. 새 버전에 대해 알려주는 상자에서 [스키마 업데이트(Update schema)]를 클릭합니다.
5. 현재 스키마와 새 스키마의 차이점을 검토합니다.
6. [모든 스키마 차이점 표시(Show all schema differences)]를 선택 더 많은 세부 정보를 볼 수 있습니다.
7. [테이블 저장(Save table)]을 선택하여 새 버전을 수락합니다.

사용 사례: AWS Glue 스트리밍

AWS Glue 스트리밍은 출력 싱크에 쓰기 전에 스트리밍 소스의 데이터를 소비하고 ETL 작업을 수행합니다. 입력 스트리밍 소스는 데이터 테이블을 사용하여 지정하거나 소스 구성을 지정하여 직접 지정할 수 있습니다.

AWS Glue 스트리밍은 에 AWS Glue 스키마 레지스트리에 있는 스키마를 사용하여 생성된 스트리밍 소스에 대한 데이터 카탈로그 테이블을 지원합니다. AWS Glue 스키마 레지스트리에서 스키마를 생성하고 이 스키마를 사용하는 스트리밍 소스가 있는 AWS Glue 테이블을 생성할 수 있습니다. 이 AWS Glue 테이블을 입력 스트림의 데이터를 역직렬화하기 위한 AWS Glue 스트리밍 작업에 대한 입력으로 사용할 수 있습니다.

여기서 유의해야 할 점은 AWS Glue 스키마 레지스트리의 스키마가 변경되면 AWS Glue 스트리밍 작업을 다시 시작해야 스키마의 변경 사항이 반영된다는 것입니다.

사용 사례: Apache Kafka Streams

Apache Kafka Streams API는 Apache Kafka에 저장된 데이터를 처리하고 분석하기 위한 클라이언트 라이브러리입니다. 이 섹션에서는 데이터 스트리밍 애플리케이션에서 스키마를 관리하고 시행할 수 있는 AWS Glue Schema Registry와 Apache Kafka Streams의 통합에 대해 설명합니다. Apache Kafka Streams에 대한 자세한 내용은 Apache Kafka Streams를 참조하세요.

SerDes 라이브러리와의 통합

Streams 애플리케이션을 구성할 수 있는 `GlueSchemaRegistryKafkaStreamsSerde` 클래스가 있습니다.

Kafka Streams 애플리케이션 예 코드

Apache Kafka Streams 애플리케이션 내에서 AWS Glue Schema Registry를 사용하려면

1. Kafka Streams 애플리케이션을 구성합니다.

```
final Properties props = new Properties();
    props.put(StreamsConfig.APPLICATION_ID_CONFIG, "avro-streams");
    props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
    props.put(StreamsConfig.CACHE_MAX_BYTES_BUFFERING_CONFIG, 0);
    props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG,
 Serdes.String().getClass().getName());
    props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG,
 AWSKafkaAvroSerDe.class.getName());
    props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");

    props.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
    props.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
    props.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());
 props.put(AWSSchemaRegistryConstants.DATA_FORMAT, DataFormat.AVRO.name());
```

2. 주제 avro-input에서 스트림을 생성합니다.

```
StreamsBuilder builder = new StreamsBuilder();
final KStream<String, GenericRecord> source = builder.stream("avro-input");
```

3. 데이터 레코드를 처리합니다(이 예에서는 favorite_color 값이 pink이거나 양의 값이 15인 레코드를 필터링함).

```
final KStream<String, GenericRecord> result = source
    .filter((key, value) -
> !"pink".equals(String.valueOf(value.get("favorite_color"))));
    .filter((key, value) -> !"15.0".equals(String.valueOf(value.get("amount"))));
```

4. avro-output 주제에 결과를 다시 씁니다.

```
result.to("avro-output");
```

5. Apache Kafka Streams 애플리케이션을 시작합니다.

```
KafkaStreams streams = new KafkaStreams(builder.build(), props);
streams.start();
```

구현 결과

이 결과는 3단계에서 favorite_color "pink" 또는 값 "15.0"으로 필터링된 레코드의 필터링 프로세스를 보여줍니다.

필터링 전 레코드:

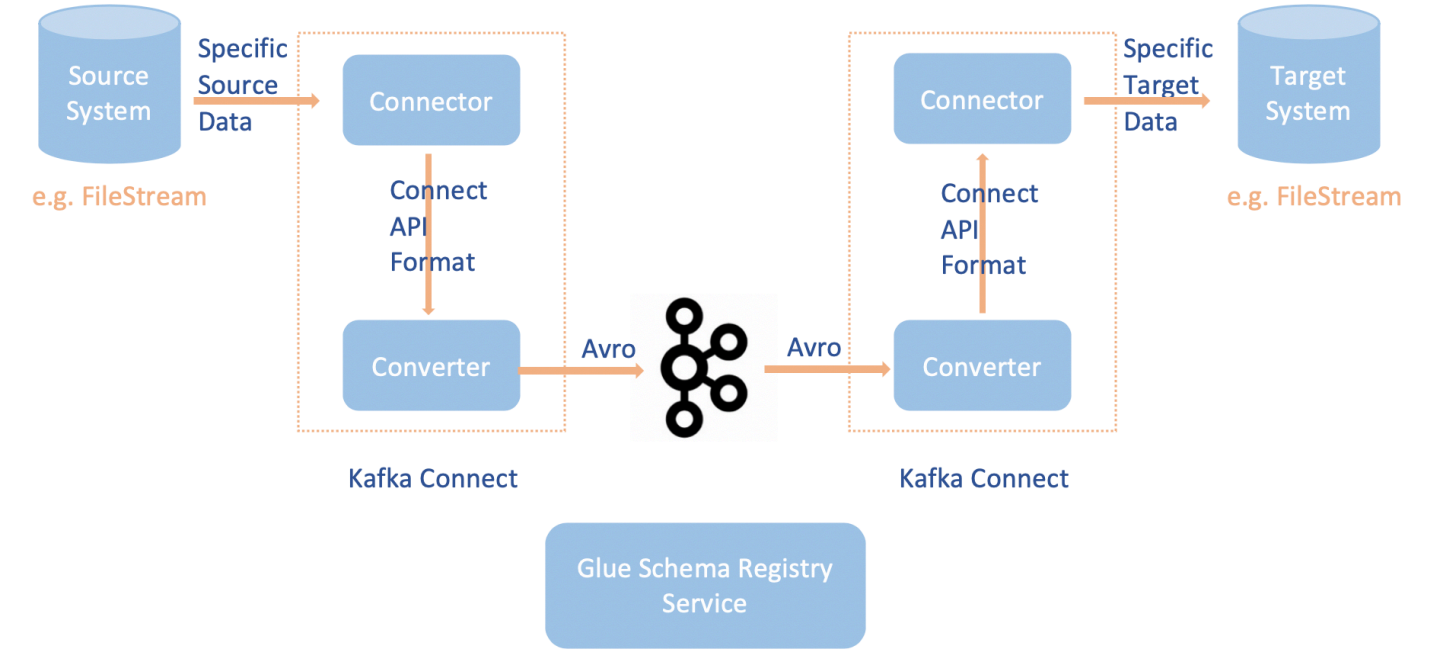
```
{"name": "Sansa", "favorite_number": 99, "favorite_color": "white"}
{"name": "Harry", "favorite_number": 10, "favorite_color": "black"}
{"name": "Hermione", "favorite_number": 1, "favorite_color": "red"}
{"name": "Ron", "favorite_number": 0, "favorite_color": "pink"}
{"name": "Jay", "favorite_number": 0, "favorite_color": "pink"}

{"id": "commute_1","amount": 3.5}
{"id": "grocery_1","amount": 25.5}
{"id": "entertainment_1","amount": 19.2}
{"id": "entertainment_2","amount": 105}
 {"id": "commute_1","amount": 15}
```

필터링 후 레코드:

```
{"name": "Sansa", "favorite_number": 99, "favorite_color": "white"}
{"name": "Harry", "favorite_number": 10, "favorite_color": "black"}
{"name": "Hermione", "favorite_number": 1, "favorite_color": "red"}
{"name": "Ron", "favorite_number": 0, "favorite_color": "pink"}

{"id": "commute_1","amount": 3.5}
{"id": "grocery_1","amount": 25.5}
{"id": "entertainment_1","amount": 19.2}
{"id": "entertainment_2","amount": 105}
```

사용 사례: Apache Kafka Connect

Apache Kafka Connect와 AWS Glue Schema Registry의 통합으로 커넥터에서 스키마 정보를 가져올 수 있습니다. Apache Kafka 변환기는 Apache Kafka 내의 데이터 포맷과 이를 Apache Kafka Connect 데이터로 변환하는 방법을 지정합니다. 모든 Apache Kafka Connect 사용자는 Apache Kafka에서 로드하거나 Apache Kafka에 저장할 때 데이터를 원하는 포맷을 기반으로 이러한 변환기를 구성해야 합니다. 이러한 방식으로 Apache Kafka Connect 데이터를 AWS Glue Schema Registry(예: Avro)에서 사용되는 유형으로 변환하고 serializer를 사용하여 스키마를 등록하고 직렬화를 수행하는 변환기를 정의할 수 있습니다. 그런 다음 변환기는 deserializer를 사용하여 Apache Kafka에서 수신한 데이터를 역직렬화하고 이를 다시 Apache Kafka Connect 데이터로 변환할 수도 있습니다. 다음은 예제 워크플로 다이어그램입니다.



1. AWS Glue Schema Registry용 Github 리포지토리를 복제하여 `aws-glue-schema-registry` 프로젝트를 설치합니다.

```
git clone git@github.com:awslabs/aws-glue-schema-registry.git
cd aws-glue-schema-registry
mvn clean install
mvn dependency:copy-dependencies
```

2. 독립 실행형 모드에서 Apache Kafka Connect를 사용하려는 경우 이 단계에 대한 아래 지침을 사용하여 connect-standalone.properties를 업데이트합니다. 분산 모드에서 Apache Kafka Connect를 사용하려는 경우 동일한 지침을 사용하여 connect-avro-distributed.properties를 업데이트합니다.

   a. Apache Kafka 연결 속성 파일에도 다음 속성을 추가합니다.

```
key.converter.region=aws-region
value.converter.region=aws-region
key.converter.schemaAutoRegistrationEnabled=true
value.converter.schemaAutoRegistrationEnabled=true
key.converter.avroRecordType=GENERIC_RECORD
value.converter.avroRecordType=GENERIC_RECORD
```

   b. kafka-run-class.sh 아래의 [시작 모드(Launch mode)] 섹션에 아래 명령을 추가합니다.

```
-cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
```

3. kafka-run-class.sh 아래의 [시작 모드(Launch mode)] 섹션에 아래 명령을 추가합니다.

```
-cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
```

형식은 다음과 같아야 합니다.

```
# Launch mode
if [ "x$DAEMON_MODE" = "xtrue" ]; then
  nohup "$JAVA" $KAFKA_HEAP_OPTS $KAFKA_JVM_PERFORMANCE_OPTS $KAFKA_GC_LOG_OPTS
 $KAFKA_JMX_OPTS $KAFKA_LOG4J_OPTS -cp $CLASSPATH:"/Users/johndoe/aws-glue-schema-
registry/target/dependency/*" $KAFKA_OPTS "$@" > "$CONSOLE_OUTPUT_FILE" 2>&1 < /dev/
null &
else
  exec "$JAVA" $KAFKA_HEAP_OPTS $KAFKA_JVM_PERFORMANCE_OPTS $KAFKA_GC_LOG_OPTS
 $KAFKA_JMX_OPTS $KAFKA_LOG4J_OPTS -cp $CLASSPATH:"/Users/johndoe/aws-glue-schema-
registry/target/dependency/*" $KAFKA_OPTS "$@"
fi
```

4. bash를 사용하는 경우 아래 명령을 실행하여 bash_profile에 CLASSPATH를 설정합니다. 다른 셸의 경우 그에 따라 환경을 업데이트합니다.

```
echo 'export GSR_LIB_BASE_DIR=<>' >>~/.bash_profile
echo 'export GSR_LIB_VERSION=1.0.0' >>~/.bash_profile
echo 'export KAFKA_HOME=<your Apache Kafka installation directory>' >>~/.bash_profile
echo 'export CLASSPATH=$CLASSPATH:$GSR_LIB_BASE_DIR/avro-kafkaconnect-converter/
target/schema-registry-kafkaconnect-converter-$GSR_LIB_VERSION.jar:$GSR_LIB_BASE_DIR/
common/target/schema-registry-common-$GSR_LIB_VERSION.jar:$GSR_LIB_BASE_DIR/
avro-serializer-deserializer/target/schema-registry-serde-$GSR_LIB_VERSION.jar'
 >>~/.bash_profile
source ~/.bash_profile
```

5. (선택 사항) 간단한 파일 소스로 테스트하려면 파일 소스 커넥터를 복제합니다.

```
git clone https://github.com/mmolimar/kafka-connect-fs.git
cd kafka-connect-fs/
```

a. 소스 커넥터 구성에서 데이터 포맷을 Avro로, 파일 리더를 AvroFileReader으로 편집하고 읽고 있는 파일 경로에서 예제 Avro 객체를 업데이트합니다. 예:

```
vim config/kafka-connect-fs.properties
```

```
fs.uris=<path to a sample avro object>
policy.regexp=^.*\.avro$
file_reader.class=com.github.mmolimar.kafka.connect.fs.file.reader.AvroFileReader
```

b. 소스 커넥터를 설치합니다.

```
mvn clean package
echo "export CLASSPATH=\$CLASSPATH:\"\$(find target/ -type f -name '*.jar'| grep
 '\-package' | tr '\n' ':')\"" >>~/.bash_profile
source ~/.bash_profile
```

c. *<your Apache Kafka installation directory>*/config/connect-file-sink.properties 아래의 싱크 속성을 업데이트하고 주제 이름과 출력 파일 이름을 업데이트합니다.

```
file=<output file full path>
topics=<my topic>
```

6. 소스 커넥터를 시작합니다 (이 예에서는 파일 소스 커넥터임).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties config/kafka-connect-fs.properties
```

7. 싱크 커넥터를 실행합니다(이 예에서는 파일 싱크 커넥터임).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties $KAFKA_HOME/config/connect-file-sink.properties
```

Kafka Connect 사용 예는 AWS Glue Schema Registry용 Github 리포지토리의 integration-tests 폴더에 있는 run-local-tests.sh 스크립트를 참조하세요.

서드 파티 Schema Registry에서 AWS Glue Schema Registry로 마이그레이션

서드 파티 Schema Registry에서 AWS Glue Schema Registry로의 마이그레이션은 기존의 현재 서드 파티 Schema Registry에 종속됩니다. Apache Kafka 주제에 서드 파티 스키마 레지스트리를 사용하여 전송된 레코드가 있는 경우 소비자는 해당 레코드를 역직렬화하기 위해 서드 파티 스키마 레지스트리가 필요합니다. `AWSKafkaAvroDeserializer`는 서드 파티 deserializer를 가리키고 해당 레코드를 역직렬화하는 데 사용되는 보조 deserializer 클래스를 지정하는 기능을 제공합니다.

서드 파티 스키마의 사용 중지에는 두 가지 기준이 있습니다. 첫째, 서드 파티 스키마 레지스트리를 사용하는 Apache Kafka 주제의 레코드가 소비자에게 더 이상 필요하지 않은 경우에만 사용 중지가 발생할 수 있습니다. 둘째, 해당 주제에 대해 지정된 보존 기간에 따라 Apache Kafka 주제에서 만료로 사용 중지가 발생할 수 있습니다. 영구 보존이 있는 주제가 있는 경우 AWS Glue Schema Registry로 계속 마이그레이션할 수 있지만 서드 파티 Schema Registry를 사용 중지할 수는 없습니다. 해결 방법으로 애플리케이션 또는 Mirror Maker 2를 사용하여 현재 주제에서 읽고 AWS Glue Schema Registry를 사용하여 새 주제를 생성할 수 있습니다.

서드 파티 Schema Registry에서 AWS Glue Schema Registry로 마이그레이션하려면

1. AWS Glue Schema Registry에서 레지스트리를 생성하거나 기본 레지스트리를 사용합니다.
2. 소비자를 중지합니다. AWS Glue Schema Registry를 기본 deserializer로 포함하고 서드 파티 Schema Registry를 보조로 포함하도록 수정합니다.
   - 소비자 속성을 설정합니다. 이 예에서 secondary_deserializer는 다른 deserializer로 설정됩니다. 동작은 다음과 같습니다. 소비자는 Amazon MSK에서 레코드를 검색하고 먼저 `AWSKafkaAvroDeserializer`를 사용하려고 시도합니다. AWS Glue Schema Registry 스키마에 대한 Avro 스키마 ID가 포함된 매직 바이트를 읽을 수 없는 경우 `AWSKafkaAvroDeserializer`는 secondary_deserializer에서 제공되는 deserializer 클래스를 사용하려고 시도합니다. 보조 deserializer와 관련된 속성도 아래와 같이 schema_registry_url_config 및 specific_avro_reader_config와 같은 소비자 속성에 제공해야 합니다.

```
consumerProps.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 StringDeserializer.class.getName());
consumerProps.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 AWSKafkaAvroDeserializer.class.getName());
consumerProps.setProperty(AWSSchemaRegistryConstants.AWS_REGION,
 KafkaClickstreamConsumer.gsrRegion);
```

```
consumerProps.setProperty(AWSSchemaRegistryConstants.SECONDARY_DESERIALIZER,
 KafkaAvroDeserializer.class.getName());
consumerProps.setProperty(KafkaAvroDeserializerConfig.SCHEMA_REGISTRY_URL_CONFIG,
 "URL for third-party schema registry");
consumerProps.setProperty(KafkaAvroDeserializerConfig.SPECIFIC_AVRO_READER_CONFIG,
 "true");
```

3. 소비자를 다시 시작합니다.
4. 생산자를 중지하고 생산자가 AWS Glue Schema Registry를 가리키도록 합니다.
   a. 생산자 속성을 설정합니다. 이 예에서 생산자는 기본 레지스트리 및 자동 등록 스키마 버전을 사용합니다.

```
producerProps.setProperty(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
 StringSerializer.class.getName());
producerProps.setProperty(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
 AWSKafkaAvroSerializer.class.getName());
producerProps.setProperty(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2");
producerProps.setProperty(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.SPECIFIC_RECORD.getName());
producerProps.setProperty(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING,
 "true");
```

5. (선택 사항) 기존 스키마 및 스키마 버전을 현재 서드 파티 Schema Registry에서 AWS Glue Schema Registry로, AWS Glue Schema Registry의 기본 레지스트리 또는 AWS Glue Schema Registry의 기본이 아닌 특정 레지스트리로 수동으로 이동합니다. 이는 JSON 포맷의 서드 파티 Schema Registry에서 스키마를 내보내고 AWS Management Console 또는 AWS CLI을 사용하여 AWS Glue Schema Registry에서 새 스키마를 생성하여 수행할 수 있습니다.

   이 단계는 AWS CLI 및 AWS Management Console을 사용하여 새로 생성된 스키마 버전에 대해 이전 스키마 버전과의 호환성 검사를 사용해야 하거나 생산자가 스키마 버전 자동 등록이 설정된 상태로 새 스키마로 메시지를 보낼 때 중요할 수 있습니다.
6. 생산자를 시작합니다.

데이터에 연결

AWS Glue 연결은 특정 데이터 스토어에 대한 로그인 자격 증명, URI 문자열, Virtual Private Cloud(VPC) 정보 등을 저장하는 데이터 카탈로그 객체입니다. AWS Glue 크롤러, 작업 및 개발 엔드포인트는 특정 유형의 데이터 스토어에 액세스하기 위해 연결을 사용합니다. 소스와 대상 모두에 대한 연결을 사용하고 여러 크롤러 또는 추출, 전환, 적재(ETL) 작업에서 동일한 연결을 재사용할 수 있습니다.

최신 버전의 AWS Glue 연결 스키마는 AWS Glue, Amazon Athena, Amazon SageMaker AI Unified Studio 등과 같은 AWS 서비스 및 애플리케이션 전반에서 데이터 연결을 관리하는 통일된 방법을 제공합니다.

커넥터 및 연결 사용 개요

특정 데이터 스토어에 연결하는 데 필요한 속성을 포함하는 연결입니다. 연결을 생성하면 AWS Glue Data Catalog에 저장됩니다. 커넥터를 선택한 다음 해당 커넥터를 기반으로 연결을 생성합니다.

AWS Marketplace에서 기본적으로 지원되지 않는 데이터 스토어에 대한 커넥터를 구독한 다음 연결을 생성할 때 해당 커넥터를 사용할 수 있습니다. 개발자는 자체 커넥터를 생성하여 연결 생성 시 사용할 수 있습니다.

> **Note**
>
> AWS Glue Studio에서 사용자 지정 또는 AWS Marketplace 커넥터를 사용하여 생성된 연결은 유형이 `UNKNOWN`으로 설정되어 AWS Glue 콘솔에 표시됩니다.

다음 단계는 AWS Glue Studio에서 커넥터를 사용하는 전체 프로세스에 대해 설명합니다.

1. AWS Marketplace에서 커넥터를 구독하거나 고유한 커넥터를 개발하여 AWS Glue Studio에 업로드합니다. 자세한 내용은 AWS Glue Studio에 커넥터 추가 섹션을 참조하세요.
2. 커넥터 사용 정보를 검토합니다. 이 정보는 커넥터 제품 페이지의 [사용(Usage)] 탭에서 찾을 수 있습니다. 예를 들어 제품 페이지 AWS Glue Connector for Google BigQuery(Google BigQuery용 커넥터)에서 Usage(사용) 탭을 클릭하면 Additional Resources(추가 리소스) 섹션에서 이 커넥터 사용에 대한 블로그 링크를 볼 수 있습니다.
3. 연결을 생성합니다. 사용할 커넥터를 선택하고 로그인 자격 증명, URI 문자열 및 Virtual Private Cloud(VPC) 정보와 같은 연결에 대한 추가 정보를 제공합니다. 자세한 내용은 커넥터에 대한 연결 생성 섹션을 참조하세요.

4. 작업에 대한 IAM 역할을 생성합니다. 작업은 생성할 때 지정한 [IAM role(IAM 역할)]의 권한을 가정합니다. 이 IAM 역할에는 데이터 스토어를 인증하고, 데이터 스토어에서 데이터를 추출하고, 데이터 스토어에 데이터를 쓰는 데 필요한 권한이 있어야 합니다.
5. ETL 작업을 생성하고 ETL 작업에 대한 데이터 원본 속성을 구성합니다. 사용자 정의 커넥터 공급자의 지시에 따라 연결 옵션 및 인증 정보를 제공합니다. 자세한 내용은 사용자 정의 커넥터로 작업 작성 섹션을 참조하세요.
6. AWS Glue Studio에서 시각적 ETL 작업 시작에 설명된 대로 변환 또는 추가 데이터 스토어를 추가하여 ETL 작업을 사용자 지정합니다.
7. 데이터 대상에 커넥터를 사용하는 경우 ETL 작업에 대한 데이터 대상 속성을 구성합니다. 사용자 정의 커넥터 공급자의 지시에 따라 연결 옵션 및 인증 정보를 제공합니다. 자세한 내용은 the section called “사용자 정의 커넥터로 작업 작성” 섹션을 참조하세요.
8. 작업 속성 수정에 설명된 대로 작업 속성을 구성하여 작업 실행 환경을 사용자 지정합니다.
9. 작업을 실행합니다.

통합 연결

AWS는 최근에 "SageMaker LakeHouse 연결" 또는 "AWS Glue 통합 연결"이라는 새로운 기능을 도입했습니다. 이 기능을 사용하면 AWS Glue 및 Amazon Athena와 같은 여러 AWS 서비스에서 사용할 수 있는 연결을 생성할 수 있습니다. Amazon Athena에서 데이터 소스를 생성하면 AWS Glue 연결 입력을 참조하는 섹션이 표시됩니다. 이 경우 Amazon Athena는 AWS Glue 연결의 `AthenaProperties` 섹션에 있는 Amazon Athena 관련 속성을 포함하여 연결을 생성합니다.

반대로 AWS Glue에서 직접 연결을 생성하는 경우 연결의 `ConnectionProperties` 및 `SparkProperties` 섹션에 저장될 AWS Glue 및 Apache Spark 관련 속성만 입력하라는 메시지가 표시됩니다.

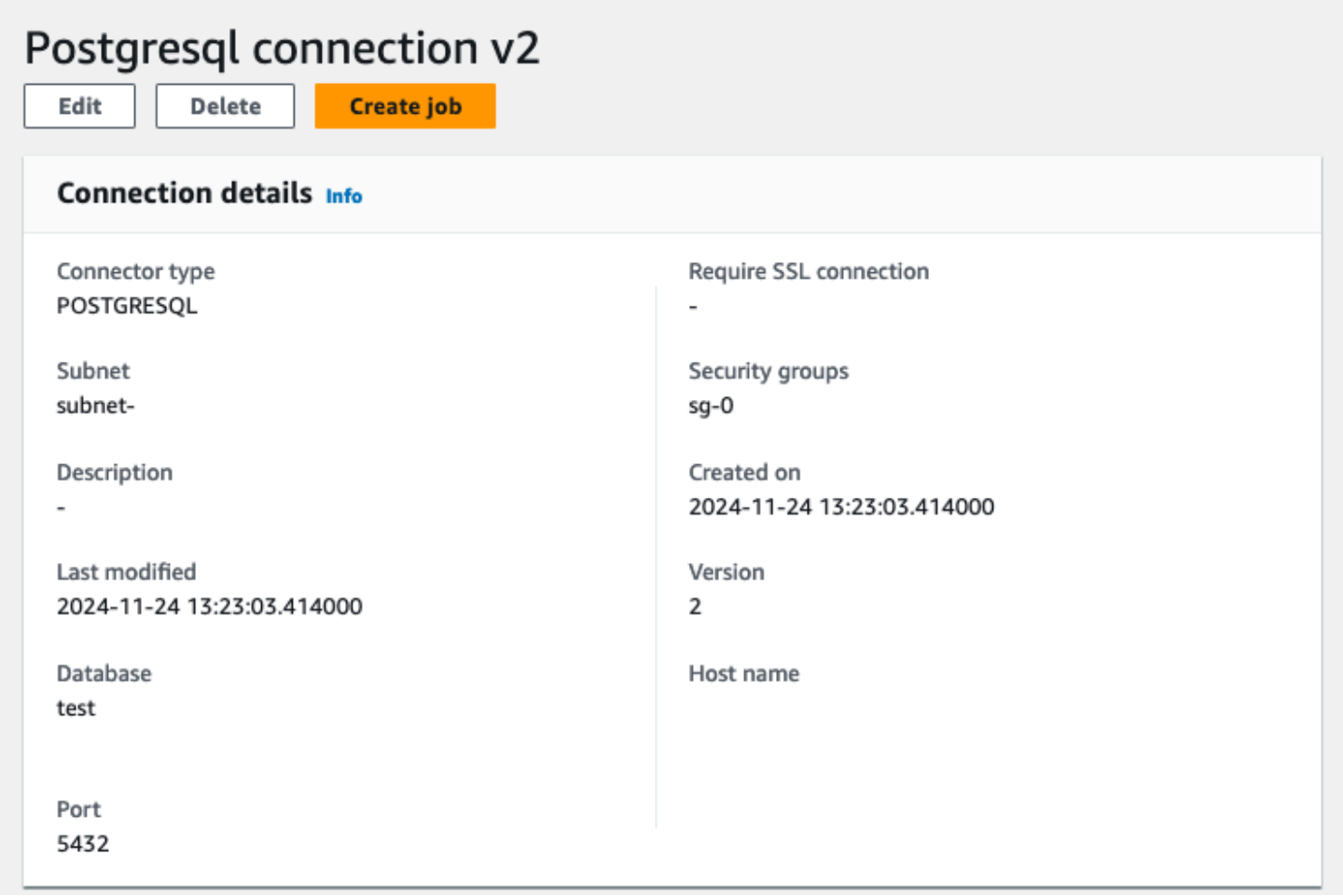
두 시나리오 모두 "통합 연결"이 생성되지만 Amazon Athena에서 생성된 연결은 Amazon Athena 내에서만 사용하도록 구성되고, AWS Glue에서 생성된 연결은 AWS Glue 내에서만 사용하도록 구성됩니다. 하지만 두 서비스 모두에서 사용할 수 있도록 누락된 속성(Amazon Athena 또는 Spark 속성)으로 연결을 업데이트할 수 있습니다. Amazon SageMaker AI Unified Studio는 AWS Glue 연결에서 필요한 모든 속성(`ConnectionProperties`, `AthenaProperties`, `SparkProperties`)을 채워 이를 자동으로 처리하여 AWS Glue 및 Amazon Athena 모두에서 연결을 사용할 수 있도록 합니다.

이를 "통합 연결"이라 하지만 두 서비스에서 사용하도록 적절하게 구성되지 않은 한 AWS Glue 또는 Amazon Athena에서 개별적으로 생성된 연결은 실제로 통합되지 않습니다. SageMaker Unified Studio를 통해 생성된 연결만 실제로 통합되고 여러 서비스에서 즉시 사용할 수 있습니다.

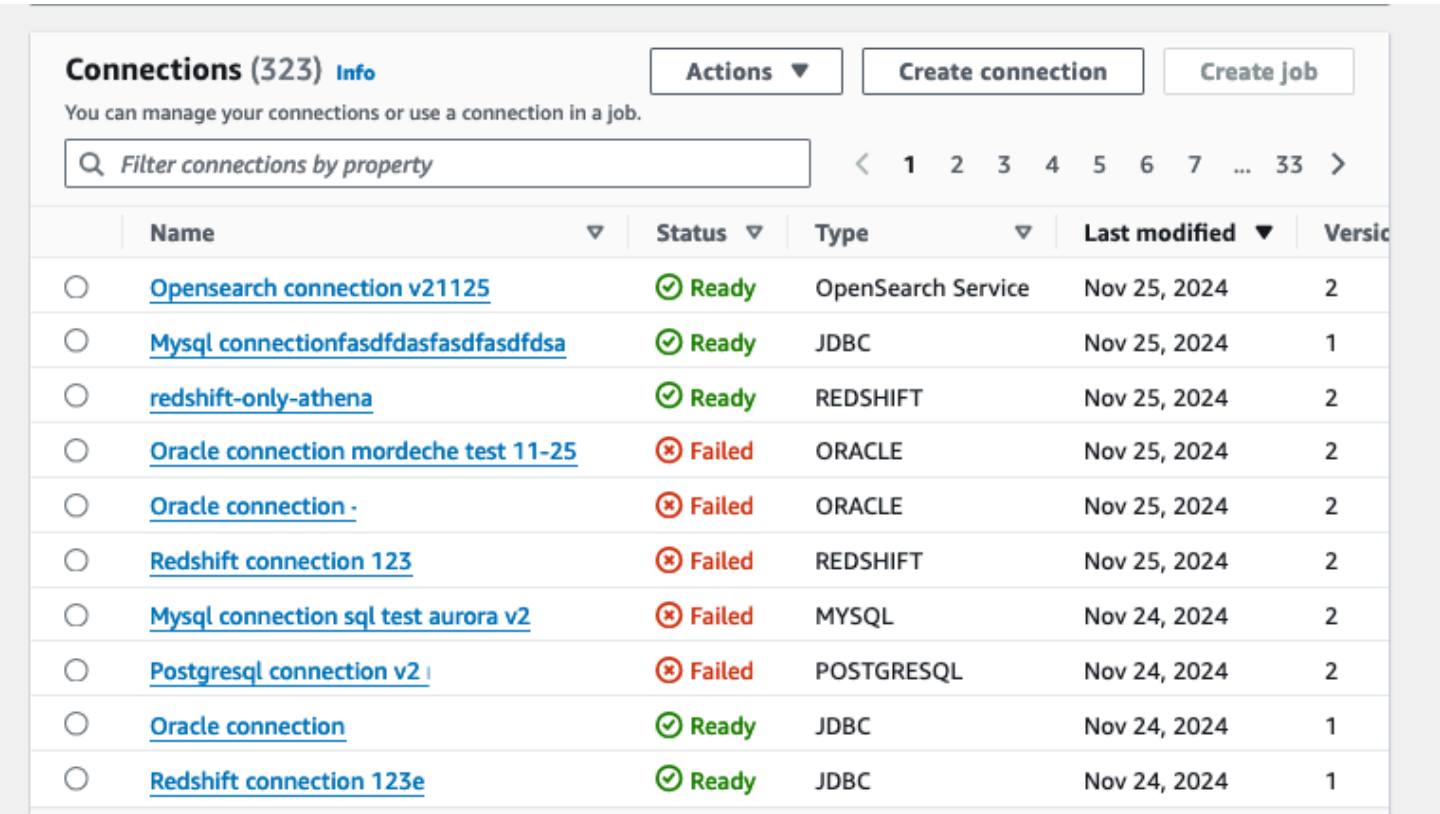
추가로 AWS Glue에서 생성된 연결은 Amazon Athena에 표시되지 않는데, Amazon Athena에 AWS Glue 연결에 대한 참조를 포함하지만 AWS Glue 연결 자체는 아닌 데이터 소스가 표시되기 때문입니다. 마찬가지로 Amazon Athena에서 생성된 연결은 AWS Glue Studio에 표시되지 않는데, AWS Glue Studio에서 AWS Glue에 필요한 설정으로 구성되지 않은 연결을 필터링하기 때문입니다.

AWS Glue Studio에서 기본적으로 통합 연결을 생성합니다. AWS Glue 콘솔에서는 연결 페이지, 연결 세부 정보 페이지 및 작업 세부 정보 페이지의 연결 테이블에서 연결 버전을 확인할 수 있습니다.

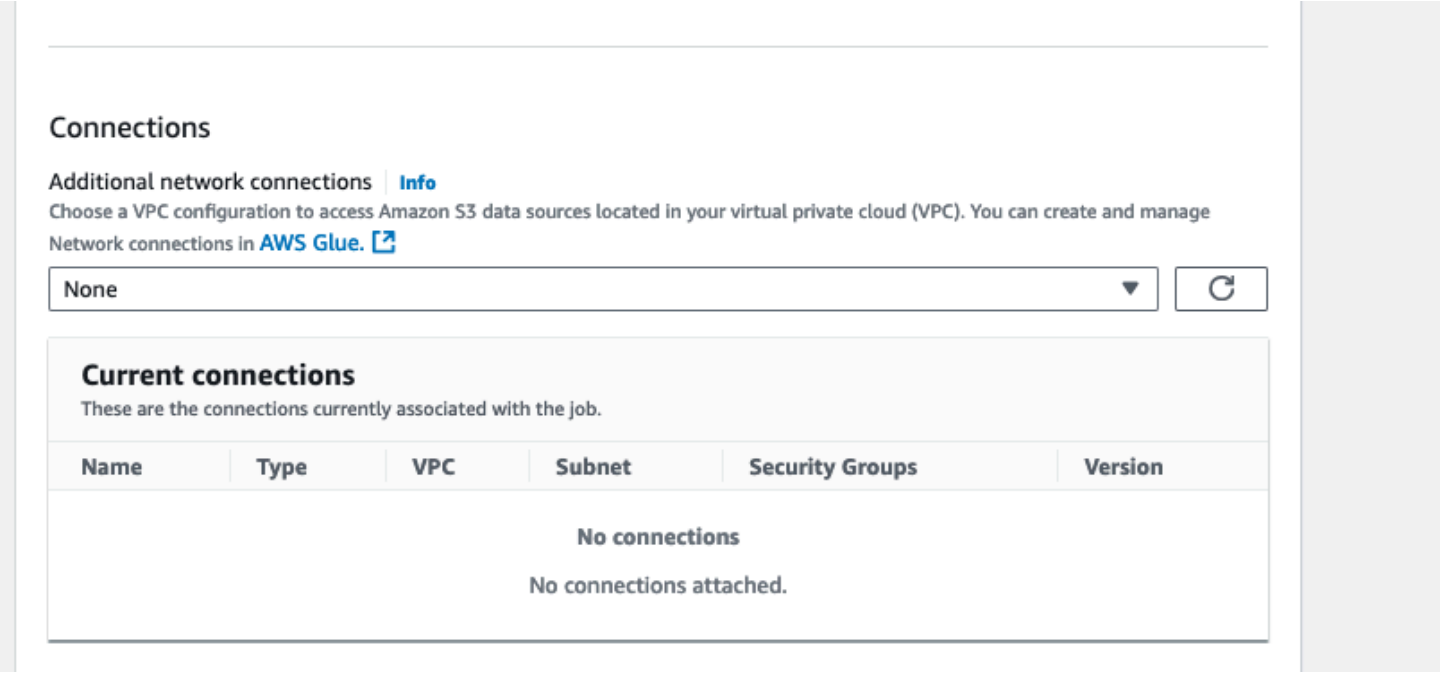
연결 버전은 연결 세부 정보에 표시됩니다.



연결 버전은 모든 연결을 볼 때도 표시됩니다.



마지막으로, 연결 버전은 작업에 대한 작업 세부 정보 탭에 표시됩니다.



버전 2 연결에서는 다음과 같은 확장된 데이터 연결 기능을 사용할 수 있습니다.

- 연결 유형 검색: 표준화된 템플릿을 사용하여 연결을 생성할 수 있도록 지원합니다. AWS Glue는 사용자가 액세스할 수 있는 연결 유형과 지정된 연결 유형에 대한 필수 입력과 선택적 입력을 자동으로 검색합니다.
- 재사용성: AWS Glue, Amazon Athena, Amazon SageMaker AI 같은 AWS 데이터 처리 엔진 및 도구 전반에서 연결 정의를 재사용할 수 있습니다. 이제 연결에는 ConnectionProperties에 저장된 공통 속성 외에도 컴퓨팅 환경/서비스별 연결 속성을 지정하는 데 사용할 수 있는 AthenaProperties, SparkProperties, PythonProperties가 포함되어 있습니다. Athena는 이제 AWS Glue에서 AthenaProperties 속성 맵의 Athena 관련 속성을 지정하여 연결을 생성합니다.
- 데이터 미리 보기: 연결된 소스에서 메타데이터를 찾아보고 데이터를 미리 볼 수 있는 기능입니다.
- 커넥터 메타데이터: 재사용 가능한 연결을 사용하여 테이블 메타데이터를 검색할 수 있습니다.
- 서비스 연결 보안 암호: 사용자는 `CreateConnection` 요청에 필요한 OAuth, 기본 또는 사용자 지정 인증 자격 증명을 제공할 수 있습니다. CreateConnection API는 사용자를 대신하여 사용자 계정에 서비스 연결 보안 암호를 생성하고 자격 증명을 저장합니다.

지원되는 인증 유형

통합 연결은 다음 인증 유형을 지원합니다.

- 기본 - 대부분의 데이터베이스 연결 유형 및 기존 AWS Glue 연결 유형은 사용자 이름과 암호로 이루어진 기본 인증을 지원합니다. 이전에는 SecretsManager의 키 이름 지정이 커넥터별로 달랐습니다(예: user, username, userName, opensearch.net.http.auth.user 등). 여기서 통합 연결은 기본 인증 연결 유형을 USERNAME 및 PASSWORD 키로 표준화했습니다.
- OAUTH2 - 새로 시작된 SaaS 연결 유형의 대부분은 OAuth2 프로토콜을 지원합니다.
- 사용자 지정 - 몇 가지 연결 유형에는 Google BigQuery와 같은 몇 가지 다른 인증 메커니즘이 있으며, 이 경우 사용자는 Google BigQuery에서 가져오는 JSON을 제공해야 합니다.

고려 사항

데이터 소스를 위한 통합 연결을 생성할 때 다음 차이점을 고려하세요.

- AWS Glue Studio를 통해 통합 연결을 생성할 때 사용자 자격 증명은 연결 자체 대신에 AWS Secrets Manager에 저장됩니다. 즉, 이제 작업에 Secrets Manager에 대한 액세스 권한이 필요합니다.
- VPC에서 작업이 실행되는 경우 AWS Secrets Manager 및 Secure Token Service(STS)에 액세스하려면 VPC 엔드포인트 또는 NAT 게이트웨이가 필요하므로 추가 비용이 발생합니다.

- 특정 데이터 소스(Redshift, SQL Server, MySQL, Oracle, PostgreSQL)의 경우 AWS Glue Studio를 통해 통합 연결을 생성하려면 AWS STS 및 AWS Secrets Manager에 액세스해야 합니다. 이는 보안 연결을 설정하고 가상 프라이빗 클라우드(VPC) 내에서 이러한 데이터 소스에 액세스하는 데 필요한 자격 증명을 검색하는 데 필요합니다.
- AWS Glue Studio를 통해 통합 연결을 생성하려면 AWS Secrets Manager에 액세스하고 VPC 리소스를 관리할 수 있는 권한이 있는 IAM 역할이 필요합니다(VPC를 사용하는 경우).
  - secretsmanager:GetSecretValue
  - secretsmanager:PutSecretValue
  - secretsmanager:DescribeSecret
  - ec2:CreateNetworkInterface
  - ec2:DeleteNetworkInterface
  - ec2:DescribeNetworkInterfaces
  - ec2:DescribeSubnets

사용 가능한 연결

AWS Glue는 다음과 같은 연결 유형을 지원합니다.

- Adobe Analytics
- Adobe Marketo Engage
- Amazon Aurora(기본 JDBC 드라이버를 사용하는 경우 지원됨, 일부 드라이버 기능은 활용할 수 없음)
- Amazon DocumentDB
- Amazon DynamoDB
- Amazon OpenSearch Service, Spark용 AWS Glue를 사용하는 경우.
- Amazon Redshift
- Asana
- Azure Cosmos, AWS Glue ETL 작업과 함께 NoSQL용 Azure Cosmos DB를 사용하는 경우
- Azure SQL, Spark용 AWS Glue를 사용하는 경우.
- Blackbaud
- CircleCI
- Datadog

- Docusign Monitor
- Domo
- Dynatrace
- Facebook Ads
- Facebook Page Insights
- Freshdesk
- Freshsales
- Google Ads
- Google Analytics 4
- Google BigQuery, Spark용 AWS Glue를 사용하는 경우.
- Google Search Console
- Google Sheets
- HubSpot
- Instagram Ads
- Intercom
- JDBC
- Jira Cloud
- Kafka
- Kustomer
- LinkedIn
- Mailchimp
- Microsoft Teams
- Mixpanel
- 월요일
- MongoDB
- MongoDB Atlas
- Okta
- Oracle NetSuite
- Paypal

- Pendo
- Pipedrive
- Productboard
- QuickBooks
- Salesforce
- Salesforce Commerce Cloud
- Salesforce Marketing Cloud
- Salesforce Marketing Cloud Account Engagement(이전의 Salesforce Pardot)
- SAP HANA, Spark용 AWS Glue를 사용하는 경우.
- SAP OData
- SendGrid
- ServiceNow
- Slack
- Smartsheet
- Snapchat Ads
- Stripe
- Snowflake, Spark용 AWS Glue를 사용하는 경우.
- Teradata Vantage, Spark용 AWS Glue를 사용하는 경우.
- Twilio
- Vertica(Spark에 대해 AWS Glue를 사용하는 경우).
- WooCommerce
- Zendesk
- Zoho CRM
- Zoom Meetings
- 각종 Amazon Relational Database Service(RDS) 제안.
- 네트워크(Amazon Virtual Private Cloud(Amazon VPC)에 있는 데이터 소스에 대한 연결 지정)

AWS Glue Studio를 사용하면 커넥터에 대한 연결을 생성할 수도 있습니다. 커넥터는 AWS Glue Studio의 데이터 스토어에 액세스하는 데 도움이 되는 선택적 코드 패키지입니다. 자세한 내용은 AWS Glue Studio에서 커넥터 및 연결 사용을 참조하세요.

온프레미스 데이터베이스에 연결하는 방법에 대한 자세한 내용은 AWS Big Data Blog 웹 사이트의 How to access and analyze on-premises data stores using AWS Glue를 참조하세요.

사용자 지정 IAM 역할을 사용하는 동안 VPC 구성과의 연결을 생성하려면 다음 VPC 액세스 작업이 있어야 합니다.

- secretsmanager:GetSecretValue
- secretsmanager:PutSecretValue
- secretsmanager:DescribeSecret
- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSubnets

제한 사항

- AWS Glue API를 사용하여 v2 연결을 생성한 경우 AWS Glue 콘솔을 통해 연결을 편집할 수 없습니다.
  - Amazon DocumentDB
  - Amazon Aurora
  - MariaDB
  - MongoDB Atlas
  - MongoDB

AWS Glue 연결 속성

이 주제에는 AWS Glue 연결 속성에 대한 정보가 포함되어 있습니다.

주제

- 필수 연결 속성
- AWS Glue JDBC 연결 속성
- AWS Glue MongoDB 및 MongoDB Atlas 연결 속성
- Salesforce 연결 속성
- Snowflake 연결

- Vertica 연결
- SAP HANA 연결
- Azure SQL 연결
- Teradata Vantage 연결
- OpenSearch Service 연결
- Azure Cosmos 연결
- AWS Glue SSL 연결 속성
- 클라이언트 인증을 위한 Apache Kafka 연결 속성
- Google BigQuery 연결
- Vertica 연결

필수 연결 속성

AWS Glue 콘솔에서 연결을 정의할 때 다음 속성에 대한 값을 제공해야 합니다.

연결 이름

연결에 고유한 이름을 입력합니다.

연결 유형

JDBC 또는 특정 연결 유형 중 하나를 선택합니다.

JDBC 연결 유형에 대한 자세한 내용은 the section called “JDBC 연결 속성” 단원을 참조하십시오.

[네트워크(Network)]를 선택하여 Amazon Virtual Private Cloud 환경(Amazon VPC) 내의 데이터 원본에 연결합니다.

선택한 유형에 따라 AWS Glue 콘솔에 다른 필수 필드가 표시됩니다. 예를 들어 [Amazon RDS]를 선택한 경우 데이터베이스 엔진을 선택해야 합니다.

필요한 SSL 연결

이 옵션을 선택하면 AWS Glue에서 데이터 스토어 연결이 신뢰할 수 있는 보안 소켓 계층(SSL)을 통해 이루어졌는지 확인해야 합니다.

이 옵션을 선택할 때 사용할 수 있는 추가 옵션을 비롯한 자세한 내용은 the section called “SSL 연결 속성” 단원을 참조하십시오.

MSK 클러스터 선택(Amazon Managed Streaming for Apache Kafka(MSK)만 해당)

다른 AWS 계정의 MSK 클러스터를 지정합니다.

Kafka 부트스트랩 서버 URL(Kafka만 해당)

쉼표로 구분된 부트스트랩 서버 URL 목록을 지정합니다. 포트 번호를 포함합니다. 예: b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094, b-2.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094, b-3.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094

AWS Glue JDBC 연결 속성

AWS Glue Studio는 이제 MySQL, Oracle, PostgresSQL, Redshift, SQL Server 데이터 소스의 통합 연결을 생성합니다.이 연결에는 Secrets Manager 및 VPC 리소스에 액세스하기 위한 추가 단계가 필요하므로 추가 비용이 발생할 수 있습니다. 각 연결의 연결 이름을 선택하여 AWS Glue Studio에서 이러한 연결에 액세스할 수 있습니다.

자세한 내용은 고려 사항 단원을 참조하십시오.

AWS Glue는 JDBC 연결을 통해 다음 데이터 스토어에 연결할 수 있습니다.

- Amazon Redshift
- Amazon Aurora
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Snowflake(AWS Glue 크롤러를 사용하는 경우)
- Aurora(기본 JDBC 드라이버를 사용하는 경우 지원됨, 일부 드라이버 기능은 사용할 수 없음)
- Amazon RDS for MariaDB

> **Important**
>
> 현재 ETL 작업은 하나의 서브넷 내에서만 JDBC 연결을 사용할 수 있습니다. 작업에서 다양한 데이터 스토어가 있으면 모두 동일한 서브넷에 있거나 서브넷에서 액세스 가능해야 합니다. AWS Glue 크롤러에 대해 자체 JDBC 드라이버 버전을 가져오기로 선택한 경우 크롤러는 AWS Glue작업 및 Amazon S3의 리소스를 소비하여 제공된 드라이버가 사용자 환경에서 실행

되도록 합니다. 리소스의 추가 사용량은 계정에 반영됩니다. 또한 자체 JDBC 드라이버를 제공한다고 해서 크롤러가 해당 드라이버의 모든 기능을 사용할 수 있는 것은 아닙니다. 드라이버는 데이터 카탈로그의 연결 정의에 설명된 속성으로 제한됩니다.

다음은 JDBC 연결 유형에 대한 추가 속성입니다.

JDBC URL

JDBC 데이터 스토어의 URL을 입력합니다. 대부분 데이터베이스 엔진의 경우, 이 필드는 다음과 같은 포맷입니다. 이 포맷에서 *protocol*, *host*, *port*, *db_name*을 고유한 정보로 바꿉니다.

```
jdbc:protocol://host:port/db_name
```

데이터베이스 엔진에 다라 다른 JDBC URL 포맷이 필요합니다. 이 포맷은 클론(:), 슬래시(/), 및 기타 키워드를 다르게 사용하여 데이터베이스를 지정합니다.

JDBC를 데이터 스토어에 연결하는 데 필요한 데이터 스토어의 db_name입니다. db_name는 지원한 username 및 password를 통해 네트워크 연결을 만드는 데 사용됩니다. 연결되면 AWS Glue는 데이터 스토어의 다른 데이터베이스에 액세스할 수 있어 크롤러나 ETL 작업을 실행할 수 있습니다.

다음 JDBC URL 예제에서는 몇 가지 데이터베이스 엔진의 구문을 보여 줍니다.

- dev 데이터베이스로 Amazon Redshift 클러스터 데이터 스토어에 연결하려면

  ```
  jdbc:redshift://xxx.us-east-1.redshift.amazonaws.com:8192/dev
  ```

- employee 데이터베이스로 Amazon RDS for MySQL 데이터 스토어에 연결하려면

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.us-east-1.rds.amazonaws.com:3306/
  employee
  ```

- employee 데이터베이스로 Amazon RDS for PostgreSQL 데이터 스토어에 연결하려면

  ```
  jdbc:postgresql://xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:5432/employee
  ```

- employee 서비스 이름으로 Amazon RDS for Oracle 데이터 스토어에 연결하려면

  ```
  jdbc:oracle:thin://@xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:1521/employee
  ```

Amazon RDS for Oracle의 구문은 다음 패턴을 따를 수 있습니다. 이러한 패턴에서 *host*, *port*, *service_name*, *SID*를 고유한 정보로 바꿉니다.

- `jdbc:oracle:thin://@`*host*`:`*port*`/`*service_name*
- `jdbc:oracle:thin://@`*host*`:`*port*`:`*SID*

- employee 데이터베이스로 Amazon RDS for Microsoft SQL Server 데이터 스토어에 연결하려면

  ```
  jdbc:sqlserver://xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:1433;databaseName=employee
  ```

  Amazon RDS for SQL Server의 구문은 다음 패턴을 따를 수 있습니다. 이러한 패턴에서 *server_name*, *port* 및 *db_name*을 사용자 자신의 정보로 바꿉니다.

  - `jdbc:sqlserver://`*server_name*`:`*port*`;database=`*db_name*
  - `jdbc:sqlserver://`*server_name*`:`*port*`;databaseName=`*db_name*

- employee 데이터베이스의 Amazon Aurora PostgreSQL 인스턴스에 연결하려면 데이터베이스 인스턴스의 엔드포인트, 포트 및 데이터베이스 이름을 지정합니다.

  ```
  jdbc:postgresql://employee_instance_1.xxxxxxxxxxxx.us-
  east-2.rds.amazonaws.com:5432/employee
  ```

- employee 데이터베이스가 포함된 Amazon RDS for MariaDB 데이터 스토어에 연결하려면 데이터베이스 인스턴스의 엔드포인트, 포트, 데이터베이스 이름을 지정합니다.

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.aws-
  region.rds.amazonaws.com:3306/employee
  ```

- > **Warning**
  >
  > Snowflake JDBC 연결은 AWS Glue 크롤러에서만 지원됩니다. AWS Glue 작업에서 Snowflake 커넥터를 사용할 때는 Snowflake 연결 유형을 사용합니다.

  sample 데이터베이스의 Snowflake 인스턴스에 연결하려면 Snowflake 인스턴스의 엔드포인트, 사용자, 데이터베이스 이름 및 역할 이름을 지정합니다. warehouse 파라미터를 선택적으로 추가할 수 있습니다.

  ```
  jdbc:snowflake://account_name.snowflakecomputing.com/?
  user=user_name&db=sample&role=role_name&warehouse=warehouse_name
  ```

> **Important**
>
> JDBC를 통한 Snowflake 연결의 경우 URL의 파라미터 순서가 적용되므로, `user`, `db`, `role_name`, `warehouse`의 순서로 정렬되어야 합니다.

- AWS 프라이빗 링크를 사용하여 `sample` 데이터베이스의 Snowflake 인스턴스에 연결하려면 다음과 같이 Snowflake JDBC URL을 지정합니다.

  ```
  jdbc:snowflake://account_name.region.privatelink.snowflakecomputing.com/?user=user_name&db=sample&role=role_name&warehouse=warehouse_name
  ```

사용자 이름

> **Note**
>
> 사용자 이름과 암호를 직접 제공하는 대신 AWS 보안 암호를 사용하여 연결 자격 증명을 저장하는 것이 좋습니다. 자세한 내용은 에 연결 보안 인증 정보 저장 AWS Secrets Manager 단원을 참조하십시오.

JDBC 데이터 스토어 연결 권한을 갖는 사용자 이름을 제공합니다.

암호

JDBC 데이터 스토어에 대한 액세스 권한이 있는 사용자 이름의 비밀번호를 입력합니다.

Port

Amazon RDS Oracle 인스턴스에 연결하기 위해 JDBC URL에 사용된 포트를 입력합니다. 이 필드는 Amazon RDS Oracle 인스턴스에 대해 [SSL 연결 필요(Require SSL connection)]가 선택된 경우에만 표시됩니다.

VPC

데이터 스토어를 포함한 Virtual Private Cloud(VPC) 이름을 선택합니다. AWS Glue 콘솔은 현재 리전의 모든 VPC를 나열합니다.

> **Important**
>
> Snowflake의 데이터와 같이 AWS 외부에서 호스팅되는 JDBC 연결을 통해 작업하는 경우 VPC에 트래픽을 퍼블릭 서브넷과 프라이빗 서브넷으로 분할하는 NAT 게이트웨이가 있어

야 합니다. 퍼블릭 서브넷은 외부 소스에 연결하는 데 사용되고 내부 서브넷은 AWS Glue에서 처리하는 데 사용됩니다. 외부 연결을 위해 Amazon VPC를 구성하는 방법에 대한 자세한 내용은 NAT 디바이스를 사용하여 인터넷 또는 다른 네트워크에 연결 및 AWS Glue에서 Amazon RDS 데이터 스토어에 대해 JDBC를 연결하도록 Amazon VPC 설정 섹션을 참조하세요.

서브넷

데이터 스토어를 포함하는 VPC 내 서브넷을 선택합니다. AWS Glue 콘솔은 VPC에 데이터 스토어에 대한 모든 서브넷을 나열합니다.

보안 그룹

데이터 스토어와 관련된 보안 그룹을 선택합니다. AWS Glue에는 연결할 AWS Glue를 허용하는 인바운드 소스 규칙과 함께 하나 이상의 보안 그룹이 필요합니다. AWS Glue 콘솔은 VPC에 인바운드 액세스를 보장하는 보안 그룹을 나열합니다. AWS Glue는 VPC 서브넷에 연결된 탄력적 네트워크 인터페이스를 통해 보안 그룹이 연결됩니다.

JDBC 드라이버 클래스 이름 - 선택 사항

사용자 지정 JDBC 드라이버 클래스 이름을 제공합니다.

- Postgres – org.postgresql.Driver
- MySQL – com.mysql.jdbc.Driver, com.mysql.cj.jdbc.Driver
- Redshift – com.amazon.redshift.jdbc.Driver, com.amazon.redshift.jdbc42.Driver
- Oracle – oracle.jdbc.driver.OracleDriver
- SQL Server – com.microsoft.sqlserver.jdbc.SQLServerDriver

JDBC 드라이버 S3 경로 - 선택 사항

사용자 지정 JDBC 드라이버에 Amazon S3 위치를 제공합니다. 이는 .jar 파일의 절대 경로입니다. 크롤러 지원 데이터베이스에서 데이터 소스에 연결하기 위해 자체 JDBC 드라이버를 제공하려는 경우 `customJdbcDriverS3Path` 및 `customJdbcDriverClassName` 파라미터에 대한 값을 지정할 수 있습니다.

고객이 제공한 JDBC 드라이버의 사용은 필요한 필수 연결 속성으로만 제한됩니다.

AWS Glue MongoDB 및 MongoDB Atlas 연결 속성

다음은 MongoDB 또는 MongoDB Atlas 연결 유형에 대한 추가 속성입니다.

MongoDB URL

MongoDB 또는 MongoDB Atlas 데이터 스토어의 URL을 입력합니다.

- MongoDB의 경우: mongodb://host:port/database. 호스트는 호스트 이름, IP 주소 또는 UNIX 도메인 소켓일 수 있습니다. 연결 문자열이 포트를 지정하지 않는 경우 기본 MongoDB 포트인 27017을 사용합니다.
- MongoDB Atlas의 경우: mongodb+srv://server.example.com/database. 호스트는 DNS SRV 레코드에 해당하는 호스트 이름일 수 있습니다. SRV 형식에는 포트가 필요하지 않으며 기본 MongoDB 포트인 27017을 사용합니다.

사용자 이름

> **Note**
>
> 사용자 이름과 암호를 직접 제공하는 대신 AWS 보안 암호를 사용하여 연결 자격 증명을 저장하는 것이 좋습니다. 자세한 내용은 에 연결 보안 인증 정보 저장 AWS Secrets Manager 단원을 참조하십시오.

JDBC 데이터 스토어 연결 권한을 갖는 사용자 이름을 제공합니다.

암호

MongoDB 또는 MongoDB Atlas 데이터 스토어에 대한 액세스 권한이 있는 사용자 이름에 대한 비밀번호를 입력합니다.

Salesforce 연결 속성

다음은 Salesforce 연결 유형에 대한 추가 속성입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기/쓰기에 사용됩니다. Salesforce에서의 객체 이름입니다.
- `API_VERSION`(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용하려는 Salesforce Rest API 버전.

- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- IMPORT_DELETED_RECORDS(문자열) - 기본값: FALSE. 읽기에 사용됩니다. 쿼리하는 동안 삭제 레코드를 가져오려면 다음을 수행합니다.
- WRITE_OPERATION(문자열) - 기본값: INSERT. 쓰기에 사용됩니다. 값은 INSERT, UPDATE, UPSERT, DELETE여야 합니다.
- ID_FIELD_NAMES(문자열) - 기본값: null. UPSERT에만 사용됩니다.

Snowflake 연결

다음 속성은 AWS Glue ETL 작업에 사용되는 Snowflake 연결을 설정하는 데 사용됩니다. Snowflake를 크롤링할 때는 JDBC 연결을 사용합니다.

Snowflake URL

Snowflake 엔드포인트의 URL입니다. Snowflake 엔드포인트 URL에 대한 자세한 내용은 Snowflake 설명서의 Connecting to Your Accounts를 참조하세요.

AWS 보안 암호

AWS Secrets Manager에서 보안 암호의 보안 암호 이름입니다. AWS Glue에서는 보안 암호의 `sfUser` 및 `sfPassword` 키를 사용하여 Snowflake에 연결합니다.

Snowflake 역할(선택 사항)

연결할 때 AWS Glue에서 사용하는 Snowflake 보안 역할입니다.

AWS PrivateLink를 사용하여 Amazon VPC에 호스팅되는 Snowflake 엔드포인트에 대한 연결을 구성할 때 다음 속성을 사용합니다.

VPC

데이터 스토어를 포함한 Virtual Private Cloud(VPC) 이름을 선택합니다. AWS Glue 콘솔은 현재 리전의 모든 VPC를 나열합니다.

서브넷

데이터 스토어를 포함하는 VPC 내 서브넷을 선택합니다. AWS Glue 콘솔은 VPC에 데이터 스토어에 대한 모든 서브넷을 나열합니다.

보안 그룹

데이터 스토어와 관련된 보안 그룹을 선택합니다. AWS Glue에는 연결할 AWS Glue를 허용하는 인바운드 소스 규칙과 함께 하나 이상의 보안 그룹이 필요합니다. AWS Glue 콘솔은 VPC에 인바운드 액세스를 보장하는 보안 그룹을 나열합니다. AWS Glue는 VPC 서브넷에 연결된 탄력적 네트워크 인터페이스를 통해 보안 그룹이 연결됩니다.

Vertica 연결

다음 속성을 사용하여 AWS Glue ETL 작업을 위한 Vertica 연결을 설정합니다.

Vertica 호스트

Vertica 설치의 호스트 이름.

Vertica 포트

Vertica 설치를 사용할 수 있는 포트입니다.

AWS 보안 암호

AWS Secrets Manager에서 보안 암호의 보안 암호 이름입니다. AWS Glue에서는 보안 암호의 키를 사용하여 Vertica에 연결합니다.

Amazon VPC에 호스팅되는 Vertica 엔드포인트에 대한 연결을 구성할 때 다음 속성을 사용합니다.

VPC

데이터 스토어를 포함한 Virtual Private Cloud(VPC) 이름을 선택합니다. AWS Glue 콘솔은 현재 리전의 모든 VPC를 나열합니다.

서브넷

데이터 스토어를 포함하는 VPC 내 서브넷을 선택합니다. AWS Glue 콘솔은 VPC에 데이터 스토어에 대한 모든 서브넷을 나열합니다.

보안 그룹

데이터 스토어와 관련된 보안 그룹을 선택합니다. AWS Glue에는 연결할 AWS Glue를 허용하는 인바운드 소스 규칙과 함께 하나 이상의 보안 그룹이 필요합니다. AWS Glue 콘솔은 VPC에 인바운드 액세스를 보장하는 보안 그룹을 나열합니다. AWS Glue는 VPC 서브넷에 연결된 탄력적 네트워크 인터페이스를 통해 보안 그룹이 연결됩니다.

SAP HANA 연결

다음 속성을 사용하여 AWS Glue ETL 작업을 위한 SAP HANA 연결을 설정합니다.

SAP HANA URL

SAP JDBC URL.

SAP HANA JDBC URL은 `jdbc:sap://saphanaHostname:saphanaPort/?databaseName=saphanaDBname,ParameterNa` 형식입니다

AWS Glue는 JDBC URL 매개 변수가 필요합니다.

- `databaseName` - 연결할 SAP HANA의 기본 데이터베이스입니다.

AWS 보안 암호

AWS Secrets Manager에서 보안 암호의 보안 암호 이름입니다. AWS Glue에서는 보안 암호의 키를 사용하여 SAP HANA에 연결합니다.

를 사용하여 Amazon VPC에 호스팅되는 SAP HANA 엔드포인트에 대한 연결을 구성할 때 다음 속성을 사용합니다.

VPC

데이터 스토어를 포함한 Virtual Private Cloud(VPC) 이름을 선택합니다. AWS Glue 콘솔은 현재 리전의 모든 VPC를 나열합니다.

서브넷

데이터 스토어를 포함하는 VPC 내 서브넷을 선택합니다. AWS Glue 콘솔은 VPC에 데이터 스토어에 대한 모든 서브넷을 나열합니다.

보안 그룹

데이터 스토어와 관련된 보안 그룹을 선택합니다. AWS Glue에는 연결할 AWS Glue를 허용하는 인바운드 소스 규칙과 함께 하나 이상의 보안 그룹이 필요합니다. AWS Glue 콘솔은 VPC에 인바운드 액세스를 보장하는 보안 그룹을 나열합니다. AWS Glue는 VPC 서브넷에 연결된 탄력적 네트워크 인터페이스를 통해 보안 그룹이 연결됩니다.

Azure SQL 연결

다음 속성을 사용하여 AWS Glue ETL 작업에 대한 Azure SQL 연결을 설정합니다.

Azure SQL URL

Azure SQL 엔드포인트의 JDBC URL입니다.

목록은 `jdbc:sqlserver://`*`databaseServerName`*`:`*`databasePort`*`;databaseName=`*`azuresqlDBname`*`;` 형식이어야 합니다.

AWS Glue는 다음 URL 속성이 필요합니다.

- `databaseName` - 연결할 Azure SQL의 기본 데이터베이스입니다.

Azure SQL 관리형 인스턴스용 JDBC URL에 대한 자세한 내용은 Microsoft 설명서를 참조하십시오.

AWS 보안 암호

AWS Secrets Manager에서 보안 암호의 보안 암호 이름입니다. AWS Glue에서는 보안 암호의 키를 사용하여 Azure SQL에 연결합니다.

Teradata Vantage 연결

다음 속성을 사용하여 AWS Glue ETL 작업을 위한 Teradata Vantage 연결을 설정합니다.

Teradata URL

Teradata 인스턴스에 연결하려면 데이터베이스 인스턴스의 호스트 이름과 관련 Teradata 매개변수를 지정합니다.

```
jdbc:teradata://teradataHostname/ParameterName=ParameterValue,ParameterName=Pa
```

AWS Glue에서는 다음 JDBC URL 파라미터를 지원합니다.

- `DATABASE_NAME` - 연결할 Teradata의 기본 데이터베이스입니다.
- `DBS_PORT` - Teradata 포트(비표준인 경우)를 지정합니다.

AWS 보안 암호

AWS Secrets Manager에서 보안 암호의 보안 암호 이름입니다. AWS Glue에서는 보안 암호의 키를 사용하여 Teradata Vantage에 연결합니다.

Amazon VPC에 호스팅되는 Teradata Vantage 엔드포인트에 대한 연결을 구성할 때 다음 속성을 사용합니다.

VPC

데이터 스토어를 포함한 Virtual Private Cloud(VPC) 이름을 선택합니다. AWS Glue 콘솔은 현재 리전의 모든 VPC를 나열합니다.

서브넷

데이터 스토어를 포함하는 VPC 내 서브넷을 선택합니다. AWS Glue 콘솔은 VPC에 데이터 스토어에 대한 모든 서브넷을 나열합니다.

보안 그룹

데이터 스토어와 관련된 보안 그룹을 선택합니다. AWS Glue에는 연결할 AWS Glue를 허용하는 인바운드 소스 규칙과 함께 하나 이상의 보안 그룹이 필요합니다. AWS Glue 콘솔은 VPC에 인바운드 액세스를 보장하는 보안 그룹을 나열합니다. AWS Glue는 VPC 서브넷에 연결된 탄력적 네트워크 인터페이스를 통해 보안 그룹이 연결됩니다.

OpenSearch Service 연결

다음 속성을 사용하여 AWS Glue ETL 작업에 대한 OpenSearch Service 연결을 설정합니다.

도메인 엔드포인트

Amazon OpenSearch Service 도메인 엔드포인트는 기본 형식이 https://search-*domainName-unstructuredIdContent*.*region*.es.amazonaws.com입니다. 도메인 엔드포인트에 관해 자세한 내용을 알아보려면 Amazon OpenSearch Service 설명서의 Amazon OpenSearch Service 도메인 생성 및 관리를 참조하십시오.

Port

엔드포인트에서 포트가 열립니다.

AWS 보안 암호

AWS Secrets Manager에서 보안 암호의 보안 암호 이름입니다. AWS Glue에서는 보안 암호의 키를 사용하여 OpenSearch Service에 연결합니다.

Amazon VPC에 호스팅되는 OpenSearch Service 엔드포인트에 대한 연결을 구성할 때 다음 속성을 사용합니다.

VPC

데이터 스토어를 포함한 Virtual Private Cloud(VPC) 이름을 선택합니다. AWS Glue 콘솔은 현재 리전의 모든 VPC를 나열합니다.

서브넷

데이터 스토어를 포함하는 VPC 내 서브넷을 선택합니다. AWS Glue 콘솔은 VPC에 데이터 스토어에 대한 모든 서브넷을 나열합니다.

보안 그룹

데이터 스토어와 관련된 보안 그룹을 선택합니다. AWS Glue에는 연결할 AWS Glue를 허용하는 인바운드 소스 규칙과 함께 하나 이상의 보안 그룹이 필요합니다. AWS Glue 콘솔은 VPC에 인바운드 액세스를 보장하는 보안 그룹을 나열합니다. AWS Glue는 VPC 서브넷에 연결된 탄력적 네트워크 인터페이스를 통해 보안 그룹이 연결됩니다.

Azure Cosmos 연결

다음 속성을 사용하여 AWS Glue ETL 작업을 위한 Azure Cosmos 연결을 설정합니다.

Azure Cosmos DB 계정 엔드포인트 URI

Amazon Cosmos에 연결하는 데 사용된 엔드포인트. 자세한 내용은 Azure 설명서를 참조하십시오.

AWS 보안 암호

AWS Secrets Manager에서 보안 암호의 보안 암호 이름입니다. AWS Glue에서는 보안 암호의 키를 사용하여 Azure Cosmos에 연결합니다.

AWS Glue SSL 연결 속성

다음은 Require SSL connection(SSL 연결 필요) 속성에 대한 세부 정보입니다.

SSL 연결이 필요하지 않은 경우 AWS Glue에서 SSL을 사용하여 데이터 스토어에 대한 연결을 암호화할 때 실패를 무시합니다. 구성 지침은 데이터 스토어의 문서를 참조하세요. 이 옵션을 선택하면 AWS Glue에서 연결할 수 없을 때 개발 엔드포인트의 작업 실행, 크롤러 또는 ETL 문이 실패합니다.

> **Note**
>
> Snowflake는 기본적으로 SSL 연결을 지원하므로 이 속성은 Snowflake에 적용되지 않습니다.

이 옵션은 AWS Glue 클라이언트 측에서 검증됩니다. JDBC 연결의 경우 AWS Glue는 인증서 및 호스트 이름 검증을 사용하여 SSL을 통해서만 연결합니다. SSL 연결 지원은 다음에 사용할 수 있습니다.

- Oracle Database
- Microsoft SQL Server
- PostgreSQL
- Amazon Redshift
- MySQL(Amazon RDS 인스턴스만 해당)
- Amazon Aurora MySQL(Amazon RDS 인스턴스만 해당)
- Amazon Aurora PostgreSQL(Amazon RDS 인스턴스만 해당)
- Amazon Managed Streaming for Apache Kafka를 포함하는 Kafka
- MongoDB

> **Note**
>
> Amazon RDS Oracle 데이터 스토어에서 [SSL 연결 필요(Require SSL connection)]를 사용할 수 있도록 하려면 옵션 그룹을 생성하여 Oracle 인스턴스에 연결해야 합니다.

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. [옵션 그룹(Option group)]을 Amazon RDS Oracle 인스턴스에 추가합니다. Amazon RDS 콘솔에서 옵션 그룹을 추가하는 방법에 대한 자세한 내용은 옵션 그룹 생성을 참조하세요.
3. 옵션을 SSL에 대한 옵션 그룹에 추가합니다. SSL에 대해 지정하는 [포트(Port)]는 나중에 Amazon RDS Oracle 인스턴스에 대한 AWS Glue JDBC 연결 URL을 생성할 때 사용됩니다. Amazon RDS 콘솔에서 옵션을 추가하는 방법에 대한 자세한 내용은 Amazon RDS User Guide의 Adding an Option to an Option Group을 참조하세요. Oracle SSL 옵션에 대한 자세한 내용은 Amazon RDS User Guide의 Oracle SSL을 참조하세요.
4. AWS Glue 콘솔에서 Amazon RDS Oracle 인스턴스에 대한 연결을 생성합니다. 연결 정의에서 SSL 연결 필요를 선택합니다. 요청 시 Amazon RDS Oracle SSL 옵션에서 사용한 [포트(Port)]를 입력합니다.

연결에 대해 [SSL 연결 필요(Require SSL connection)]를 선택한 경우 다음과 같은 추가 선택적 속성을 사용할 수 있습니다.

S3의 사용자 지정 JDBC 인증서

현재 온프레미스 또는 클라우드 데이터베이스와의 SSL 통신에 사용 중인 인증서가 있는 경우 AWS Glue 데이터 원본 또는 대상에 대한 SSL 연결에 해당 인증서를 사용할 수 있습니다. 사용자 정의 루트 인증서가 포함된 Amazon Simple Storage Service(Amazon S3) 위치를 입력합니다. AWS Glue는 이 인증서를 사용하여 데이터베이스에 대한 SSL 연결을 설정합니다. AWS Glue는 X.509 인증서만 처리합니다. 인증서는 DER로 인코딩되고 base64 인코딩 PEM 형식으로 제공되어야 합니다.

이 필드를 비워두면 기본 인증서가 사용됩니다.

사용자 지정 JDBC 인증서 문자열

JDBC 데이터베이스에 특정한 인증서 정보를 입력합니다. 이 문자열은 도메인 일치 또는 고유 이름(DN) 일치에 사용됩니다. Oracle Database의 경우 이 문자열은 `tnsnames.ora` 파일의 보안 섹션에 있는 `SSL_SERVER_CERT_DN` 파라미터에 매핑됩니다. Microsoft SQL Server의 경우 이 문자열은 `hostNameInCertificate`로 사용됩니다.

다음은 Oracle Database `SSL_SERVER_CERT_DN` 파라미터의 예입니다.

```
cn=sales,cn=OracleContext,dc=us,dc=example,dc=com
```

Kafka 프라이빗 CA 인증서 위치

Kafka 데이터 스토어와의 SSL 통신에 현재 사용 중인 인증서가 있는 경우 해당 인증서를 AWS Glue 연결과 함께 사용할 수 있습니다. 이 옵션은 Kafka 데이터 스토어의 경우 필수이고 Amazon Managed Streaming for Apache Kafka 데이터 스토어의 경우 선택 사항입니다. 사용자 정의 루트 인증서가 포함된 Amazon Simple Storage Service(Amazon S3) 위치를 입력합니다. AWS Glue는 이 인증서를 사용하여 Kafka 데이터 스토어에 대한 SSL 연결을 설정합니다. AWS Glue는 X.509 인증서만 처리합니다. 인증서는 DER로 인코딩되고 base64 인코딩 PEM 형식으로 제공되어야 합니다.

인증서 검증 건너뛰기

AWS Glue의 사용자 정의 인증서 검증을 건너뛰려면 [인증서 검증 건너뛰기(Skip certificate validation)] 확인란을 선택합니다. 검증하기로 한 경우, AWS Glue가 인증서의 서명 알고리즘 및 주제 퍼블릭 키 알고리즘을 검증합니다. 인증서가 검증에 실패하면, 연결을 사용하는 모든 ETL 작업 또는 크롤러가 실패합니다.

허용되는 유일한 서명 알고리즘은 SHA256withRSA, SHA384withRSA 또는 SHA512withRSA. 주제 퍼블릭 키 알고리즘의 경우, 키 길이는 2048 이상이어야 합니다.

Kafka 클라이언트 키 스토어 위치

Kafka 클라이언트 측 인증을 위한 클라이언트 키 스토어 파일의 Amazon S3 위치입니다. 경로는 s3://bucket/prefix/filename.jks 형식이어야 합니다. 파일 이름과 .jks 확장자로 끝나야 합니다.

Kafka 클라이언트 키 스토어 암호(선택 사항)

제공된 키 스토어에 액세스하기 위한 암호입니다.

Kafka 클라이언트 키 암호(선택 사항)

키 스토어는 여러 키로 구성 될 수 있으므로 Kafka 서버 측 키와 함께 사용할 클라이언트 키에 액세스하기 위한 암호입니다.

클라이언트 인증을 위한 Apache Kafka 연결 속성

AWS Glue는 Apache Kafka 연결을 생성할 때 인증을 위한 SASL(Simple Authentication and Security Layer) 프레임워크를 지원합니다. SASL 프레임워크는 다양한 인증 메커니즘을 지원하며, AWS Glue는 SCRAM(사용자 이름 및 암호), GSSAPI(Kerberos 프로토콜), PLAIN 프로토콜을 제공합니다.

AWS Glue Studio를 사용하여 다음 클라이언트 인증 방법 중 하나를 구성합니다. 자세한 내용은 AWS Glue Studio 사용 안내서의 커넥터에 대한 연결 생성을 참조하세요.

- 없음(None) - 인증이 없습니다. 테스트 목적으로 연결을 생성하는 경우에 유용합니다.
- SASL/SCRAM-SHA-512 - 이 인증 방법을 선택하면 인증 자격 증명을 지정할 수 있습니다. 두 가지 옵션을 사용할 수 있습니다.
  - AWS Secrets Manager 사용(권장) - 이 옵션을 선택하면 AWS Secrets Manager에 사용자 이름과 암호를 저장하여 필요할 때 AWS Glue에서 액세스하도록 할 수 있습니다. SSL 또는 SASL 인증 자격 증명을 저장하는 보안 암호를 지정합니다. 자세한 내용은 에 연결 보안 인증 정보 저장 AWS Secrets Manager 단원을 참조하십시오.
  - 사용자 이름과 암호를 직접 제공합니다.
- SASL/GSSAPI (Kerberos)(SASL/GSSAPI(Kerberos)) - 이 옵션을 선택하면 keytab 파일, krb5.conf 파일의 위치를 선택하고 Kerberos 보안 주체 이름과 Kerberos 서비스 이름을 입력할 수 있습니다. keytab 파일과 krb5.conf 파일의 위치는 Amazon S3 위치에 있어야 합니다. MSK는 아직 SASL/GSSAPI를 지원하지 않으므로 이 옵션은 고객 관리형 Apache Kafka 클러스터에만 사용할 수 있습니다. 자세한 내용은 MIT Kerberos Documentation: Keytab(MIT Kerberos 설명서: Keytab)을 참조하세요.
- SASL/PLAIN - 인증 자격 증명을 지정하려면 이 인증 방법을 선택합니다. 두 가지 옵션을 사용할 수 있습니다.
  - AWS Secrets Manager 사용(권장) - 이 옵션을 선택하면 AWS Secrets Manager에 보안 인증 정보를 저장하여 필요할 때 AWS Glue에서 정보에 액세스하도록 할 수 있습니다. SSL 또는 SASL 인증 자격 증명을 저장하는 보안 암호를 지정합니다.
  - 사용자 이름 및 암호를 직접 제공합니다.
- SSL 클라이언트 인증(SSL Client Authentication) - 이 옵션을 선택하면 Amazon S3를 검색하여 Kafka 클라이언트 키 스토어의 위치를 선택할 수 있습니다. 선택 사항으로 Kafka 클라이언트 키 스토어 암호와 Kafka 클라이언트 키 암호를 입력할 수 있습니다.

Google BigQuery 연결

다음 속성은 AWS Glue ETL 작업에 사용되는 Google BigQuery 연결을 설정하는 데 사용됩니다. 자세한 내용은 the section called “BigQuery 연결” 단원을 참조하십시오.

AWS 보안 암호

AWS Secrets Manager 보안 암호의 보안 암호 이름. AWS Glue ETL 작업은 귀하의 보안 암호인 `credentials` 키를 사용하여 Google BigQuery에 연결됩니다.

Vertica 연결

다음 속성은 AWS Glue ETL 작업에 사용되는 Vertica 연결을 설정하는 데 사용됩니다. 자세한 내용은 the section called “수직 연결” 단원을 참조하십시오.

에 연결 보안 인증 정보 저장 AWS Secrets Manager

AWS Secrets Manager 를 사용하여 데이터 스토어에 대한 연결 자격 증명을 제공하는 것이 좋습니다. 이러한 방식으로 Secrets Manager를 사용하면 ETL 작업 및 크롤러 실행을 위해 런타임 시 보안 암호에 AWS Glue 액세스할 수 있으며 보안 인증을 안전하게 유지할 수 있습니다.

사전 조건

에서 Secrets Manager를 사용하려면 IAM 역할에 AWS Glue 보안 암호 값을 검색할 수 있는 권한을 부여 AWS Glue해야 합니다. AWS 관리형 정책에는 AWS Secrets Manager 권한이 포함되지 `AWSGlueServiceRole` 않습니다. 예제 IAM 정책은 AWS Secrets Manager 사용 설명서의 예제: 보안 암호 값을 검색할 수 있는 권한을 참조하세요.

네트워크 설정에 따라 VPC 엔드포인트를 생성하여 VPC 와 Secrets Manager 간에 프라이빗 연결을 설정해야 할 수도 있습니다. 자세한 내용은 VPC 엔드포인트 사용을 참조하세요 AWS Secrets Manager.

에 대한 보안 암호를 생성하려면 AWS Glue

1. AWS Secrets Manager 사용 설명서의 보안 암호 생성 및 관리의 지침을 따르세요. 다음 예제에서는 에 대한 보안 암호를 생성할 때 일반 텍스트 탭에서 보안 인증을 지정하는 방법을 JSON 보여줍니다 AWS Glue.

   ```
   {
     "username": "EXAMPLE-USERNAME",
     "password": "EXAMPLE-PASSWORD"
   }
   ```

2. AWS Glue Studio 인터페이스를 사용하여 보안 암호를 연결에 연결합니다. 자세한 지침은 AWS Glue Studio 사용 설명서의 커넥터에 대한 연결 생성을 참조하세요.

AWS Glue 연결 추가

Spark에 대한 AWS Glue를 프로그래밍 방식으로 데이터 소스에 연결할 수 있습니다. 자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.

AWS Glue 콘솔을 사용하여 연결을 추가, 편집, 삭제 및 테스트할 수도 있습니다. AWS Glue 연결에 대한 자세한 내용은 데이터에 연결 단원을 참조하십시오.

주제

- Adobe Analytics에 연결
- Adobe Marketo Engage에 연결
- AWS Glue Studio에서 Amazon Redshift에 연결
- Asana에 연결
- AWS Glue Studio에서 Azure Cosmos DB에 연결
- AWS Glue Studio에서 Azure SQL에 연결
- Blackbaud Raiser's Edge NXT에 연결
- CircleCI에 연결
- Datadog에 연결
- Docusign Monitor에 연결
- Domo에 연결
- Dynatrace에 연결
- Facebook Ads에 연결
- Facebook Page Insights에 연결
- Freshdesk에 연결
- Freshsales에 연결
- Google Ads에 연결
- Google Analytics에 연결 4
- AWS Glue Studio에서 Google BigQuery에 연결
- Google Search Console에 연결
- Google Sheets에 연결
- 에 연결 HubSpot

- Instagram Ads에 연결
- AWS Glue Studio에서 Intercom에 연결
- Jira Cloud에 연결
- Kustomer에 연결
- LinkedIn에 연결
- Mailchimp에 연결
- Microsoft Teams에 연결
- Mixpanel에 연결
- Monday에 연결
- AWS Glue Studio에서 MongoDB에 연결
- Oracle에 연결 NetSuite
- AWS Glue Studio에서 OpenSearch Service에 연결
- Okta에 연결
- PayPal에 연결
- Pendo에 연결
- Pipedrive에 연결
- Productboard에 연결
- QuickBooks에 연결
- Salesforce에 연결
- Salesforce Marketing Cloud에 연결
- Salesforce Commerce Cloud에 연결
- Salesforce Marketing Cloud Account Engagement에 연결
- AWS Glue Studio에서 SAP HANA에 연결
- SAP OData에 연결
- SendGrid에 연결
- 에 연결 ServiceNow
- AWS Glue Studio에서 Slack에 연결
- Smartsheet에 연결
- AWS Glue Studio에서 Snapchat Ads에 연결

- AWS Glue Studio에서 Snowflake에 연결
- AWS Glue Studio에서 Stripe에 연결
- AWS Glue Studio의 Teradata 밴티지에 연결
- Twilio에 연결
- AWS Glue Studio의 Vertica 입력에 연결
- WooCommerce에 연결
- Zendesk에 연결
- Zoho CRM에 연결
- Zoom Meetings에 연결
- 자체 JDBC 드라이버를 사용하여 JDBC 연결 추가

Adobe Analytics에 연결

Adobe Analytics는 고객 여정을 지원하는 다중 채널 디지털 경험에서 데이터를 수집하고 데이터 분석을 위한 도구를 제공하는 강력한 데이터 분석 플랫폼입니다. 일반적으로 마케터와 비즈니스 분석가가 비즈니스 분석 목적으로 사용하는 플랫폼입니다. Adobe Analytics 사용자인 경우 Adobe Analytics 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, Adobe Analytics을 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Adobe Analytics 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 Adobe Analytics 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Adobe Analytics 구성
- Adobe Analytics 연결 구성
- Adobe Analytics 엔터티에서 읽기
- Adobe Analytics 연결 옵션
- Adobe Analytics 계정 생성
- 제한 사항

AWS Glue의 Adobe Analytics 지원

AWS Glue에서는 다음과 같이 Adobe Analytics를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Adobe Analytics에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Adobe Analytics API 버전

v2.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess-정책이 연결된 자격 증명이 AWS Management 콘솔을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Adobe Analytics 구성

AWS Glue를 사용하여 Adobe Analytics에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호를 사용하는 Adobe Analytics 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Adobe Analytics 계정 생성을 참조하세요.
- Adobe Analytics 계정이 API 액세스에 대해 활성화되어 있습니다. API 액세스는 Select, Prime 및 Ultimate 에디션에서 기본적으로 활성화되어 있습니다.

이러한 요구 사항을 충족하면 Adobe Analytics 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Adobe Analytics에서 다른 작업을 수행하지 않아도 됩니다.

Adobe Analytics 연결 구성

Adobe Analytics는 `OAuth2`에 대한 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' `OAuth`로 간주됩니다. 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Adobe Analytics에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Adobe Analytics로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

AUTHORIZATION_CODE OAuth 흐름용 연결 앱을 생성하는 방법에 대한 퍼블릭 Adobe Analytics 설명서는 Adobe Analytics API를 참조하세요.

Adobe Analytics 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.

고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

> **Note**
>
> AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 연결 유형을 선택할 때 Adobe Analytics를 선택합니다.
   b. 연결하려는 Adobe Analytics의 `x_api_key, instanceUrl`을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
    "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "secretsmanager:DescribeSecret",
                    "secretsmanager:GetSecretValue",
                    "secretsmanager:PutSecretValue",
                    "ec2:CreateNetworkInterface",
                    "ec2:DescribeNetworkInterface",
                    "ec2:DeleteNetworkInterface",
                ],
                "Resource": "*"
            }
        ]
    }
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Adobe Analytics 엔터티에서 읽기

사전 조건

읽으려는 Adobe Analytics 객체입니다. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Annotation	예	예	예	예	아니요
Calculated Metrics	예	예	예	예	아니요
Calculated Metrics Function	예	아니요	아니요	예	아니요
Component Metadata Shares	예	예	아니요	예	아니요
Date Ranges	예	예	아니요	예	아니요
Dimensions	예	아니요	아니요	예	아니요
Metrics	예	아니요	아니요	예	아니요
Projects	예	아니요	아니요	예	아니요
Reports Top Item	예	예	아니요	예	아니요
Segments	예	예	예	예	아니요
Usage Logs	예	예	아니요	예	아니요

예

```
adobeAnalytics_read = glueContext.create_dynamic_frame.from_options(
    connection_type="adobeanalytics",
```

```
    connection_options={
       "connectionName": "connectionName",
       "ENTITY_NAME": "annotation/ex*****",
       "API_VERSION": "v2.0"
})
```

Adobe Analytics 엔터티 및 필드 세부 정보

- Annotations
- Calculated Metrics
- Component Meta Data
- Date Ranges
- Dimensions
- Metrics
- Projects
- Reports
- Segments
- Users
- Usage Logs

Adobe Analytics 연결 옵션

다음은 Adobe Analytics의 연결 옵션입니다.

- `ENTITY_NAME`(문자열)-(필수) 읽기/쓰기에 사용됩니다. Adobe Analytics에서의 객체 이름입니다.
- `API_VERSION`(문자열)-(필수) 읽기/쓰기에 사용됩니다. 사용하려는 Adobe Analytics Rest API 버전입니다. 예: v2.0.
- `X_API_KEY`(문자열)-(필수) 읽기/쓰기에 사용됩니다. API를 요청하는 개발자 또는 애플리케이션을 인증해야 합니다.
- `SELECTED_FIELDS`(List<String>)-기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Adobe Analytics 계정 생성

1. Adobe 파트너 프로그램에 액세스하여 교환 파트너 프로그램에 등록합니다.
2. 교환 프로그램 가입을 선택합니다.
3. 회사 이메일 주소를 사용하여 계정을 등록하거나 생성합니다.
4. 제안 상자에서 Adobe Analytics 제품 구독이 있는 적절한 회사를 선택합니다.
5. 계정이 활성 Adobe Analytics 구독이 있는 유효한 조직(사용 가능한 목록에서)에 등록되었는지 확인합니다.
6. 회사 관리자가 승인한 후 승인 이메일의 링크를 클릭하여 계정을 활성화합니다.

생성한 계정이 Adobe Analytics 서비스에 액세스할 수 있는지 확인

1. Adobe Admin Console에 로그인합니다.
2. 페이지 오른쪽 상단에 있는 조직 이름을 확인하여 올바른 회사에 로그인했는지 확인합니다.
3. 제품을 선택하고 Adobe Analytics를 사용할 수 있는지 확인합니다.

> **Note**
>
> 사용할 수 있는 조직이 없거나 Adobe Analytics 제품이 회색으로 표시되거나 사용할 수 없는 경우 계정이 조직과 연결되지 않았거나 활성 Adobe Analytics 구독이 없는 것일 수 있습니다. 이 서비스에 대한 액세스를 요청하려면 시스템 관리자에게 문의하세요.

프로젝트 및 **OAuth2.0** 자격 증명 생성

1. OAuth 2.0 앱을 생성할 Adobe Analytics 계정에 로그인합니다.
2. 프로젝트를 선택한 다음 새 프로젝트 생성을 선택합니다.
3. 프로젝트를 추가하려면 프로젝트에 추가를 선택한 다음 API를 선택합니다.
4. Adobe Analytics API를 선택합니다.
5. 사용자 인증으로 OAUTH를 선택합니다.
6. 웹을 0AUTH로 선택하고 리디렉션 URI를 제공합니다.

   리디렉션 URI 및 패턴은 다음을 참조하세요.

- OAuth 2.0 기본 리디렉션 URI-기본 리디렉션 URI는 인증 프로세스 중에 Adobe가 액세스할 페이지의 URL입니다. 예제: https://ap-southeast-2.console.aws.amazon.com/appflow/oauth
- OAuth 2.0 리디렉션 URI 패턴-리디렉션 URI 패턴은 로그인 흐름이 완료될 때 Adobe가 리디렉션(요청된 경우)할 수 있는 URI 경로(또는 쉼표로 구분된 경로 목록)입니다. 예제: https://ap-southeast-2\\.console\\.aws\\.amazon\\.com

7. 다음 법위를 추가합니다.

   - openid
   - read_organizations
   - additional_info.projectedProductContext
   - additional_info.job_function

8. 자격 증명 저장을 선택합니다.
9. 앱을 생성한 후 Client ID 및 Client Secret 값을 텍스트 파일에 복사합니다.

제한 사항

다음은 Adobe Analytics 커넥터의 제한 사항입니다.

- Adobe Analytics는 필드 기반 분할과 레코드 기반 분할을 지원하지 않습니다. 분할하는 필드를 쿼리할 수 없으므로 필드 기반 분할은 지원되지 않습니다. 페이지 매김을 위해 '오프셋'을 가져오는 프로비저닝이 없으므로 레코드 기반 분할을 지원할 수 없습니다.
- Report Top Item 엔터티에서 startDate 및 endDate 쿼리 파라미터가 예상대로 작동하지 않습니다. 응답은 이 파라미터들을 기준으로 필터링되지 않으므로이 엔터티의 필터 및 증분 흐름에 문제가 발생합니다.
- Annotation, Calculated Metrics, Calculated Metrics Function, Date Ranges, Dimension, Metric, Project, Report Top Items, 및 Segment 엔터티의 경우 locale 쿼리 파라미터는 응답의 현지화된 섹션에 사용할 언어를 지정하며 레코드를 필터링하지 않습니다. 예를 들어 locale="ja_JP"는 일본어로 데이터를 표시합니다.
- Report Top Item 엔터티-dateRange 및 lookupNoneValues 필드의 필터가 현재 작동하지 않습니다.
- Segment 엔터티: 필터 값 includeType="templates"인 경우 다른 필드의 필터가 작동하지 않습니다.
- Date Range 엔터티-curatedRsid 필드의 필터가 작동하지 않습니다.

- `Metric entity` 엔터티-“false” 값을 사용하여 세그먼트화 가능한 필드를 필터링하면 true 값과 false 값이 모두 표시됩니다.

Adobe Marketo Engage에 연결

Adobe Marketo Engage는 마케팅 담당자가 잠재 고객과 고객에게 맞춤형 다중 채널 프로그램과 캠페인을 관리할 수 있는 마케팅 자동화 플랫폼입니다.

주제

- Adobe Marketo Engage의 AWS Glue 지원
- 연결 생성 및 사용 API 작업이 포함된 정책
- Adobe Marketo Engage 구성
- Adobe Marketo Engage 연결 구성
- Adobe Marketo Engage 엔터티에서 읽기
- Adobe Marketo Engage 연결 옵션
- Adobe Marketo Engage 커넥터에 대한 제한 사항 및 참고 사항

Adobe Marketo Engage의 AWS Glue 지원

AWS Glue는 다음과 같이 Adobe Marketo Engage를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Adobe Marketo Engage에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Adobe Marketo Engage API 버전

다음 Adobe Marketo Engage API 버전이 지원됩니다.

- v1

버전별 엔터티 지원은 소스에 대해 지원되는 엔터티를 참조하세요.

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려면 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS 관리 콘솔을 사용할 때 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

Adobe Marketo Engage 구성

AWS Glue 를 사용하여 Adobe Marketo Engage에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- 클라이언트 자격 증명이 있는 Adobe Marketo Engage 계정이 있습니다.
- Adobe Marketo Engage 계정에는 유효한 라이선스가 있는 API 액세스 권한이 있습니다.

이러한 요구 사항을 충족하면 Adobe Marketo Engage 계정에 AWS Glue 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Adobe Marketo Engage에서 다른 작업을 수행할 필요가 없습니다.

OAuth 2.0 자격 증명 가져오기

인스턴스에 인증된 호출을 수행할 수 있도록 API 보안 인증을 얻으려면 Adobe Marketo Engage 개발자 안내서RESTAPI의 섹션을 참조하세요.

Adobe Marketo Engage 연결 구성

Adobe Marketo Engage는 OAuth2에 대한 CLIENT CREDENTIALS 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 클라이언트가 사용자의 컨텍스트 외부에서 액세스 토큰을 얻는 데 사용하므로 2각 OAuth 2.0으로 간주됩니다. AWS Glue는 클라이언트 ID와 클라이언트 암호를 사용하여 사용자가 정의한 사용자 지정 서비스에서 제공하는 Adobe Marketo Engage API를 인증할 수 있습니다.
- 각 사용자 지정 서비스는 API 전용 사용자가 소유하며, API 전용 사용자는 서비스에 특정 작업을 수행하도록 권한을 부여하는 역할 및 권한 집합을 가집니다. 액세스 토큰은 단일 사용자 지정 서비스와 연결됩니다.
- 이 권한 부여 유형은 수명이 짧은 액세스 토큰을 생성하며 ID 엔드포인트를 호출하여 갱신할 수 있습니다.
- 클라이언트 자격 증명이 포함된 OAuth 2.0용 퍼블릭 Adobe Marketo Engage 설명서는 Adobe Marketo Engage 개발자 안내서의 Authentication을 참조하세요.

Adobe Marketo Engage 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결당 시크릿을 생성해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Adobe Marketo Engage를 선택합니다.
   b. 연결하려는 Adobe Marketo Engage 인스턴스의 `INSTANCE_URL`을 제공합니다.

c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Adobe Marketo Engage 엔터티에서 읽기

사전 조건

읽으려는 Adobe Marketo Engage 객체. 객체 이름(리드, 활동 또는 사용자 지정 객체)이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티(동기식):

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
리드	예	예	아니요	예	아니요
활동	예	예	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
사용자 지정 객체	예	예	아니요	예	아니요

소스에 대해 지원되는 엔터티(비동기식):

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
리드	예	아니요	아니요	예	예
활동	예	아니요	아니요	예	아니요
사용자 지정 객체	예	아니요	아니요	예	예

예시:

```
adobe-marketo-engage_read = glueContext.create_dynamic_frame.from_options(
    connection_type="adobe-marketo-engage",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "leads",
        "API_VERSION": "v2",
        "INSTANCE_URL": "https://539-t**-6**.mktorest.com"
    }
```

Adobe Marketo Engage 엔터티 및 필드 세부 정보:

정적 메타데이터를 포함하는 엔터티:

개체	필드	데이터 유형	지원되는 연산자
활동	sinceDatetime(동기식에서만 지원됨)	DateTime	>=(동기 모드에서만 해당)

개체	필드	데이터 유형	지원되는 연산자
	createdAt(비동기식에서만 지원됨)	DateTime	between(비동기 모드에 대해서만 해당)
	activitiesTypeId	Integer	=
	adobe-marketo-engageGUID	Long	=(동기 모드에 대해서만 해당)
	leadId	Long	N/A
	activityDate	DateTime	N/A
	campaignId	Long	N/A
	primaryAttributeValueId	Integer	N/A
	primaryAttributeValue	String	해당 사항 없음
	attributes	String	해당 사항 없음

동적 메타데이터를 포함하는 엔터티:

다음 엔터티에 대해 Adobe Marketo Engage에서는 메타데이터를 동적으로 가져오도록 엔드포인트를 제공하므로 운영자 지원은 각 엔터티의 데이터 유형 수준에서 캡처됩니다.

개체	데이터 유형	지원되는 연산자
리드	Integer	=(동기 모드에 대해서만 해당)
	DateTime	between(비동기 모드에 대해서만 해당)
	String	=(동기 모드에 대해서만 해당)
	Long	N/A
	불	N/A

개체	데이터 유형	지원되는 연산자
	날짜	N/A
	Float	N/A
사용자 지정 객체	Integer	N/A
	DateTime	between(비동기 모드에 대해서만 해당)
	String	=(동기 모드에 대해서만 해당)
	날짜	N/A
	Long	N/A
	불	N/A
	Float	N/A

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  DateTime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2024-07-01T00:00:00.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

다음 표에서는 엔터티 분할 필드 지원 세부 정보를 설명합니다.

엔터티 이름	분할 필드	데이터 유형
리드	createdAt	DateTime
	updateAt	DateTime
사용자 지정 객체	updatedAt	DateTime

예시:

```
adobe-marketo-engage_read = glueContext.create_dynamic_frame.from_options(
    connection_type="adobe-marketo-engage",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "leads",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "createdAt"
        "LOWER_BOUND": "2024-07-01T00:00:00.000Z"
        "UPPER_BOUND": "2024-07-02T00:00:00.000Z"
        "NUM_PARTITIONS": "10"
    }
```

Adobe Marketo Engage 연결 옵션

다음은 Adobe Marketo Engage의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Adobe Marketo Engage의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Adobe Marketo Engage Rest API 버전입니다. 예: v1.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.

- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- TRANSFER_MODE(문자열) - 기본값: SYNC. 비동기 읽기에 사용됩니다.

Adobe Marketo Engage 커넥터에 대한 제한 사항 및 참고 사항

다음은 Adobe Marketo Engage 커넥터의 제한 사항 또는 참고 사항입니다.

- 'sinceDatetime' 및 'activityTypeId'는 동기화 활동 엔터티의 필수 필터 파라미터입니다.
- 구독에는 하루에 50,000개의 API 직접 호출이 할당됩니다(오전 12:00 CST에 매일 재설정됨). Adobe Marketo Engage 구독의 일부로 추가 일일 용량을 구매할 수 있습니다.
- 날짜 범위 필터(createdAt 또는 updatedAt)의 최대 기간은 31일입니다.
- 구독에는 언제든지 대기열에 최대 10개의 대량 추출 작업이 할당됩니다.
- 기본적으로 추출 작업은 하루에 500MB로 제한됩니다(오전 12:00 CST에 매일 재설정됨). Adobe Marketo Engage 구독의 일부로 추가 일일 용량을 구매할 수 있습니다.
- 동시 내보내기 최대 작업 수는 2입니다.
- 대기열에 있는 내보내기 작업의 최대 수(현재 내보내는 작업 포함)는 10개입니다.
- 허용되는 최대 파일 크기는 대량 작업에서 추출할 수 있는 1GB입니다.
- 비동기 작업이 생성되면 파일 보존 기간은 만료 7일 전입니다.
- createdAt 또는 updatedAt은 Async Leads 엔터티의 필수 필터 파라미터입니다.
- createdAt은 비동기 활동 엔터티의 필수 필터 파라미터입니다.
- updatedAt은 비동기 사용자 지정 객체 엔터티의 필수 필터 파라미터입니다.

자세한 내용은 Adobe Marketo Engage 통합 모범 사례 및 대량 추출을 참조하세요.

AWS Glue Studio에서 Amazon Redshift에 연결

> **Note**
>
> AWS Glue for Spark를 사용하여 AWS Glue Studio 외부에 있는 Amazon Redshift 데이터베이스의 테이블에서 읽고 쓸 수 있습니다. 프로그래밍 방식으로 AWS Glue 작업을 통해 Amazon Redshift을(를) 구성하려면 Redshift 연결을(를) 참조하세요.

AWS Glue에서는 Amazon Redshift에 대한 기본 제공 지원을 제공합니다. AWS Glue Studio에서는 Amazon Redshift에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스를 제공합니다.

주제

- Amazon Redshift 연결 생성
- Amazon Redshift 소스 노드 생성
- Amazon Redshift 대상 노드 생성
- 고급 옵션

Amazon Redshift 연결 생성

필요한 권한

Amazon Redshift 클러스터와 Amazon Redshift 서버리스 환경을 사용하려면 추가 권한이 필요합니다. ETL 작업에 권한을 추가하는 방법에 대한 자세한 내용은 ETL 작업에 필요한 IAM 권한 검토를 참조하세요.

- redshift:DescribeClusters
- redshift-serverless:ListWorkgroups
- redshift-serverless:ListNamespaces

개요

Amazon Redshift 연결을 추가할 때 AWS Glue Studio에서 데이터 소스 - Redshift 노드를 추가하면 기존 Amazon Redshift 연결을 선택하거나 새 연결을 생성할 수 있습니다.

AWS Glue에서는 Amazon Redshift 클러스터와 Amazon Redshift 서버리스 환경을 모두 지원합니다. 연결을 생성할 때 Amazon Redshift 서버리스 환경은 연결 옵션 옆에 서버리스 레이블을 표시합니다.

Amazon Redshift 연결을 생성하는 방법에 대한 자세한 내용은 Amazon Redshift 간 데이터 이동을 참조하세요.

Amazon Redshift 소스 노드 생성

필요한 권한

Amazon Redshift 데이터 소스를 사용하는 AWS Glue Studio 작업에는 추가 권한이 필요합니다. ETL 작업에 권한을 추가하는 방법에 대한 자세한 내용은 ETL 작업에 필요한 IAM 권한 검토를 참조하세요.

Amazon Redshift 연결을 사용하려면 다음 권한이 필요합니다.

- redshift-data:ListSchemas
- redshift-data:ListTables
- redshift-data:DescribeTable
- redshift-data:ExecuteStatement
- redshift-data:DescribeStatement
- redshift-data:GetStatementResult

Amazon Redshift 데이터 소스 추가

데이터 소스 - Amazon Redshift 노드를 추가하려면:

1. Amazon Redshift 액세스 유형을 선택합니다.
   - 직접 데이터 연결(권장) - Amazon Redshift 데이터에 직접 액세스하려면 이 옵션을 선택합니다. 이 옵션은 권장 옵션이며 기본값이기도 합니다.
   - Data Catalog tables - 사용하려는 데이터 카탈로그 테이블이 있는 경우 이 옵션을 선택합니다.
2. 직접 데이터 연결을 선택하는 경우 Amazon Redshift 데이터 소스의 연결을 선택합니다. 이 경우 연결이 이미 존재하며 기존 연결 중에서 선택할 수 있다고 가정합니다. 연결을 생성해야 하는 경우 Redshift 연결 생성을 선택합니다. 자세한 내용은 커넥터 및 연결 사용 개요를 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다. URL, 보안 그룹, 서브넷, 가용 영역, 설명, 생성 날짜(UTC) 및 최종 업데이트(UTC) 타임스탬프를 비롯한 연결 정보가 표시됩니다.
3. 다음과 같은 Amazon Redshift 소스 옵션을 선택합니다.
   - 단일 테이블 선택 - 단일 Amazon Redshift 테이블에서 액세스하려는 데이터가 들어 있는 테이블입니다.
   - 사용자 지정 쿼리 입력 - 사용자 지정 쿼리를 기반으로 여러 Amazon Redshift 테이블의 데이터 세트에 액세스할 수 있습니다.
4. 단일 테이블을 선택한 경우 Amazon Redshift 스키마를 선택합니다. 선택할 수 있는 스키마 목록은 선택한 테이블에 따라 결정됩니다.

또는 사용자 지정 쿼리 입력을 선택합니다. 여러 Amazon Redshift 테이블에서 사용자 지정 데이터 세트에 액세스하려면 이 옵션을 선택합니다. 이 옵션을 선택하는 경우 Amazon Redshift 쿼리를 입력합니다.

Amazon Redshift 서버리스 환경에 연결할 때 사용자 지정 쿼리에 다음 권한을 추가합니다.

```
GRANT SELECT ON ALL TABLES IN <schema> TO PUBLIC
```

스키마 추론을 선택하여 입력한 쿼리를 기반으로 스키마를 읽을 수 있습니다. Redshift 쿼리 편집기 열기를 선택하여 Amazon Redshift 쿼리를 입력할 수도 있습니다. 자세한 내용은 쿼리 편집기를 사용하여 데이터베이스 쿼리을 참조하세요.

5. 성능 및 보안에서 Amazon S3 스테이징 디렉터리와 IAM 역할을 선택합니다.
   - Amazon S3 스테이징 디렉터리 - 데이터를 임시로 스테이징할 Amazon S3 위치를 선택합니다.
   - IAM 역할 - 선택한 Amazon S3 위치에 기록할 수 있는 IAM 역할을 선택합니다.
6. 사용자 지정 Redshift 파라미터 - 선택 사항에서 파라미터와 값을 입력합니다.

Amazon Redshift 대상 노드 생성

필요한 권한

Amazon Redshift 데이터 대상을 사용하는 AWS Glue Studio 작업에는 추가 권한이 필요합니다. ETL 작업에 권한을 추가하는 방법에 대한 자세한 내용은 ETL 작업에 필요한 IAM 권한 검토를 참조하세요.

Amazon Redshift 연결을 사용하려면 다음 권한이 필요합니다.

- redshift-data:ListSchemas
- redshift-data:ListTables

Amazon Redshift 대상 노드 추가

Amazon Redshift 대상 노드를 생성하려면:

1. 기존 Amazon Redshift 테이블을 대상으로 선택하거나 새 테이블 이름을 입력합니다.
2. 데이터 대상 - Redshift 대상 노드를 사용하는 경우 다음 옵션 중에서 선택할 수 있습니다.

- 추가 - 테이블이 이미 있는 경우 모든 새 데이터를 테이블에 삽입으로 덤프합니다. 테이블이 없으면 새로 생성한 후 새 데이터를 모두 삽입합니다.

  또한 대상 테이블의 기존 레코드를 업데이트(업서트)하려면 상자를 선택합니다. 테이블이 먼저 있어야 합니다. 그렇지 않으면 작업에 실패합니다.

- 병합 - AWS Glue는 사용자가 지정한 조건에 따라 대상 테이블에 데이터를 추가하거나 업데이트합니다.

  > **Note**
  >
  > AWS Glue에서 병합 작업을 사용하려면 Amazon Redshift 병합 기능을 활성화해야 합니다. Amazon Redshift 인스턴스에 대해 병합을 활성화하는 방법에 대한 지침은 MERGE(평가판)를 참조하세요.

  다음과 같은 옵션을 선택합니다.

  - 키 및 간단한 작업 선택 - 소스 데이터와 대상 데이터 세트 사이에서 일치하는 키로 사용할 열을 선택합니다.

    일치하는 경우 다음 옵션을 지정합니다.

    - 대상 데이터 세트의 레코드를 소스의 데이터로 업데이트합니다.
    - 대상 데이터 세트에서 레코드를 삭제합니다.

    일치하지 않는 경우 다음 옵션을 지정합니다.

    - 소스 데이터를 대상 데이터 세트에 새 행으로 삽입합니다.
    - 아무 작업 안 함.

  - 사용자 지정 MERGE 명령문 입력 - 그런 다음 병합 명령문 검증을 선택하여 명령문이 유효한지 여부를 검증할 수 있습니다.

- 잘라내기 - 테이블이 이미 있는 경우 먼저 대상 테이블의 콘텐츠를 지워서 테이블 데이터를 잘라냅니다. 잘라내기에 성공하면 모든 데이터를 삽입합니다. 테이블이 없는 경우 테이블을 생성하고 모든 데이터를 삽입합니다. 잘라내기에 실패하면 작업에 실패합니다.

- 삭제 - 테이블이 이미 있는 경우 테이블 메타데이터와 데이터를 삭제합니다. 삭제에 성공하면 모든 데이터를 삽입합니다. 테이블이 없는 경우 테이블을 생성하고 모든 데이터를 삽입합니다. 삭제에 실패하면 작업에 실패합니다.

- 생성 - 기본 이름을 사용하여 새 테이블을 생성합니다. 테이블 이름이 이미 있는 경우 고유성을 유지하기 위해 이름에 `job_datetime`의 이름 접미사를 사용하는 새 테이블을 생성합니다. 그러면 모든 데이터가 새 테이블에 삽입됩니다. 테이블이 있는 경우 최종 테이블 이름에는 접미사가 추가됩니다. 테이블이 없는 경우 테이블이 생성됩니다. 어느 경우든 새 테이블이 생성됩니다.

고급 옵션

Using the Amazon Redshift Spark connector on AWS Glue를 참조하세요.

Asana에 연결

Asana는 팀이 작업과 프로젝트를 구성, 계획, 완료할 수 있도록 지원하는 클라우드 기반 팀 협업 솔루션입니다. Asana 사용자인 경우 계정에는 워크스페이스, 프로젝트, 작업, 팀 등에 대한 데이터가 포함됩니다. Asana에서 특정 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송할 수 있습니다.

주제

- AWS Glue의 Asana 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Asana 구성
- Asana 연결 구성
- Asana 엔터티에서 읽기
- Asana 연결 옵션
- Asana 계정 생성
- 제한 사항

AWS Glue의 Asana 지원

AWS Glue에서는 다음과 같이 Asana을 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Asana에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Asana API 버전

1.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess-정책이 연결된 자격 증명이 AWS Management 콘솔을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Asana 구성

AWS Glue를 사용하여 Asana에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호가 있는 Asana 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Asana 계정 생성을 참조하세요.
- AWS Glue에 대한 서비스 액세스 권한으로 AWS 계정을 생성해야 합니다.
- Asana 계정에서 다음 리소스 중 하나를 생성했는지 확인합니다.
  - `OAuth 2.0` 인증을 지원하는 개발자 앱이 있습니다. 자세한 지침은 Asana 개발자 설명서의 OAuth를 참조하세요. 또는 the section called “Asana 계정 생성” 단원을 참조하십시오.
  - 개인 액세스 토큰입니다. 자세한 내용은 Asana 개발자 설명서의 개인 액세스 토큰https://developers.asana.com/docs/personal-access-token을 참조하세요.

이러한 요구 사항을 충족하면 Adobe Analytics 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Adobe Analytics에서 다른 작업을 수행하지 않아도 됩니다.

Asana 연결 구성

Asana는 `OAuth2`에 대한 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' `OAuth`로 간주됩니다. 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Asana에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Asana로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

`AUTHORIZATION_CODE OAuth` 흐름을 위해 연결된 앱을 생성하는 방법에 대한 퍼블릭 Asana 설명서는 Asana API를 참조하세요.

Asana 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

- > **Note**
  >
  > AWS Glue에서 연결에 대한 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 연결 유형을 선택할 때 Asana을 선택합니다.
   b. Asana 환경을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Asana 엔터티에서 읽기

사전 조건

읽으려는 Asana 객체입니다. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

소스에 대해 지원되는 엔터티

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Workspace	아니요	예	아니요	예	아니요
Tag	아니요	예	아니요	예	아니요
User	아니요	예	아니요	예	아니요
Portfolio	아니요	예	아니요	예	아니요
Team	아니요	예	아니요	예	아니요
Project	예	예	아니요	예	아니요
Section	아니요	예	아니요	예	아니요
Task	예	아니요	아니요	예	예
Goal	예	예	아니요	예	아니요
AuditLogEvent	예	예	아니요	예	아니요
Status Update	예	예	아니요	예	아니요
Custom Field	아니요	예	아니요	예	아니요
Project Brief	예	아니요	아니요	예	예

예

```
read_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Asana",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "task/workspace:xxxx",
        "API_VERSION": "1.0",
        "PARTITION_FIELD": "created_at",
        "LOWER_BOUND": "2024-02-05T14:09:30.115Z",
```

```
        "UPPER_BOUND": "2024-06-07T13:30:00.134Z",
        "NUM_PARTITIONS": "3"
    }
```

Asana 엔터티 및 필드 세부 정보

- Workspace
- Tag
- User
- Portfolio
- Team
- Project
- Section
- Task
- Goal
- AuditLogEvent
- Status Update
- Custom Field
- Project Brief

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: 2024-06-07T13:30:00.134Z.
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준의 분할 필드 지원 세부 정보는 다음 표에 캡처되어 있습니다.

Entity Name	분할 필드	데이터 형식
Task	created_at	DateTime
Task	modified_at	DateTime

예제

```
read_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Asana",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "task/workspace:xxxx",
        "API_VERSION": "1.0",
        "PARTITION_FIELD": "created_at",
        "LOWER_BOUND": "2024-02-05T14:09:30.115Z",
        "UPPER_BOUND": "2024-06-07T13:30:00.134Z",
        "NUM_PARTITIONS": "3"
    }
```

Asana 연결 옵션

다음은 Asana의 연결 옵션입니다.

- ENTITY_NAME(문자열)-(필수) 읽기/쓰기에 사용됩니다. Asana에서의 객체 이름입니다.
- API_VERSION(문자열)-(필수) 읽기/쓰기에 사용됩니다. 사용할 Asana Rest API 버전입니다. 예: 1.0.
- SELECTED_FIELDS(List<String>)-기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Asana 계정 생성

1. Asana 계정에 가입하고 가입을 선택합니다.
2. 로그인하면 계정 설정 페이지로 리디렉션됩니다. 다음 단계를 완료합니다.
   - 계정 설정 양식을 검토합니다.
   - 관련 세부 정보를 모두 입력하여 Asana 계정을 생성합니다.
   - 정보가 정확한지 다시 확인합니다.
3. 계정 생성 또는 제출(정확한 버튼 텍스트는 다를 수 있음)을 선택하여 계정 설정을 완료합니다.

OAuth2.0용 Asana에서 앱 생성

1. Asana 고객 자격 증명을 사용하여 Asana 계정에 로그인합니다.
2. 오른쪽 상단 모서리에서 사용자 프로필 아이콘을 선택하고 드롭다운 메뉴에서 내 설정을 선택합니다.
3. 앱 탭을 선택한 다음 개발자 앱 관리를 선택합니다.
4. 새 앱 생성을 선택하고 관련 세부 정보를 입력합니다.
5. 앱 생성을 선택합니다.
6. 내 앱 페이지에서:
   a. OAuth를 선택하고 앱 자격 증명 섹션에서 클라이언트 ID와 클라이언트 보안 암호를 기록해 둡니다.
   b. 리디렉션 URL 섹션에서 필요한 리디렉션 URL을 추가합니다.

> **Note**
>
> `https://{aws-region-code}.console.aws.amazon.com/gluestudio/oauth`의 이 형식을 사용하여 리디렉션 URI를 입력합니다. 예시: 미국 동부(버지니아 북부)의 경우 `https://us-east-1.console.aws.amazon.com/gluestudio/oauth`를 사용합니다.

PAT 토큰용 Asana에서 앱 생성

1. Asana 고객 자격 증명을 사용하여 Asana 계정에 로그인합니다.

2. 오른쪽 상단 모서리의 사용자 프로필 아이콘에서 선택하고 드롭다운 메뉴에서 내 프로필 설정을 선택합니다.
3. 앱 탭을 선택한 다음 서비스 계정을 선택합니다.
4. 새 앱 생성을 선택하고 관련 세부 정보를 입력합니다.
5. 서비스 계정 추가를 선택하세요.
6. 다음 페이지에 토큰이 표시되고 토큰을 복사하여 안전하게 저장합니다.

> **Important**
>
> 이 토큰은 한 번만 표시됩니다. 복사하여 안전하게 저장해야 합니다.

제한 사항

다음은 Asana 커넥터의 제한 사항입니다.

- 엔터프라이즈 도메인의 서비스 계정은 감사 로그 API 엔드포인트에만 액세스할 수 있습니다. 이러한 엔드포인트에 액세스하려면 서비스 계정의 개인 액세스 토큰으로 인증해야 합니다.
- Goal 엔터티는 프리미엄 요금제 이상이 있는 사용자 계정에 대해서만 액세스할 수 있습니다.
- `Audit Log Event Entity`-커넥터에서 `start_at` 및 `end_at` 필드는 필터링 및 증분 전송을 지원하기 위해 단일 필드 "start_end_at"로 결합됩니다.
- 필드 분할은 greater-than-or-equal-to 연산자와 less-than-or-equal-to 연산자를 지원하더라도 `Date` 필드에 대해 지원되지 않습니다. 시나리오: `partitionField`를 due_on(데이터 유형: 날짜)으로, `lowerBound`를 `2019-09-14`로, `upperBound`를 `2019-09-16`으로, `numPartition`을 2로 작업을 생성했습니다. 엔드포인트 URL의 필터 부분은 다음과 같이 생성됩니다.
  - partition1: due_on.before=2019-09-14&due_on.after=2019-09-14
  - partition2: due_on.before=2019-09-15&due_on.after=2019-09-15 Output:
  - partition1에서는 due_date가 2019-09-14 및 2019-09-15인 데이터를 가져옵니다.
  - partition2에서는 다른 데이터와 함께 due_date가 2019-09-15(partition1에서 가져온 데이터)인 동일한 데이터를 가져와 데이터 중복을 일으킵니다.
- SaaS 끝에서 잘못된 요청 오류가 발생하므로 동일한 필드에서 필터링 및 분할을 지원할 수 없습니다.
- 작업 엔터티에는 필터 기준에 최소 1개의 필드가 필요합니다. Asana에는 시간 기반 필드를 기반으로 레코드를 정렬하지 않고 페이지 매김이 식별되지 않는 제한이 있습니다. 따라서 Created_at 필드는 페이지 매김과 함께 다음 레코드 세트를 구분하는 데 사용됩니다. Created_at 필드는 필터에서 필수

로 표시되며, 제공되지 않은 경우 기본값은 2000-01-01T00:00:00Z입니다. 페이지 매김에 대한 자세한 내용은 작업 영역의 작업을 참조하세요.

AWS Glue Studio에서 Azure Cosmos DB에 연결

AWS Glue에서는 Azure Cosmos DB를 기본으로 지원합니다. AWS Glue Studio에서는 NoSQL의 Azure Cosmos DB에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스를 제공합니다.

주제

- Azure Cosmos DB 연결 생성
- Azure Cosmos DB 소스 노드 생성
- Azure Cosmos DB 대상 노드 생성
- 고급 옵션

Azure Cosmos DB 연결 생성

사전 조건:

- Azure에서는, AWS Glue, `cosmosKey`에서 사용할 Azure Cosmos DB 키를 식별하거나 생성해야 합니다. 자세한 내용은 Azure 설명서의 Azure Cosmos DB의 데이터에 대한 보안 액세스를 참조하십시오.

Azure Cosmos DB에 대한 연결 구성 방법:

1. AWS Secrets Manager에서 Azure Cosmos DB 키를 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *`secretName`*을 유지합니다.

   - 키/값 페어를 선택하면 값 *`cosmosKey`*가 포함된 키 `spark.cosmos.accountKey`에 대한 페어를 생성합니다.
2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *`connectionName`*을 유지합니다.

   - 연결 유형을 선택할 때는 Azure Cosmos DB를 선택합니다.

- AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

Azure Cosmos DB 소스 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Azure Cosmos DB 연결 생성”에서 설명한 대로 AWS Secrets Manager 암호로 구성된 AWS Glue Azure Cosmos DB 연결입니다
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 읽으려는 NoSQL 컨테이너용 Azure Cosmos 데이터베이스. 컨테이너의 식별 정보가 필요합니다.

  NoSQL용 Azure Cosmos 컨테이너는 해당 데이터베이스 및 컨테이너로 식별됩니다.
  NoSQL API용 Azure Cosmos에 연결할 때 데이터베이스, *cosmosDBName*, 및 컨테이너너, *cosmosContainerName*, 이름을 제공해야 합니다.

Azure Cosmos DB 데이터 소스 추가

데이터 소스 - Azure Cosmos DB 노드 추가하는 방법:

1. Azure Cosmos DB 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 Azure Cosmos DB 연결 생성을 선택합니다. 자세한 내용은 이전the section called “Azure Cosmos DB 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. Cosmos DB 데이터베이스 이름 선택 - 읽으려는 *####### ### cosmosDBName*을 입력합니다.
3. Azure Cosmos DB 컨테이너 선택 - 읽으려는 컨테이너의 이름인 *cosmosContainerName*을 입력합니다.
4. 선택적으로 Azure Cosmos DB 사용자 지정 쿼리를 선택합니다. Azure Cosmos DB에서 특정 정보를 검색할 수 있는 SQL SELECT 쿼리를 제공합니다.
5. 사용자 지정 Azure Cosmos 속성(선택 사항)에서 필요한 경우 파라미터와 값을 입력합니다.

Azure Cosmos DB 대상 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Azure Cosmos DB 연결 생성”에서 설명한 대로 AWS Secrets Manager 암호로 구성된 AWS Glue Azure Cosmos DB 연결입니다

- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 쓰려는 Azure Cosmos DB 테이블. 컨테이너의 식별 정보가 필요합니다. 연결 방법을 호출하기 전에 컨테이너를 만들어야 합니다.

  NoSQL용 Azure Cosmos 컨테이너는 해당 데이터베이스 및 컨테이너로 식별됩니다. NoSQL API용 Azure Cosmos에 연결할 때 데이터베이스, *cosmosDBName*, 및 컨테이너너, *cosmosContainerName*, 이름을 제공해야 합니다.

Azure Cosmos DB 데이터 대상 추가

데이터 대상 - Azure Cosmos DB 노드 추가 방법:

1. Azure Cosmos DB 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 Azure Cosmos DB 연결 생성을 선택합니다. 자세한 내용은 이전the section called “Azure Cosmos DB 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. Cosmos DB 데이터베이스 이름 선택 - 읽으려는 *####### ### cosmosDBName*을 입력합니다.
3. Azure Cosmos DB 컨테이너 선택 - 읽으려는 컨테이너의 이름인 *cosmosContainerName*을 입력합니다.
4. 사용자 지정 Azure Cosmos 속성(선택 사항)에서 필요한 경우 파라미터와 값을 입력합니다.

고급 옵션

Azure Cosmos DB 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “Azure Cosmos DB 연결”를 참조하세요.

AWS Glue Studio에서 Azure SQL에 연결

AWS Glue에서는 Azure SQL을 기본으로 지원합니다. AWS Glue Studio에서는 Azure SQL에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스를 제공합니다.

주제

- Azure SQL 연결 만들기

- Azure SQL 소스 노드 생성
- Azure SQL 대상 노드 생성
- 고급 옵션

Azure SQL 연결 만들기

AWS Glue에서 Azure SQL에 연결하려면 Azure SQL 보안 인증 정보를 만들어 AWS Secrets Manager 암호에 저장한 다음 해당 암호를 Azure SQL AWS Glue 연결에 연결해야 합니다.

Azure SQL에 대한 연결을 구성하는 방법:

1. AWS Secrets Manager에서 Azure SQL 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
   - 키/값 페어를 선택하면 값 *azuresqlUsername*이 포함된 키 `user`에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *azuresqlPassword*가 포함된 키 `password`에 대한 페어를 생성합니다.
2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때 Azure SQL를 선택합니다.
   - Azure SQL URL을 제공할 때는 JDBC 엔드포인트 URL을 제공하십시오.

     목록은 `jdbc:sqlserver://`*databaseServerName*`:`*databasePort*`;databaseName=`*azuresqlDBnam* 형식이어야 합니다.

     AWS Glue는 다음 URL 속성이 필요합니다.

     - `databaseName` - 연결할 Azure SQL의 기본 데이터베이스입니다.

     Azure SQL 관리형 인스턴스용 JDBC URL에 대한 자세한 내용은 Microsoft 설명서를 참조하십시오.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

Azure SQL 소스 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Azure SQL 연결 만들기”에서 설명한 대로 AWS Secrets Manager 암호로 구성된 AWS Glue Azure SQL 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 읽으려는 Azure SQL 테이블. *`tableName`*.

  Azure SQL 테이블은 데이터베이스, 스키마 및 테이블 이름으로 식별됩니다. Azure SQL에 연결할 때 데이터베이스 이름과 테이블 이름을 제공해야 합니다. 스키마가 기본값인 "public"이 아닌 경우에도 스키마를 제공해야 합니다. 데이터베이스는 *`connectionName`*의 URL 속성을, `dbtable`을 통해 스키마 및 테이블 이름을 제공 받습니다.

Azure SQL 데이터 소스 추가

데이터 소스 - Azure SQL 노드 추가 방법:

1. Azure SQL 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 Azure SQL 연결 생성을 선택합니다. 자세한 내용은 이전the section called “Azure SQL 연결 만들기” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. Azure SQL 소스 옵션을 선택합니다.

   - 단일 테이블 선택 - 단일 테이블에서 모든 데이터에 액세스할 수 있습니다.
   - 사용자 지정 쿼리 입력 - 사용자 지정 쿼리를 기반으로 여러 테이블의 데이터 세트에 액세스할 수 있습니다.
3. 단일 테이블을 선택한 경우 *`tableName`*을 입력합니다.

   사용자 지정 쿼리 입력을 선택한 경우 TransactSQL SELECT 쿼리를 입력합니다.
4. 사용자 지정 Azure SQL 속성에서 필요한 경우 파라미터와 값을 입력합니다.

Azure SQL 대상 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Azure SQL 연결 만들기”에서 설명한 대로 AWS Secrets Manager 암호로 구성된 AWS Glue Azure SQL 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 쓰려는 Azure SQL 테이블, *`tableName`*.

  Azure SQL 테이블은 데이터베이스, 스키마 및 테이블 이름으로 식별됩니다. Azure SQL에 연결할 때 데이터베이스 이름과 테이블 이름을 제공해야 합니다. 스키마가 기본값인 "public"이 아닌 경우에도 스키마를 제공해야 합니다. 데이터베이스는 *`connectionName`*의 URL 속성을, `dbtable`을 통해 스키마 및 테이블 이름을 제공 받습니다.

Azure SQL 데이터 대상 추가

데이터 대상 - Azure SQL 노드 추가 방법:

1. Azure SQL 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 Azure SQL 연결 생성을 선택합니다. 자세한 내용은 이전the section called “Azure SQL 연결 만들기” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. *`tableName`*을 제공하여 테이블 이름을 구성합니다.
3. 사용자 지정 Azure SQL 속성에서 필요한 경우 파라미터와 값을 입력합니다.

고급 옵션

Azure SQL 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “Azure SQL 연결”를 참조하세요.

Blackbaud Raiser's Edge NXT에 연결

Blackbaud Raiser’s Edge NXT는 비영리 단체와 사회적 선행 커뮤니티를 위해 특별히 구축된 포괄적인 클라우드 기반 기부금 모금 및 기부자 관리 소프트웨어 솔루션입니다. 이 커넥터는 Blackbaud Raiser’s

Edge NXT의 SKY API를 기반으로 구축되었으며, Raisers Edge NXT 내 엔터티를 관리하는 데 도움이 되는 작업을 제공합니다.

주제

- AWS Glue의 Blackbaud Raiser's Edge NXT 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Blackbaud Raiser’s Edge NXT 구성
- Blackbaud Raiser’s Edge NXT 연결 구성
- Blackbaud Raiser’s Edge NXT 엔터티에서 읽기
- Blackbaud Raiser’s Edge NXT 연결 옵션
- Blackbaud Raiser’s Edge NXT 제한 사항

AWS Glue의 Blackbaud Raiser's Edge NXT 지원

AWS Glue는 다음과 같이 Blackbaud Raiser's Edge NXT를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Blackbaud Raiser's Edge NXT에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Blackbaud Raiser’s Edge NXT API 버전

다음 Blackbaud Raiser’s Edge NXT API 버전이 지원됩니다.

- v1

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
```

```
"Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Blackbaud Raiser's Edge NXT 구성

AWS Glue를 사용하여 Blackbaud Raiser's Edge NXT에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Blackbaud Raiser's Edge NXT 계정이 있습니다.
- API에 액세스하기 위해 적절한 읽기/쓰기 범위가 할당된 Blackbaud Raiser's Edge NXT 계정에서 액세스 토큰을 생성했습니다. 자세한 내용은 Authorization을 참조하세요.

이러한 요구 사항을 충족하면 Blackbaud Raiser's Edge NXT 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Blackbaud Raiser's Edge NXT 연결 구성

Blackbaud Raiser's Edge NXT는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Blackbaud Raiser's Edge NXT로 리디렉션합니다. 사용자가 로그인하고 Blackbaud Raiser's Edge NXT 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Blackbaud Raiser's Edge NXT에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID, 구독 키 및 인스턴스 URL을 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Blackbaud Raiser's Edge NXT로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 공개 Blackbaud Raiser's Edge NXT 설명서는 Authorization을 참조하세요.

다음 단계를 따라 Blackbaud Raiser's Edge NXT 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 *USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET*과 함께 연결된 앱 API를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 데이터 소스를 선택할 때 Blackbaud Raiser's Edge NXT를 선택합니다.
   b. 연결하려는 Blackbaud Raiser's Edge NXT 계정의 `INSTANCE_URL`을 제공합니다.
   c. 사용자 관리형 클라이언트 애플리케이션 `clientId`를 제공합니다.
   d. 계정과 연결된 구독 키를 입력합니다.

e. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

f. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

g. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Blackbaud Raiser's Edge NXT 엔터티에서 읽기

사전 조건

읽으려는 Blackbaud Raiser's Edge NXT 객체. 객체 이름이 필요합니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Constituent Address	예	예	아니요	예	예
Constituent Education	예	예	아니요	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Constituent Email address	예	예	아니요	예	예
Constituent Phone	예	예	아니요	예	예
Constituent Note	예	예	아니요	예	예
Constituent Relationship	예	예	아니요	예	예
Constituent Online presence	예	예	아니요	예	예
기회	예	예	아니요	예	예
Appeal	예	예	아니요	예	예
캠페인	예	예	아니요	예	예
Fund	예	예	아니요	예	예
패키지	예	예	아니요	예	예
Gift Batch	예	예	아니요	예	아니요
Event Participant	예	예	예	예	예
Constituent Fundraiser Assignment	아니요	아니요	아니요	예	아니요
Gift	예	예	예	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
멤버십	예	예	아니요	예	예
작업	예	예	아니요	예	아니요
Constituent	예	예	예	예	예
Constituent Goods	예	예	아니요	예	예
Event	예	예	예	예	예
Gift custom field	예	예	아니요	예	예

예시:

```
blackbaud_read = glueContext.create_dynamic_frame.from_options(
    connection_type="BLACKBAUD",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v1",
        "SUBSCRIPTION_KEY": <Subscription key associated with one's developer account>
    }
```

Blackbaud Raiser’s Edge NXT 엔터티 및 필드 세부 정보

엔터티 및 필드 세부 정보에 대한 자세한 내용은 다음을 참조하세요.

- 작업
- Constituent
- Constituent Address
- Constituent Membership
- Constituent Fundraiser Assignment
- Constituent Education

- Constituent Email Address
- Constituent Phone
- Constituent Note
- Constituent Online Presence
- Constituent Relationship
- Event
- Event Participant
- Appeal
- 캠페인
- Fund
- Package
- Gift
- Gift Custom Field
- Gift Batch
- Opportunity
- Constituent Codes

> **Note**
>
> 커넥터의 응답에서 Struct 및 List 데이터 유형은 String 데이터 유형으로 변환되며, DateTime 데이터 유형은 타임스탬프로 변환됩니다.

분할 쿼리

필드 기반 분할:

Blackbaud Raiser's Edge NXT는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.

레코드 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(`NUM_PARTITIONS`)을 제공할 수 있습니다. 이 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 `NUM_PARTITIONS`개의 하위 쿼리로 원본 쿼리가 분할됩니다.

레코드 기반 분할에서는 존재하는 총 레코드 수를 Blackbaud Raiser's Edge NXT API에서 쿼리하고 제공된 NUM_PARTITIONS 수로 나눕니다. 그런 다음, 결과 레코드 수를 각 하위 쿼리에서 동시에 가져옵니다.

- NUM_PARTITIONS: 파티션 수.

예시:

```
blackbaud_read = glueContext.create_dynamic_frame.from_options(
    connection_type="BLACKBAUD",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v1",
        "NUM_PARTITIONS": "2",
        "SUBSCRIPTION_KEY": <Subscription key associated with one's developer account>
    }
```

Blackbaud Raiser's Edge NXT 연결 옵션

다음은 Blackbaud Raiser's Edge NXT에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Blackbaud Raiser's Edge NXT에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Blackbaud Raiser's Edge NXT Rest API 버전.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수. 예시 값: 10.
- SUBSCRIPTION_KEY(문자열) - (필수) 기본값: 비어 있음. 읽기에 사용됩니다. 개발자 계정과 연결된 구독 키.

Blackbaud Raiser's Edge NXT 제한 사항

다음은 Blackbaud Raiser's Edge NXT에 대한 제한 사항 또는 참고 사항입니다.

- SaaS는 지정된 날짜 또는 그 이후에 생성되거나 수정된 결과를 반환하는 EQUAL_TO 연산자만 지원합니다. 추가로 'id' 필드는 문자열 데이터 유형입니다. 또한 null이 허용된 필드를 식별할 수 없습니다. 따라서 필드 기반 분할이 지원되지 않습니다.
- 증분 풀은 일별, 월별 및 주별 빈도가 있는 Event 엔터티에서만 지원됩니다.
- Constituent Fundraiser Assignment 엔터티는 최대 20개의 레코드를 반환합니다.
- 레코드 기반 분할:
  - Action, Constituent Fundraiser Assignment 또는 Gift Batch 엔터티에서는 지원되지 않습니다.
  - 필터 조건자를 사용한 레코드 기반 분할은 Event 및 Event Participant 엔터티에서만 지원됩니다. 필터 조건자를 다른 레코드 기반 지원 엔터티에 사용하는 경우 예외가 발생합니다.
- Gift Custom Field 엔터티에서 'value' 필드는 'category' 필드와 함께 사용해야 합니다. 그러지 않으면 필터링되지 않은 응답이 발생합니다. 따라서 사용자가 'value' 필드로 필터링하는 동안 'category' 필드를 연결하도록 강제하기 위해 앞서 언급한 요구 사항을 따르지 않은 경우 예외가 발생합니다.
- 적용 가능한 모든 엔터티의 date_added 및 last_modified 필드는 비교 연산자를 지원하지 않습니다. 같음 연산자만 지원합니다. 또한 앞서 언급한 필드와 페어링하여 레코드 범위를 제공할 수 있는 필드가 없습니다. 따라서 이러한 필드는 쿼리만 가능하고 증분 전송을 지원할 수 없습니다.
- Gift Batch 엔터티의 added_by 필드는 올바른 결과를 내보내지 못할 수 있으므로 필터링 가능한 것으로 간주되지 않습니다.
- Gift 엔터티에 데이터를 삽입할 때 /GET Gift List 엔드포인트를 통해 레코드를 검색하는 데 약 30분의 지연 시간이 있습니다.
- 데이터 소스의 종단에서 제한으로 인해 Gift 엔터티에 대한 증분 전송 지원이 중단되었습니다.
- Opportunity 엔터티의 status 필드에는 10분의 지연 시간이 있습니다.
- Fundraiser Assignment 엔터티에는 종속 엔터티로 Constituent가 있습니다. 커넥터는 응답 크기가 최대 허용 페이로드 크기를 초과하지 않도록 선택할 ID를 최대 5,000개까지 로드합니다.

CircleCI에 연결

CircleCI는 지속적 통합 및 지속적 전달 플랫폼입니다. CircleCI 계정에는 프로젝트, 파이프라인, 워크플로 등에 대한 데이터가 포함되어 있습니다. CircleCI 사용자인 경우 CircleCI 계정에 AWS Glue를 연결

할 수 있습니다. 그런 다음, CircleCI를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 CircleCI 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 CircleCI 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- CircleCI 구성
- CircleCI 연결 구성
- CircleCI 엔터티에서 읽기
- CircleCI 연결 옵션
- CircleCI 제한 사항

AWS Glue의 CircleCI 지원

AWS Glue는 다음과 같이 CircleCI를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 CircleCI에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 CircleCI API 버전

다음 CircleCI API 버전이 지원됩니다.

- v2

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
```

```
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

CircleCI 구성

AWS Glue를 사용하여 CircleCI에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- 전송하려는 데이터가 포함된 CircleCI 계정이 있습니다.
- 계정의 사용자 설정에서 개인 API 토큰을 생성했습니다. 자세한 내용은 Creating a personal API token을 참조하세요.
- 연결을 생성하는 동안 AWS Glue에 개인 API 토큰을 제공합니다.

이러한 요구 사항을 충족하면 CircleCI 계정에 AWS Glue를 연결할 준비가 된 것입니다.

CircleCI 연결 구성

CircleCI는 사용자 지정 인증을 지원합니다.

다음 단계를 따라 CircleCI 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `Circle-Token`과 함께 연결된 앱 API를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 데이터 소스를 선택할 때 CircleCI를 선택합니다.
   b. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   c. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   d. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

CircleCI 엔터티에서 읽기

사전 조건

읽으려는 CircleCI 객체. 객체 이름이 필요합니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
컨텍스트	예	아니요	아니요	예	아니요
Organization Summary Metric	예	아니요	아니요	예	아니요
파이프라인	아니요	아니요	아니요	예	아니요
파이프라인 워크플로	예	아니요	아니요	예	아니요
Project Branch	예	아니요	아니요	예	아니요
Project Flaky Test	아니요	아니요	아니요	예	아니요
Project Summary Metric	예	아니요	아니요	예	아니요
일정	아니요	아니요	아니요	예	아니요
Workflow Job Timeseries	예	아니요	아니요	예	아니요
Workflow Metric And Trend	예	아니요	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Workflow Recent Run	예	아니요	아니요	예	아니요
Workflow Summary Metric	예	아니요	아니요	예	아니요
Workflow Test Metric	예	아니요	아니요	예	아니요

예시:

```
circleci_read = glueContext.create_dynamic_frame.from_options(
    connection_type="circleci",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "context/e7ea2945-dccb-4205-b673-8391fe1b3a4c",
        "API_VERSION": "v2"
    }
```

CircleCI 엔터티 및 필드 세부 정보

엔터티 및 필드 세부 정보에 대한 자세한 내용은 다음을 참조하세요.

- Contexts
- Project Summary Metrics
- Workflow Job Timeseries
- Organization Summary Metrics
- Project Branches
- Project Flaky Tests
- Workflow Recent Runs
- Workflow Summary Metrics
- Workflow Metrics and Trends

- Workflow Test Metrics
- Pipelines
- Pipeline Workflows
- Schedules

정적 메타데이터를 포함하는 엔터티:

개체	필드	데이터 유형	지원되는 연산자
컨텍스트	Created At	String	
	ID	String	
	명칭	String	
	Owner Type	String	EQUAL_TO
Organization Summary Metric	All Projects	나열	
	Org Data	Struct	
	Org Project Data	나열	
	Project Names	String	EQUAL_TO
	Reporting Window	String	EQUAL_TO
파이프라인	브랜치	String	EQUAL_TO
	Created At	String	
	오류	나열	
	ID	String	
	숫자	Integer	
	Project Slug	String	
	State	String	

개체	필드	데이터 유형	지원되는 연산자
	트리거	Struct	
	Trigger Parameters	Struct	
	Updated At	String	
	VCS	Struct	
파이프라인 워크플로	Canceled By	String	
	Created At	String	
	Errorer By	String	
	ID	String	
	명칭	String	
	파이프라인 ID	String	
	Pipeline Number	Integer	
	Project Slug	String	
	Started By	String	
	상태 표시기	String	
	Stopped At	String	
	태그	String	
Project Branch	브랜치	나열	
	Org ID	String	
	Project ID	String	
	워크플로 이름	String	EQUAL_TO

개체	필드	데이터 유형	지원되는 연산자
Project Flaky Test	Classname	String	
	파일	String	
	Job Name	String	
	Job Number	Integer	
	Pipeline Number	Integer	
	소스	String	
	Test Name	String	
	Time Wasted	Integer	
	Times Flaked	Integer	
	Workflow Created At	String	
	워크플로 ID	String	
	워크플로 이름	String	
Project Summary Metric	All Branches	나열	
	All Workflows	나열	
	브랜치	String	EQUAL_TO
	조직 ID	String	
	Project Data	Struct	
	Project ID	String	
	Project Workflow Branch Data	나열	
	Project Workflow Data	나열	

개체	필드	데이터 유형	지원되는 연산자
	Reporting Window	String	EQUAL_TO
	Workflow Names	String	EQUAL_TO
일정	액터	Struct	
	Created At	String	
	설명	String	
	ID	String	
	명칭	String	
	파라미터	Struct	
	Project Slug	String	
	Timetable	Struct	
	Updated At	String	
Workflow Job Timeseries	브랜치	String	EQUAL_TO
	세부 수준	String	EQUAL_TO
	Max Ended At	String	
	Metrics	Struct	
	Min Started At	String	
	명칭	String	
	Start End Date	DateTime	EQUAL_TO, BETWEEN
	Timestamp	String	

개체	필드	데이터 유형	지원되는 연산자
Workflow Metric and Trend	All Branches	불	EQUAL_TO
	브랜치	String	EQUAL_TO
	Metrics	Struct	
	Trends	Struct	
	Workflow Names	나열	
Workflow Recent Run	All Brances	불	EQUAL_TO
	브랜치	String	EQUAL_TO
	Created At	String	
	Credits Used	Integer	
	지속 시간	Integer	
	ID	String	
	Is Approval	불	
	Start End Date	DateTime	EQUAL_TO, BETWEEN
	상태 표시기	String	
	Stopped At	String	
Workflow Summary Metric	All Branches	불	EQUAL_TO
	브랜치	String	EQUAL_TO
	Metrics	Struct	
	명칭	String	
	Project ID	String	

개체	필드	데이터 유형	지원되는 연산자
	Reporting Window	String	EQUAL_TO
	Window End	String	
	Window Start	String	
Workflow Test Metric	Average Test Count	Integer	
	브랜치	String	EQUAL_TO
	Most Failed Tests	나열	
	Most Failed Tests Extra	Integer	
	Slowest Tests	나열	
	Slowest Tests Extra	Integer	
	Test Runs	나열	
	Total Test Runs	Integer	

> **Note**
> 구조체 및 목록 데이터 유형은 커넥터의 응답에서 문자열 데이터 유형으로 변환됩니다.

분할 쿼리

CircleCI는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.

CircleCI 연결 옵션

다음은 CircleCI의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. CircleCI에서의 객체 이름.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 CircleCI Rest API 버전.

- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

CircleCI 제한 사항

다음은 CircleCI의 제한 사항 또는 참고 사항입니다.

- CircleCI는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.
- '-'(하이픈)이 포함된 필터 필드는 백틱 내에 래핑된 경우에만 작동합니다. 예: `workflow-name` = "abc"
- GitLab VCS 엔터티 경로에 필요한 'Project ID'를 검색할 프로그래밍 방법이 없으므로 GitLab VCS 유형을 지원할 수 없습니다.

Datadog에 연결

Datadog은 인프라, 애플리케이션, 서비스, 도구 등 클라우드 규모의 애플리케이션을 위한 모니터링 및 분석 플랫폼입니다.

주제

- AWS Glue의 Datadog 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Datadog 구성
- Datadog 연결 구성
- Datadog 엔터티에서 읽기
- Datadog 연결 옵션
- Datadog 계정 생성
- 제한 사항

AWS Glue의 Datadog 지원

AWS Glue에서는 다음과 같이 Datadog을 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Datadog에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Datadog API 버전

- v1
- v2

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절

차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess-정책이 연결된 자격 증명이 AWS Management 콘솔을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Datadog 구성

AWS Glue를 사용하여 Datadog에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- DD-API-KEY 및 DD-APPLICATION-KEY가 있는 Datadog 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Datadog 계정 생성을 참조하세요.
- Datadog 계정에 유효한 라이선스가 있는 API 액세스 권한이 있습니다.

Datadog은 다음 6개의 URL을 지원합니다. 모든 Datadog API 클라이언트는 기본적으로 Datadog US1 사이트 API를 사용하도록 구성됩니다. Datadog EU 사이트에 있는 경우 API에 액세스하려면 Datadog EU 사이트의 `DD-API-KEY` 및 `DD-APPLICATION-KEY`가 있는 https://api.datadoghq.eu URL을 선택해야 합니다. 마찬가지로 다른 사이트의 경우 해당 사이트의 `DD-API-KEY and DD-APPLICATION-KEY`를 사용하여 해당 URL을 선택해야 합니다.

- US1 API URL-https://api.datadoghq.comhttps://api.datadoghq.com
- EU API URL-https://api.datadoghq.eu
- US3 API URL-https://api.us3.datadoghq.com
- US5 API URL-https://api.us5.datadoghq.com
- S1-FED API URL-https://api.ddog-gov.com
- 일본 API URL-https://api.ap1.datadoghq.com

이러한 요구 사항을 충족하면 Datadog 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Datadog 연결 구성

Datadog은 사용자 지정 인증을 지원합니다. 다음은 Datadog 연결을 구성하는 단계입니다.

Datadog 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.

   고객 관리형 연결된 앱의 경우-보안 암호는 키 역할을 하는 **API_KEY** 및 **APPLICATION_KEY**와 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

   > **Note**
   >
   > AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.

   a. 연결 유형을 선택할 때 Datadog를 선택합니다.

   b. 연결하려는 Datadog의 `Instance_Url`을 제공합니다.

   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

   ```
   {
   "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "secretsmanager:DescribeSecret",
                   "secretsmanager:GetSecretValue",
                   "secretsmanager:PutSecretValue",
                   "ec2:CreateNetworkInterface",
                   "ec2:DescribeNetworkInterface",
                   "ec2:DeleteNetworkInterface",
               ],
               "Resource": "*"
           }
       ]
   }
   ```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Datadog 엔터티에서 읽기

사전 조건

읽으려는 Datadog 객체입니다. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Metrics Timeseries	예	아니요	아니요	예	아니요
Log Queries	예	예	예	예	아니요

예

```
Datadog_read = glueContext.create_dynamic_frame.from_options(
    connection_type="datadog",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "log-queries",
        "API_VERSION": "v2",
        "INSTANCE_URL": "https://api.datadoghq.com",
        "FILTER_PREDICATE": "from = `2023-10-03T09:00:26Z`"
    }
```

Datadog 엔터티 및 필드 세부 정보:

엔터티	필드	데이터 형식	지원되는 연산자
Metrics Timeseries	error	String	NA
	aggr	String	NA
	attributes	Struct	NA
	display_name	String	NA

엔터티	필드	데이터 형식	지원되는 연산자
	end	DateTime	NA
	expression	String	NA
	interval	Integer	NA
	length	Integer	NA
	metric	String	NA
	pointlist	List	NA
	query_index	Integer	NA
	scope	String	NA
	start	DateTime	NA
	tag_set	List	NA
	unit	Struct	NA
	from_to_date	DateTime	BETWEEN
	query	String	EQUAL_TO
	status	String	NA
	type	String	NA
	host	String	NA
Log Queries	id	String	NA
	attributes	Struct	NA
	timestamp	DateTime	NA
	type	String	NA

엔터티	필드	데이터 형식	지원되는 연산자
	from	DateTime	BETWEEN,EQUAL_TO
	indexes	List	EQUAL_TO
	storage_tier	String	EQUAL_TO
	query	String	EQUAL_TO

Datadog 연결 옵션

다음은 Datadog의 연결 옵션입니다.

- `ENTITY_NAME`(문자열)-(필수) 읽기/쓰기에 사용됩니다. Datadog에서의 객체 이름입니다.
- `API_VERSION`(문자열)-(필수) 읽기/쓰기에 사용됩니다. 사용하려는 Datadog Rest API 버전입니다. `v1` 버전은 `metrics-timeseries` 엔터티를 지원하는 반면 `v2` 버전은 `log-queries` 엔터티를 지원합니다.
- `INSTANCE_URL`(문자열)-(필수) 읽기에 사용됩니다. Datadog 인스턴스 URL입니다. Datadog 인스턴스 URL은 리전마다 다릅니다.
- `SELECTED_FIELDS`(List<String>)-기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Datadog 계정 생성

1. https://www.datadoghq.com/으로 이동합니다.
2. 무료로 시작을 선택합니다.
3. 필요한 정보를 입력하고 가입합니다.
4. 제안대로 Datadog 에이전트 설치 프로그램을 설치합니다.
5. 계정이 활성 Datadog 구독이 있는 유효한 조직(사용 가능한 목록에서)에 등록되었는지 확인합니다.

6. Datadog 계정에 로그인한 후 오른쪽 상단 모서리의 사용자 이름 위로 마우스를 가져가 키 세부 정보를 확인합니다.

   a. API 키를 가져오려면 API 키를 선택합니다.

   b. 애플리케이션 키를 가져오려면 애플리케이션 키를 선택합니다.

제한 사항

다음은 Datadog 커넥터의 제한 사항입니다.

- Datadog은 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.
- `from`은 `Log Queries` 엔터티의 필수 필터 파라미터입니다.
- `from_to_date` 및 `query`는 `Metrics Timeseries` 엔터티를 위한 필수 필터 파라미터입니다.

Docusign Monitor에 연결

Docusign Monitor는 24시간 활동 추적을 통해 조직이 계약을 보호하는 데 도움을 줍니다. Monitor API는 이 활동 추적 정보를 기존 보안 스택 또는 데이터 시각화 도구에 직접 제공하고, 팀은 이를 활용하여 무단 활동을 감지하고, 인시던트를 조사하고, 확인된 위협에 신속하게 대응할 수 있습니다. 또한 보안 팀은 특정 비즈니스 요구 사항에 맞게 대시보드와 알림을 유연하게 사용자 지정할 수 있습니다.

주제

- AWS Glue의 Docusign Monitor 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Docusign Monitor 구성
- Docusign Monitor 연결 구성
- Docusign Monitor 엔터티에서 읽기
- Docusign Monitor 연결 옵션
- Docusign Monitor 제한 사항

AWS Glue의 Docusign Monitor 지원

AWS Glue는 다음과 같이 Docusign Monitor를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Docusign Monitor에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Docusign Monitor API 버전

다음 Docusign Monitor API 버전이 지원됩니다.

- v2.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Docusign Monitor 구성

AWS Glue를 사용하여 Docusign Monitor에서 지원되는 대상으로 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Docusign Monitor에서 Docusign Software 제품을 사용하는 Docusign 계정이 있습니다.
- Docusign 계정의 개발자 콘솔에서 AWS Glue용 OAuth 2.0 통합 앱을 생성했습니다.

  이 앱에서는 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하는 데 사용하는 클라이언트 자격 증명을 제공합니다. 자세한 내용은 Docusign Monitor 설명서의 OAuth 2.0을 참조하세요.

이러한 요구 사항을 충족하면 Docusign Monitor 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Docusign Monitor 연결 구성

Docusign Monitor는 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 3각 OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.
- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Docusign Monitor에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Docusign Monitor로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 공개 Docusign Monitor 설명서는 OAuth for Docusign Connect를 참조하세요.

다음 단계를 따라 Docusign Monitor 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 API를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Docusign Monitor를 선택합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. Docusign Monitor 앱의 사용자 관리형 클라이언트 애플리케이션 ClientId를 입력합니다.
   e. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   f. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Docusign Monitor 엔터티에서 읽기

사전 조건

읽으려는 Docusign Monitor 객체.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
데이터 모니터링	예	예	아니요	예	아니요

예시:

```
docusignmonitor_read = glueContext.create_dynamic_frame.from_options(
    connection_type="docusign_monitor",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "monitoring-data",
        "API_VERSION": "v2.0"
    }
```

Docusign Monitor 엔터티 및 필드 세부 정보

정적 메타데이터를 포함하는 엔터티:

개체	필드	데이터 유형	지원되는 연산자
데이터 모니터링	타임스탬프	DateTime	N/A
	eventId	String	해당 사항 없음
	애플리케이션	String	해당 사항 없음
	환경	String	해당 사항 없음
	site	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	traceToken	String	해당 사항 없음
	organizationId	String	해당 사항 없음
	accountId	String	해당 사항 없음
	userId	String	해당 사항 없음
	객체	String	해당 사항 없음
	작업	String	해당 사항 없음
	property	String	해당 사항 없음
	필드	String	해당 사항 없음
	결과	String	해당 사항 없음
	IntegratorKey	String	해당 사항 없음
	customerVisible	String	해당 사항 없음
	version	String	해당 사항 없음
	userAgent	String	해당 사항 없음
	userAgentClientInfo	Struct	N/A
	ipAddress	String	해당 사항 없음
	ipAddressLocation	Struct	N/A
	data	String	해당 사항 없음
	source	String	해당 사항 없음
	latitude	배정밀도 실수	N/A
	longitude	배정밀도 실수	N/A

개체	필드	데이터 유형	지원되는 연산자
	city	String	해당 사항 없음
	state	String	해당 사항 없음
	country	String	해당 사항 없음
	usUserMemberOfDomain	불	N/A
	affectedUserIsMemberOfDomain	불	N/A
	proxyStatus	String	해당 사항 없음
	proxyType	String	해당 사항 없음
	proxyLevel	String	해당 사항 없음
	referencedUserId	String	해당 사항 없음
	device	String	해당 사항 없음
	브라우저	String	해당 사항 없음
	cursor	DateTime	EQUAL_TO

분할 쿼리

Docusign Monitor는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.

Docusign Monitor 연결 옵션

다음은 Docusign Monitor의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Docusign Monitor에서의 객체 이름.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Docusign Monitor Rest API 버전.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.

- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.

Docusign Monitor 제한 사항

다음은 Docusign Monitor의 제한 사항 또는 참고 사항입니다.

- cursor 필드를 사용하여 필터가 적용되면 API는 지정된 날짜부터 시작하여 다음 7일 동안의 레코드를 검색합니다.
- 필터가 제공되지 않으면 API는 API 요청의 현재 날짜로부터 이전 7일 동안의 레코드를 검색합니다.
- Docusign Monitor는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.
- Docusign Monitor는 순서 기준 특성을 지원하지 않습니다.

Domo에 연결

Domo는 클라우드 기반 대시보딩 도구입니다. Domo의 엔터프라이즈 애플리케이션 플랫폼을 사용하면 Domo를 확장하는 데 필요한 기반이 마련되므로 사용자 지정 솔루션을 더욱 빠르게 구축할 수 있습니다.

주제

- AWS Glue의 Domo 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Domo 구성
- Domo 연결 구성
- Domo 엔터티에서 읽기
- Domo 연결 옵션
- Domo 제한 사항

AWS Glue의 Domo 지원

AWS Glue에서는 다음과 같이 Domo를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Domo에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Domo API 버전

다음 Domo API 버전이 지원됩니다.

- v1

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Domo 구성

AWS Glue를 사용하여 Domo에서 지원되는 대상으로 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Domo 계정에서 API 액세스가 활성화되어 있습니다.
- 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하기 위해 사용하는 클라이언트 자격 증명을 제공하는 앱이 Domo 개발자 계정 아래에 있습니다. 자세한 내용은 Domo 개발자 앱 생성 단원을 참조하십시오.

이러한 요구 사항을 충족하면 Domo 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Domo 개발자 앱 생성

클라이언트 ID 및 클라이언트 보안 암호를 가져오려면 개발자 계정을 생성합니다.

1. Domo 개발자 로그인 페이지로 이동합니다.
2. 로그인을 선택합니다.
3. 도메인 이름을 입력하고 Continue를 클릭합니다.
4. My Account로 마우스를 가리킨 다음 New Client를 선택합니다.
5. 이름 및 설명을 입력하고 범위('data')를 선택한 다음 Create를 선택합니다.
6. 생성된 새 클라이언트에서 생성된 클라이언트 ID와 클라이언트 보안 암호를 가져옵니다.

Domo 연결 구성

Domo는 OAuth2에 대한 CLIENT CREDENTIALS 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 클라이언트 애플리케이션만 서버에 인증하고, 사용자는 관여하지 않으므로 2각 OAuth로 간주됩니다.

- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Domo에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 공개 Domo 설명서는 OAuth Authentication을 참조하세요.

다음 단계를 따라 Domo 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 앱 액세스 토큰, `client_id` 및 `client_secret`을 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Domo를 선택합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Domo 엔터티에서 읽기

사전 조건

읽으려는 Domo 객체. 객체 이름(예: Data Set 또는 Data Permission Policies)이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
데이터 세트	예	예	예	예	예
Data Permission Policies	아니요	아니요	아니요	예	아니요

예시:

```
Domo_read = glueContext.create_dynamic_frame.from_options(
    connection_type="domo",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "dataset",
        "API_VERSION": "v1"
    }
```

Domo 엔터티 및 필드 세부 정보

정적 메타데이터를 포함하는 엔터티:

개체	필드	데이터 유형	지원되는 연산자
Data Permission Policies	id	Long	N/A
	type	String	해당 사항 없음
	name	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	filters	나열	N/A
	사용자	나열	N/A
	virtualUsers	나열	N/A
	그룹	나열	N/A

다음 엔터티에 대해 Domo에서는 메타데이터를 동적으로 가져도록 엔드포인트를 제공하므로 운영자 지원이 엔터티의 데이터 유형 수준에서 캡처됩니다.

개체	데이터 유형	지원되는 연산자
데이터 세트	Integer	=, !=, <, >, >=, <=
	Long	=, !=, <, >, >=, <=
	String	=, !=, CONTAINS
	날짜	=, >, >=, <, <=, BETWEEN
	DateTime	=, >, >=, <, <=, BETWEEN
	Boolean	=, !=
	배정밀도 실수	=, !=, <, >, >=, <=
	나열	N/A
	Struct	N/A

분할 쿼리

필드 기반 분할

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  DateTime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2023-01-15T11:18:39.205Z"
  ```

  Date 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2023-01-15"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).

  유효한 값의 예제:

  ```
  "2023-02-15T11:18:39.205Z"
  ```

- NUM_PARTITIONS: 파티션 수.

엔터티 수준 분할 필드 지원 세부 정보는 다음 표에 나와 있습니다.

엔터티 이름	분할 필드	데이터 유형
데이터세트	모든 날짜/시간 기반 필드[동적 메타데이터]	DateTime
	모든 날짜 기반 필드[동적 메타데이터]	날짜

예시:

```
Domo_read = glueContext.create_dynamic_frame.from_options(
    connection_type="domo",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "dataset",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "permissionTime"
        "LOWER_BOUND": "2023-01-15T11:18:39.205Z"
        "UPPER_BOUND": "2023-02-15T11:18:39.205Z"
        "NUM_PARTITIONS": "2"
    }
```

레코드 기반 분할

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(NUM_PARTITIONS)을 제공할 수 있습니다. 이 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

레코드 기반 분할에서는 존재하는 총 레코드 수를 Domo에서 쿼리하고 제공된 NUM_PARTITIONS 수로 나눕니다. 그런 다음, 결과 레코드 수를 각 하위 쿼리에서 동시에 가져옵니다.

예시:

```
Domo_read = glueContext.create_dynamic_frame.from_options(
    connection_type="domo",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "dataset",
        "API_VERSION": "v1",
        "NUM_PARTITIONS": "2"
    }
```

Domo 연결 옵션

다음은 Domo의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Domo에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Domo Rest API 버전.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.

- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Domo 제한 사항

다음은 Domo의 제한 사항 또는 참고 사항입니다.

- SDK 제한으로 인해 '_'로 시작하는 쿼리 가능한 필드(예: _BATCH_ID)에 대해 필터링이 예상대로 작동하지 않습니다.
- API 제한으로 인해 입력한 날짜 이전의 날짜에 필터링이 작동합니다. 이는 증분 풀에도 영향을 미칩니다. 이 제한을 해결하려면 UTC 시간대에 따라 날짜를 선택하여 필요한 날짜에 대한 데이터를 가져옵니다.

Dynatrace에 연결

Dynatrace는 포괄적인 관찰성과 보안을 위한 분석 및 자동화를 제공하는 플랫폼입니다. 애플리케이션 성능, 인프라 및 사용자 경험을 모니터링 및 최적화하는 데 특화되어 있습니다.

주제

- AWS Glue의 Dynatrace 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Dynatrace 구성
- Dynatrace 연결 구성
- Dynatrace 엔터티에서 읽기
- Dynatrace 연결 옵션
- Dynatrace 제한 사항

AWS Glue의 Dynatrace 지원

AWS Glue는 다음과 같이 Dynatrace를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Dynatrace에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Dynatrace API 버전

다음 Dynatrace API 버전이 지원됩니다.

- v2

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
     "Statement": [
      {
             "Effect": "Allow",
             "Action": [
                 "glue:ListConnectionTypes",
                 "glue:DescribeConnectionType",
                 "glue:RefreshOAuth2Tokens",
                 "glue:ListEntities",
                 "glue:DescribeEntity"
             ],
             "Resource": "*"
         }
     ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Dynatrace 구성

AWS Glue를 사용하여 Dynatrace에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Dynatrace 계정이 있습니다.
- API에 액세스하기 위해 적절한 읽기/쓰기 범위가 할당된 Dynatrace 계정에서 액세스 토큰을 생성했습니다. 자세한 내용은 키 생성을 참조하세요.

이러한 요구 사항을 충족하면 Dynatrace 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Dynatrace 연결 구성

Dynatrace는 사용자 지정 인증을 지원합니다.

다음 단계를 따라 Dynatrace 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 *apiToken*과 함께 연결된 앱 API를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 데이터 소스를 선택할 때 Dynatrace를 선택합니다.

b. 연결하려는 Dynatrace 계정의 INSTANCE_URL을 제공합니다.

c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 secretName을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 secretName을 읽을 수 있는 권한을 부여합니다.

Dynatrace 엔터티에서 읽기

사전 조건

읽으려는 Dynatrace 객체. 객체 이름(예: ‘problem’)이 필요합니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
문제	예	예	예	예	아니요

예시:

```
Dynatrace_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Dynatrace",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "problem",
        "API_VERSION": "v2",
        "INSTANCE_URL": "https://[instanceName].live.dynatrace.com"
    }
```

Dynatrace 엔터티 및 필드 세부 정보:

Dynatrace에서는 지원되는 엔터티에 대해 메타데이터를 동적으로 가져오도록 엔드포인트를 제공합니다. 따라서 운영자 지원은 데이터 유형 수준에서 캡처됩니다.

개체	필드	데이터 유형	지원되는 연산자
문제	affectedEntities	나열	EQUAL_TO
	displayId	String	EQUAL_TO
	endTime	DateTime	
	entityTags	나열	
	evidenceDetails	Struct	
	impactAnalysis	Struct	
	impactLevel	String	EQUAL_TO
	impactedEntities	나열	EQUAL_TO
	linkedProblemInfo	Struct	
	managementZones	나열	EQUAL_TO
	problemFilters	나열	
	recentComments	Struct	
	rootCauseEntity	Struct	EQUAL_TO

개체	필드	데이터 유형	지원되는 연산자
	problemId	String	EQUAL_TO
	severityLevel	String	EQUAL_TO
	startTime	DateTime	BETWEEN
	status	String	EQUAL_TO
	제목	String	
	from	DateTime	EQUAL_TO, BETWEEN
	problemFilterIds	String	EQUAL_TO
	problemFilterNames	String	EQUAL_TO
	managementZoneIds	String	EQUAL_TO
	텍스트	String	EQUAL_TO
	underMaintenance	불	EQUAL_TO
	message	String	

분할 쿼리

Dynatrace는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.

Dynatrace 연결 옵션

다음은 Dynatrace의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Dynatrace에서의 객체 이름.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Dynatrace Rest API 버전.
- `INSTANCE_URL`(문자열) - 읽기에 사용됩니다. 유효한 Dynatrace 인스턴스 URL.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.

- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Dynatrace 제한 사항

다음은 Dynatrace의 제한 사항 또는 참고 사항입니다.

- Dynatrace는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.
- Select All 특성의 경우 필터에 'field'를 입력하면 페이지당 레코드가 10개를 초과할 수 없습니다.
- 지원되는 최대 페이지 크기는 500입니다. 흐름을 생성하는 동안 [evidenceDetails, impactAnalysis, recentComments] 필드 중 하나를 선택하면 페이지당 레코드는 기본적으로 10으로 설정됩니다.

Facebook Ads에 연결

Facebook Ads는 모든 규모의 비즈니스에서 대상 고객에게 도달하고 다양한 마케팅 목표를 달성하기 위해 사용하는 강력한 디지털 광고 플랫폼입니다. 이 플랫폼을 통해 광고주는 Facebook 및 메신저를 포함한 Facebook의 앱 및 서비스 제품군에 표시할 수 있는 맞춤형 광고를 생성할 수 있습니다. Facebook Ads는 고급 타겟팅 기능을 통해 비즈니스가 특정 인구 통계, 관심사, 행동 및 위치에 도달할 수 있도록 지원합니다.

주제

- Facebook Ads에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Facebook Ads 구성
- Facebook Ads 연결 구성
- Facebook Ads 엔터티에서 읽기
- Facebook Ads 연결 옵션
- Facebook Ads 커넥터에 대한 제한 사항 및 참고 사항

Facebook Ads에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Facebook Ads를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Facebook Ads에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Facebook Ads API 버전

다음 Facebook Ads API 버전이 지원됩니다.

- v17.0
- v18.0
- v19.0
- v20.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Facebook Ads 구성

AWS Glue를 사용하여 Facebook Ads에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Facebook Standard 계정에는 Facebook을 통해 직접 액세스됩니다.
- 액세스 토큰을 생성하려면 사용자 인증이 필요합니다.
- Facebook Ads SDK 커넥터는 사용자 액세스 토큰 OAuth 흐름을 구현합니다.
- OAuth2.0을 사용하여 Facebook Ads에 대한 API 요청을 인증합니다. 이 웹 기반 인증은 2FA의 상위 세트인 다중 인증(MFA) 아키텍처에 속합니다.
- 사용자에게 엔드포인트에 액세스할 수 있는 권한을 부여해야 합니다. 사용자의 데이터에 액세스하기 위해 엔드포인트 권한 부여는 권한 및 기능을 통해 처리됩니다.

OAuth 2.0 자격 증명 가져오기

인스턴스에 대해 인증된 직접 호출을 수행할 수 있도록 API 자격 증명을 확보하려면 Facebook Ads 개발자 안내서의 REST API를 참조하세요.

Facebook Ads 연결 구성

Facebook Ads에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 3각 OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.

- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Facebook Ads에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Facebook Ads로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 액세스 토큰을 생성합니다. 만료되는 시스템 사용자 토큰은 생성 날짜 또는 새로 고친 날짜로부터 60일 동안 유효합니다. 연속성을 생성하려면 개발자가 60일 이내에 액세스 토큰을 새로 고쳐야 합니다. 그렇지 않으면 액세스 토큰이 몰수되고 개발자가 API 액세스 권한을 다시 획득하기 위해 새 토큰을 확보해야 합니다. 액세스 토큰 새로 고침을 참조하세요.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Facebook Ads 설명서는 Google for Developers 안내서의 Using OAuth 2.0 to Access Google APIs를 참조하세요.

Facebook Ads 연결을 구성하는 방법:

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Facebook Ads를 선택하세요.
   b. 연결하려는 Facebook Ads 인스턴스의 `INSTANCE_URL`을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Facebook Ads 엔터티에서 읽기

사전 조건

읽으려는 Facebook Ads 객체. 객체 이름이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Campaign	예	예	아니요	예	예
광고 세트	예	예	아니요	예	예
광고	예	예	아니요	예	예
광고 크리에이티브	아니요	예	아니요	예	아니요
인사이트 - 계정	아니요	예	아니요	예	아니요
Adaccounts	예	예	아니요	예	아니요
인사이트 - 광고	예	예	아니요	예	예
인사이트 - 광고 세트	예	예	아니요	예	예
인사이트 - 캠페인	예	예	아니요	예	예

예제:

```
FacebookAds_read = glueContext.create_dynamic_frame.from_options(
```

```
    connection_type="FacebookAds",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v20.0"
    }
```

Facebook Ads 엔터티 및 필드 세부 정보

엔터티 및 필드 세부 정보에 대한 자세한 내용은 다음을 참조하세요.

- 광고 계정
- 캠페인
- 광고 세트
- 광고
- 광고 크리에이티브
- 인사이트 광고 계정
- 인사이트 광고
- 인사이트 광고 세트
- 인사이트 캠페인

자세한 내용은 마케팅 API를 참조하세요.

> **Note**
>
> 구조체 및 목록 데이터 유형은 커넥터의 응답에서 문자열 데이터 유형으로 변환됩니다.

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리를 파티셔닝하는 데 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

DateTime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

유효한 값의 예제:

```
"2022-01-01"
```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예제:

```
FacebookADs_read = glueContext.create_dynamic_frame.from_options(
    connection_type="FacebookAds",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v20.0",
        "PARTITION_FIELD": "created_time"
        "LOWER_BOUND": "2022-01-01"
        "UPPER_BOUND": "2024-01-02"
        "NUM_PARTITIONS": "10"
    }
```

Facebook Ads 연결 옵션

다음은 Facebook Ads에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Facebook Ads에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Facebook Ads Rest API 버전. 예: v1.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리를 파티셔닝하는 데 사용할 필드.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).

- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- TRANSFER_MODE(문자열) - 기본값: SYNC. 비동기 읽기에 사용됩니다.

Facebook Ads 커넥터에 대한 제한 사항 및 참고 사항

다음은 Facebook Ads 커넥터에 대한 제한 사항 또는 참고 사항입니다.

- Facebook Ads는 동적 메타데이터를 지원하므로 모든 필드를 쿼리할 수 있습니다. 모든 필드는 필터링을 지원하며 데이터를 사용할 수 있는 경우 레코드를 가져옵니다. 그렇지 않으면 Facebook에서 적절한 오류 메시지와 함께 잘못된 요청(400) 응답을 반환합니다.
- 앱의 직접 호출 수는 롤링 1시간 기간에 사용자가 수행할 수 있는 직접 호출 수(200에 사용자 수를 곱한 값). 요금 제한에 대한 자세한 내용은 사용 제한 및 비즈니스 사용 사례 사용 제한을 참조하세요.

Facebook Page Insights에 연결

Facebook Pages를 사용하면 회사 및 기타 관심 그룹이 Facebook.com 소셜 네트워크에 대한 페이지를 생성할 수 있습니다. 기업은 이러한 페이지를 사용하여 온라인에서 영업 시간을 공유하고, 공지를 하며, 고객과 소통합니다. Facebook Page Insights 사용자인 경우 Facebook Page Insights 계정에 AWS Glue를 연결할 수 있습니다. Facebook Page Insights를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Facebook Page Insights에서 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송합니다.

주제

- AWS Glue의 Facebook Page Insights 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Facebook Page Insights 구성
- Facebook Page Insights 연결 구성
- Facebook Page Insights 엔터티에서 읽기
- Facebook Page Insights 연결 옵션
- Facebook Page Insights 커넥터에 대한 제한 사항 및 참고 사항

AWS Glue의 Facebook Page Insights 지원

AWS Glue에서는 다음과 같이 Facebook Page Insights를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Facebook Page Insights에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Facebook Page Insights API 버전

다음 Facebook Page Insights API 버전이 지원됩니다.

- v17
- v18
- v19
- v20
- v21

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Facebook Page Insights 구성

AWS Glue를 사용하여 Facebook Page Insights에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Facebook Standard 계정에는 Facebook을 통해 직접 액세스됩니다.
- 액세스 토큰을 생성하려면 사용자 인증이 필요합니다.
- Facebook Page Insights 커넥터는 사용자 액세스 토큰 OAuth 흐름을 구현합니다.
- 커넥터는 OAuth2.0을 사용하여 Facebook Page Insights에 대한 API 요청을 인증합니다. 이는 2FA의 상위 세트인 다중 인증(MFA) 아키텍처에 속합니다. 웹 기반 인증입니다.
- 사용자에게 엔드포인트에 액세스할 수 있는 권한을 부여해야 합니다. 사용자의 데이터에 액세스하기 위해 엔드포인트 권한 부여는 권한 및 기능을 통해 처리됩니다.

Facebook Page Insights 연결 구성

Facebook Ads 연결을 구성하는 방법:

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Facebook Page Insights를 선택하세요.
   b. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
```

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

c. 권한 부여 코드 URL을 선택합니다.

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Facebook Page Insights 엔터티에서 읽기

사전 조건

읽으려는 Facebook Ads 객체입니다. 객체 이름이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Page Content	예	아니요	예	예	예
Page CTA Clicks	예	아니요	아니요	예	예
Page Engagement	예	아니요	아니요	예	예

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Page Impressions	예	아니요	아니요	예	예
Page Posts	예	아니요	아니요	예	예
Page Post Engagement	아니요	아니요	아니요	예	아니요
Page Post Reactions	아니요	아니요	아니요	예	아니요
Page Reactions	예	아니요	아니요	예	예
Stories	예	아니요	아니요	예	예
Page User Demographics	예	아니요	아니요	예	예
Page Video Views	예	아니요	아니요	예	예
Page Views	예	아니요	아니요	예	예
Page Video Posts	예	아니요	아니요	예	예
Pages	아니요	예	아니요	예	아니요
Feeds	예	예	아니요	예	예

예시:

```
facebookPageInsights_read = glueContext.create_dynamic_frame. from options(
    connection_type="facebookpageinsights",
    connection_options={
```

```
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v21"
    }
```

Facebook Page Insights 필드 세부 정보:

엔터티	필드	데이터 유형	지원되는 연산자
Page Content	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page CTA Clicks	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
Page Engagement	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page Impressions	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page Posts	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO

엔터티	필드	데이터 유형	지원되는 연산자
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page Post Engagement	Name	String	N/A
	Period	Period	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page Post Reactions	Name	String	N/A
	Period	Period	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
Page User Demographics	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page Video Views	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page Views	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO

엔터티	필드	데이터 유형	지원되는 연산자
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Page Video Posts	Name	String	N/A
	Period	Period	EQUAL_TO
	Since	DateTime	EQUAL_TO
	Values	List	N/A
	Title	String	N/A
	Description	String	N/A
	description_from_api_doc	String	N/A
	Id	String	N/A
Pages	Name	String	N/A
	About	String	N/A
	access_token	String	N/A
	ad_campaign	String	N/A
	Affiliation	String	N/A
	app_id	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	artists_we_like	String	N/A
	Attire	String	N/A
	Awards	String	N/A
	band_interests	String	N/A
	band_members	String	N/A
	best_page	String	N/A
	Bio	String	N/A
	Birthday	String	N/A
	booking_agent	String	N/A
	Built	String	N/A
	can_checkin	String	N/A
	can_post	String	N/A
	Category	String	N/A
	category_list	List	N/A
	Checkins	Integer	N/A
	company_overview	String	N/A
	connected_instagram_account	String	N/A
	contact_address	String	N/A
	country_page_likes	Integer	N/A
	Cover	Struct	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	culinary_team	String	N/A
	current_location	String	N/A
	delivery_and_pickup_option_info	List	N/A
	Description	String	N/A
	description_html	String	N/A
	differently_open_offerings	List	N/A
	directed_by	String	N/A
	display_subtext	String	N/A
	displayed_message_response_time	String	N/A
	Emails	String	N/A
	Engagement	String	N/A
	fan_count	Integer	N/A
	featured_video	String	N/A
	Features	String	N/A
	followers_count	Integer	N/A
	food_styles	List	N/A
	Founded	String	N/A
	general_info	String	N/A
	general_manager	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	Genre	String	N/A
	global_brand_page_name	String	N/A
	global_brand_root_id	String	N/A
	has_added_app	Boolean	N/A
	has_transitioned_to_new_page_experience	Boolean	N/A
	has_whatsapp_business_number	Boolean	N/A
	has_whatsapp_number	Boolean	N/A
	Hometown	String	N/A
	Hours	Struct	N/A
	Impressum	String	N/A
	Influence	String	N/A
	instagram_business_account	String	N/A
	is_always_open	Boolean	N/A
	is_chain	Boolean	N/A
	is_community_page	Boolean	N/A
	is_eligible_for_branded_content	Boolean	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	is_messenger_bot_get_started_enabled	Boolean	N/A
	is_messenger_platform_bot	Boolean	N/A
	is_owned	Boolean	N/A
	is_permanently_closed	Boolean	N/A
	is_published	Boolean	N/A
	Name	String	N/A
	Tasks	List	N/A
	is_unclaimed	Boolean	N/A
	is_webhooks_subscribed	Boolean	N/A
	leadgen_tos_acceptance_time	DateTime	N/A
	leadgen_tos_accepted	Boolean	N/A
	leadgen_tos_accepting_user	String	N/A
	leadgen_tos_accepting_user	Struct	N/A
	Link	Link	N/A
	Location	Struct	N/A
	Members	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	merchant_review_status	String	N/A
	messenger_ads_default_icebreakers	List	N/A
	messenger_ads_default_page_welcome_message	Struct	N/A
	messenger_ads_default_quick_replies	List	N/A
	messenger_ads_quick_replies_type	String	N/A
	Mission	String	N/A
	Mpg	String	N/A
	name_with_location_descriptor	String	N/A
	Network	String	N/A
	new_like_count	Integer	N/A
	offer_eligible	Boolean	N/A
	overall_star_rating	Float	N/A
	page_token	String	N/A
	parent_page	String	N/A
	Parking	String	N/A
	payment_options	Struct	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	personal_info	String	N/A
	personal_interests	String	N/A
	pharma_safety_info	String	N/A
	Phone	String	N/A
	pickup_options	List	N/A
	place_type	String	N/A
	plot_outline	String	N/A
	press_contact	String	N/A
	price_range	String	N/A
	privacy_info_url	String	N/A
	produced_by	String	N/A
	Products	String	N/A
	promotion_eligible	Boolean	N/A
	promotion_ineligible_reason	String	N/A
	public_transit	String	N/A
	rating_count	Integer	N/A
	record_label	String	N/A
	release_date	String	N/A
	restaurant_services	Struct	N/A
	restaurant_specialties	Struct	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	Schedule	String	N/A
	screenplay_by	String	N/A
	Season	String	N/A
	single_line_address	String	N/A
	Starring	String	N/A
	start_info	Struct	N/A
	store_code	String	N/A
	store_location_descriptor	String	N/A
	store_number	Integer	N/A
	Studio	String	N/A
	supports_donate_button_in_live_video	Boolean	N/A
	talking_about_count	Integer	N/A
	temporary_status	String	N/A
	unread_message_count	Integer	N/A
	unread_notif_count	Integer	N/A
	unseen_message_count	Integer	N/A
	Username	String	N/A
	verification_status	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	voip_info	Struct	N/A
	Website	String	N/A
	were_here_count	Integer	N/A
	whatsapp_number	String	N/A
written_by	String	N/A	
Feeds	Id	String	N/A
	Actions	List	N/A
	admin_creator	Object	N/A
	Application	Object	N/A
	Attachments	Objects	N/A
	backdated_time	DateTime	N/A
	call_to_action	Object	N/A
	can_reply_privately	Boolean	N/A
	child_attachments	List	N/A
	Coordinates	Struct	N/A
	created_time	DateTime	N/A
	Event	Struct	N/A
	expanded_height	Integer	N/A
	expanded_width	Integer	N/A
	feed_targeting	Object	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	From	Object	N/A
	full_picture	String	N/A
	Height	Integer	N/A
	Icon	String	N/A
	instagram_eligibility	String	N/A
	is_eligible_for_promotion	Boolean	N/A
	is_expired	Boolean	N/A
	is_hidden	Boolean	N/A
	is_inline_created	Boolean	N/A
	is_instagram_eligible	Boolean	N/A
	is_popular	Boolean	N/A
	is_published	Boolean	N/A
	is_spherical	Boolean	N/A
	Message	String	N/A
	message_tags	List	N/A
	multi_share_end_card	Boolean	N/A
	multi_share_optimized	Boolean	N/A
	parent_id	String	N/A
	permalink_url	String	N/A
	Place	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	Privacy	Object	N/A
	promotable_id	String	N/A
	promotion_status	String	N/A
	Properties	List	N/A
	scheduled_publish_time	Float	N/A
	Shares	Object	N/A
	status_type	String	N/A
	Story	String	N/A
	story_tags	List	N/A
	Subscribed	Boolean	N/A
	Target	Struct	N/A
	Targeting	Object	N/A
	To	Object	N/A
	timeline_visibility	String	N/A
	updated_time	DateTime	N/A
	Via	Struct	N/A
	video_buying_eligibility	List	N/A
	Width	Integer	N/A
	Since	DateTime	EQUAL_TO

분할 쿼리

필터 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  "2024-09-30T01:01:01.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예시:

```
facebookPageInsights_read = glueContext.create_dynamic_frame.from_options(
     connection_type="facebookpageinsights",
     connection_options={
         "connectionName": "connectionName",
         "ENTITY_NAME": "entityName",
         "API_VERSION": "v21",
         "PARTITION_FIELD": "created_Time"
         "LOWER_BOUND": "2024-10-27T07:00:00+0000"
         "UPPER_BOUND": "2024-10-27T07:00:00+0000"
         "NUM_PARTITIONS": "10"
     }
```

Facebook Page Insights 연결 옵션

다음은 Facebook Page Insights의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Facebook Page Insights에서의 객체 이름입니다.

- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Facebook Page Insights Rest API 버전입니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - (필수) 읽기에 사용됩니다. 유효한 Facebook Page Insights 인스턴스 URL입니다.

Facebook Page Insights 커넥터에 대한 제한 사항 및 참고 사항

다음은 Facebook Ads 커넥터의 제한 사항 또는 참고 사항입니다.

- 대부분의 지표는 24시간마다 한 번씩 업데이트됩니다.
- 최근 2년간의 인사이트 데이터만 사용할 수 있습니다.
- since 및 until 파라미터를 사용할 때는 한 번에 90일의 인사이트만 볼 수 있습니다.

Freshdesk에 연결

Freshdesk는 기능이 다양하고 사용하기 쉬운 클라우드 기반 고객 지원 소프트웨어입니다. 라이브 채팅, 이메일, 전화 및 소셜 미디어를 포함한 여러 지원 채널을 사용하면 고객이 선호하는 커뮤니케이션 방법을 통해 고객을 지원할 수 있습니다. Freshdesk 사용자인 경우 Freshdesk 계정에 AWS Glue를 연결할 수 있습니다. Freshdesk를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Freshdesk에서 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송합니다.

주제

- AWS Glue의 Freshdesk 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Freshdesk 구성

- Freshdesk 연결 구성
- Freshdesk 엔터티에서 읽기
- Freshdesk 연결 옵션
- Freshdesk 커넥터의 제한 사항 및 참고 사항

AWS Glue의 Freshdesk 지원

AWS Glue에서는 다음과 같이 Freshdesk를 지원합니다.

소스로 지원되나요?

예 - 동기화 및 비동기. AWS Glue ETL 작업을 사용하여 Freshdesk에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Freshdesk API 버전

다음 Freshdesk API 버전이 지원됩니다.

- v2

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Freshdesk 구성

AWS Glue를 사용하여 Freshdesk에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Freshdesk 계정이 있어야 합니다. Free, Growth, Pro 또는 Enterprise 에디션 중에서 선택할 수 있습니다.
- Freshdesk 사용자의 API 키가 있어야 합니다.

Freshdesk 연결 구성

Freshdesk는 사용자 지정 인증을 지원합니다.

사용자 지정 인증에 필요한 API 키 생성에 대한 퍼블릭 Freshdesk 설명서는 Freshdesk 인증을 참조하세요.

다음은 Freshdesk 연결을 구성하는 단계입니다.

- AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.

  - 고객 관리형의 연결된 앱의 경우 보안 암호는 키 역할을 하는 apiKey와 함께 연결된 앱 API를 포함해야 합니다. AWS Glue에서 연결당 보안 암호를 생성해야 합니다.
- AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
  - 데이터 소스를 선택할 때 Freshdesk를 선택합니다.
  - 연결하려는 Freshdesk 인스턴스의 INSTANCE_URL을 제공합니다.
  - 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

  - 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 secretName을 선택합니다.
  - 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
- AWS Glue 작업 권한과 연결된 IAM 역할에 secretName을 읽을 수 있는 권한을 부여합니다.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 connectionName을 제공합니다.

Freshdesk 엔터티에서 읽기

사전 조건

읽으려는 Stripe 객체입니다. 객체 이름이 필요합니다.

동기화 소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Agents	예	예	아니요	예	예
Business Hours	아니요	예	아니요	예	예
Company	예	예	아니요	예	예
Contacts	예	예	아니요	예	예
Conversations	아니요	예	아니요	예	아니요
Email Configs	아니요	예	아니요	예	아니요
Email Inboxes	예	예	예	예	아니요
Forum Categories	아니요	예	아니요	예	아니요
Forums	아니요	예	아니요	예	아니요
Groups	아니요	예	아니요	예	아니요
Products	아니요	예	아니요	예	아니요
Roles	아니요	예	아니요	예	아니요
Satisfaction Ratings	예	예	아니요	예	아니요
Skills	아니요	예	아니요	예	아니요
Solutions	예	예	아니요	예	아니요
Surveys	아니요	예	아니요	예	아니요
Tickets	예	예	예	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Time Entries	예	예	아니요	예	아니요
주제	아니요	예	아니요	예	아니요
Topic Comments	아니요	예	아니요	예	아니요

비동기 소스에 대해 지원되는 엔터티:

개체	API 버전	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Companies	v2	아니요	아니요	아니요	아니요	아니요
Contacts	v2	아니요	아니요	아니요	아니요	아니요

예시:

```
freshdesk_read = glueContext.create_dynamic_frame.from_options(
    connection_type="freshdesk",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v2"
    }
```

Freshdesk 엔터티 및 필드 세부 정보

개체	필드
Agents	https://developers.freshdesk.com/api/#list_all_agents
Business-hours	https://developers.freshdesk.com/api/#list_all_business_hours

개체	필드
설명	https://developers.freshdesk.com/api/#comment_attributess
Company	https://developers.freshdesk.com/api/#companies
Contacts	https://developers.freshdesk.com/api/#list_all_contacts
Conversations	https://developers.freshdesk.com/api/#list_all_ticket_notes
Email-configs	https://developers.freshdesk.com/api/#list_all_email_configs
Email-inboxes	https://developers.freshdesk.com/api/#list_all_email_mailboxes
Forum-categories	https://developers.freshdesk.com/api/#category_attributes
Forums	https://developers.freshdesk.com/api/#forum_attributes
Groups	https://developers.freshdesk.com/api/#list_all_groups
Products	https://developers.freshdesk.com/api/#list_all_products
Roles	https://developers.freshdesk.com/api/#list_all_roles
Satisfaction-rating	https://developers.freshdesk.com/api/#view_all_satisfaction_ratingss
Skills	https://developers.freshdesk.com/api/#list_all_skills

개체	필드
Solutions	https://developers.freshdesk.com/api/#solution_content
Surveys	https://developers.freshdesk.com/api/#list_all_survey
Tickets	https://developers.freshdesk.com/api/#list_all_tickets
Time-entries	https://developers.freshdesk.com/api/#list_all_time_entries
주제	https://developers.freshdesk.com/api/#topic_attributes

분할 쿼리

필터 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  "2024-09-30T01:01:01.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예시:

```
freshDesk_read = glueContext.create_dynamic_frame.from_options(
     connection_type="freshdesk",
     connection_options={
         "connectionName": "connectionName",
         "ENTITY_NAME": "entityName",
         "API_VERSION": "v2",
         "PARTITION_FIELD": "Created_Time"
         "LOWER_BOUND": " 2024-10-27T23:16:08Z“
         "UPPER_BOUND": " 2024-10-27T23:16:08Z"
         "NUM_PARTITIONS": "10"
     }
```

Freshdesk 연결 옵션

다음은 Freshdesk의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Freshdesk에서의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Freshdesk Rest API 버전입니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - (필수) 읽기에 사용됩니다. 유효한 Freshdesk 인스턴스 URL입니다.
- TRANSFER_MODE(문자열) - SYNC 또는 ASYNC 등의 처리 유형이 기본적으로 SYNC로 설정되어 있는지 여부를 나타내는 데 사용됩니다. (선택 사항)

Freshdesk 커넥터의 제한 사항 및 참고 사항

다음은 Freshdesk 커넥터의 제한 사항입니다.

- 필터링이 있는 Company, Contacts 및 Tickets 엔터티에는 페이지 매김 제한이 있습니다. 페이지당 30개의 레코드만 반환하며 페이지 값은 최대 10개(최대 300개의 레코드 가져오기)로 설정할 수 있습니다.
- Tickets 엔터티는 30일보다 오래된 레코드를 가져오지 않습니다.
- Company, Contacts 및 Tickets엔터티는 필터링에서 '날짜' 데이터 유형을 지원합니다. 이 세 엔터티에 대해 '일일' 이후 트리거 빈도를 선택해야 합니다. '분' 또는 '시간당'을 선택하면 데이터가 중복될 수 있습니다. 또한 필터링을 위해 이러한 필드를 선택하는 동안 선택한 타임스탬프의 날짜 부분만 고려하므로 날짜 값만 선택해야 합니다.
- 분당 API 직접 호출 수는 해당 계획을 기반으로 합니다. 이 제한은 호출에 사용된 에이전트 수 또는 IP 주소와 같은 요인에 관계없이 계정 전체에 적용됩니다. 모든 평가판 사용자의 경우 기본 API 제한은 분당 50회 호출입니다. 자세한 내용은 Freshdesk를 참조하세요.
- 엔터티의 경우 한 번에 하나의 내보내기/비동기 작업만 처리됩니다. 새 작업은 기존 작업이 성공적으로 완료되거나 실패한 후에만 처리됩니다. 자세한 내용은 Freshdesk를 참조하세요.
- 다음 필드는 동기화 API 직접 호출에 지원되지만 비동기 API 요청 본문에서 전달이 지원되지 않거나 허용되지 않습니다.
  - id
  - created_at
  - updated_at
  - updated_since
  - 활성화
  - company_id
  - other_companies
  - avatar
  - view_all_tickets
  - deleted
  - other_emails
  - state
  - 태그
  - tags

Freshsales에 연결

Freshsales는 영업 담당자가 영업에서 추측을 배제할 수 있도록 도와주는 직관적인 CRM입니다. 전화와 이메일, 작업, 약속, 메모가 기본으로 제공되므로 영업 담당자는 잠재 고객에 대한 후속 조치를 위해 여러 탭을 전환할 필요가 없습니다. 파이프라인 보기를 통해 거래를 더 잘 관리하고 더 많은 거래를 성사시킬 수 있습니다. Freshsales 사용자인 경우 Freshsales 계정에 AWS Glue를 연결할 수 있습니다. Freshsales를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Freshsales에서 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송합니다.

주제

- AWS Glue의 Freshsales 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Freshsales 구성
- Freshsales 연결 구성
- Freshsales 엔터티에서 읽기
- Freshsales 연결 옵션
- Freshsales 제한 사항

AWS Glue의 Freshsales 지원

AWS Glue에서는 다음과 같이 Freshsales를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Freshsales에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Freshsales API 버전

다음 Freshsales API 버전이 지원됩니다.

- v1.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Freshsales 구성

AWS Glue를 사용하여 Freshsales에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Freshsales 계정이 있습니다.
- 사용자 API 키가 있습니다.

이러한 요구 사항을 충족하면 Freshsales 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Freshsales에서 다른 작업을 수행하지 않아도 됩니다.

Freshsales 연결 구성

Freshsales는 사용자 지정 인증을 지원합니다.

사용자 지정 인증에 필요한 API 키 생성에 대한 퍼블릭 Freshsales 설명서는 Authentication을 참조하세요.

Freshsales 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 *apiSecretKey*과 함께 연결된 앱 API를 포함해야 합니다. 또한 보안 암호에는 *apiKey*을 키로 사용하고 *token*을 값으로 사용하는 다른 키-값 쌍이 포함되어야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 데이터 소스를 선택할 때 Freshsales를 선택합니다.
   b. 연결하려는 Freshsales 계정의 `INSTANCE_URL`을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Freshsales 엔터티에서 읽기

사전 조건

읽으려는 Freshsales 객체입니다. 객체 이름이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Accounts	예	예	예	예	예
Contacts	예	예	예	예	예

예시:

```
freshSales_read = glueContext.create_dynamic_frame.from_options(
     connection_type="freshsales",
     connection_options={
         "connectionName": "connectionName",
         "ENTITY_NAME": "entityName",
         "API_VERSION": "v1.0"
     }
```

Freshsales 엔터티 및 필드 세부 정보:

Freshsales에서는 지원되는 엔터티에 대해 메타데이터를 동적으로 가져오도록 엔드포인트를 제공합니다. 따라서 운영자 지원은 데이터 유형 수준에서 캡처됩니다.

엔터티	데이터 유형	지원되는 연산자
Freshsale 엔터티(모두)	Integer	!=,=,<,<=,>,>=,BETWEEN
	String	Like, =, !=
	BigInteger	!=,=,<,<=,>,>=,BETWEEN
	Boolean	=
	Double	!=,=,<,<=,>,>=,BETWEEN
	BigDecimal	!=,=,<,<=,>,>=,BETWEEN
	Date	!=,=,<,<=,>,>=,BETWEEN
	DateTime	!=,=,<,<=,>,>=,BETWEEN
	Struct	N/A
	List	N/A

분할 쿼리

필터 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2024-09-30T01:01:01.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).

- NUM_PARTITIONS: 파티션 수.

예시:

```
freshSales_read = glueContext.create_dynamic_frame.from_options(
     connection_type="freshsales",
     connection_options={
         "connectionName": "connectionName",
         "ENTITY_NAME": "entityName",
         "API_VERSION": "v1",
         "PARTITION_FIELD": "Created_Time"
         "LOWER_BOUND": " 2024-10-15T21:16:25Z"
         "UPPER_BOUND": " 2024-10-20T21:25:50Z"
         "NUM_PARTITIONS": "10"
     }
```

Freshsales 연결 옵션

다음은 Freshsales의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Freshsales에서의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Freshsales Rest API 버전입니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - 읽기에 사용됩니다. 유효한 Freshsales 인스턴스 URL입니다.

Freshsales 제한 사항

다음은 Freshsales의 제한 사항 또는 참고 사항입니다.

- Freshsales에서 API 속도 제한은 계정당 시간당 1,000개의 API 요청으로 설정되어 있습니다(오류 참조). 단, 이 제한은 엔터프라이즈 구독 요금제로 확장할 수 있습니다(플랜 비교 참조).

Google Ads에 연결

Google Ads API는 대규모 또는 복잡한 Google Ads 계정 및 캠페인을 관리하는 데 사용되는 Google Ads에 대한 프로그래밍 방식 인터페이스입니다. Google Ads 사용자인 경우 Google Ads 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, Google Ads를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Google Ads 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- Google Ads의 AWS Glue 지원
- 연결 생성 및 사용 API 작업이 포함된 정책
- Google Ads 구성
- Google Ads 연결 구성
- Google Ads 엔터티에서 읽기
- Google Ads 연결 옵션
- Google Ads 계정 생성
- 제한 사항

Google Ads의 AWS Glue 지원

AWS Glue는 다음과 같이 Google Ads를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Google Ads에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Google Ads API 버전

v18

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

다음 관리형 IAM 정책을 사용하여 액세스를 허용할 수도 있습니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 관리 콘솔을 사용할 때 AWS Glue 리소스에 AWS 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

Google Ads 구성

AWS Glue 를 사용하여 Google Ads에서 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호가 있는 Google Ads 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Google Ads 계정 생성을 참조하세요.
- Google Ads 계정에 API 액세스할 수 있습니다. Google Ads는 추가 비용 없이 사용할 API 수 있습니다.
- Google Ads 계정을 사용하면 연결된 앱을 설치할 수 있습니다. 이 기능에 대한 액세스 권한이 없는 경우 Google Ads 관리자에게 문의하세요.

이러한 요구 사항을 충족하면 Google Ads 계정에 AWS Glue 연결할 준비가 된 것입니다.

Google Ads 연결 구성

Google Ads는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Google Ads로 리디렉션합니다. 사용자가 로그인하고 Google Ads 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Google Ads에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Google Ads로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

자세한 내용은 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성의 퍼블릭 Google Ads 설명서를 참조하세요.

Google Ads 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. AuthorizationCode 권한 부여 유형의 경우:
      - 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.

   a. 연결 유형을 선택할 때 Google Ads를 선택합니다.

   b. 연결하려는 Facebook Ads의 `developer token`을 제공합니다.

   c. 관리자로 로그인하려는 경우 Google Ads의 `MANAGER ID`를 제공합니다.

   d. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   e. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

   f. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Google Ads 엔터티에서 읽기

사전 조건

- 읽으려는 Google Ads 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
광고 그룹 광고	예	예	예	아니요	예
광고 그룹	예	예	예	아니요	예
캠페인 예산	예	예	예	예	예
계정 예산	예	아니요	예	예	No
캠페인	예	예	예	예	예
계정	예	아니요	예	아니요	No

예

```
googleAds_read = glueContext.create_dynamic_frame.from_options(
    connection_type="googleads",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "campaign-3467***",
        "API_VERSION": "v16"
    }
```

Google Ads 엔터티 및 필드 세부 정보

개체	필드	데이터 형식	지원되는 연산자
계정	resourceName	String	!=, =
계정	callReportingEnabled	불	!=, =
계정	callConversionReportingEnabled	불	!=, =
계정	callConversionAction	String	!=, =

개체	필드	데이터 형식	지원되는 연산자
계정	conversionTrackingId	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	crossAccountConversionTrackingId	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	payPerConversionEligibilityFailureReasons	나열	
계정	id	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	currencyCode	String	!=, =, LIKE
계정	timeZone	String	!=, =, LIKE
계정	autoTaggingEnabled	불	!=, =
계정	hasPartnersBadge	불	!=, =
계정	manager	불	!=, =
계정	testAccount	불	!=, =
계정	date	날짜	BETWEEN, =, <, >, <=, >=
계정	costMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	acceptedCustomerDataTerms	불	
계정	conversionTrackingStatus	String	!=, =, LIKE

개체	필드	데이터 형식	지원되는 연산자
계정	enhancedConversionsForLeadsEnabled	불	
계정	googleAdsConversionCustomer	String	
계정	status	String	!=, =
계정	allConversionsByConversionDate	배정밀도 실수	!=, =, <, >
계정	allConversionsValueByConversionDate	배정밀도 실수	!=, =, <, >
계정	conversionsByConversionDate	배정밀도 실수	!=, =, <, >
계정	conversionsValueByConversionDate	배정밀도 실수	!=, =, <, >
계정	valuePerAllConversionsByConversionDate	배정밀도 실수	!=, =, <, >
계정	videoViews	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	clicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	invalidClicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	costPerAllConversions	배정밀도 실수	!=, =, <, >
계정	costPerConversion	배정밀도 실수	!=, =, <, >

개체	필드	데이터 형식	지원되는 연산자
계정	conversions	배정밀도 실수	!=, =, <, >
계정	absoluteTopImpressionPercentage	배정밀도 실수	!=, =, <, >
계정	노출	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	topImpressionPercentage	배정밀도 실수	!=, =, <, >
계정	averageCpc	배정밀도 실수	!=, =, <, >
계정	activeViewMeasurableCostMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정	averageCost	배정밀도 실수	!=, =, <, >
계정	ctr	배정밀도 실수	!=, =, <, >
계정	activeViewCtr	배정밀도 실수	!=, =, <, >
계정	searchImpressionShare	배정밀도 실수	!=, =, <, >
계정	conversionAction	String	!=, =
계정	conversionActionCategory	String	!=, =
계정	conversionActionName	String	!=, =, LIKE
계정 예산	resourceName	String	!=, =
계정 예산	status	String	!=, =
계정 예산	proposedEndTimeType	String	!=, =

개체	필드	데이터 형식	지원되는 연산자
계정 예산	approvedEndTimeType	String	!=, =
계정 예산	id	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정 예산	billingSetup	String	!=, =
계정 예산	name	String	!=, =, LIKE
계정 예산	approvedStartDateTime	DateTime	BETWEEN, =, <, >, <=, >=
계정 예산	proposedSpendingLimitMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정 예산	approvedSpendingLimitMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정 예산	adjustedSpendingLimitMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
계정 예산	amountServedMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	resourceName	String	!=, =, LIKE
광고 그룹	status	String	!=, =, LIKE
광고 그룹	type	String	!=, =, LIKE
광고 그룹	id	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	name	String	!=, =, LIKE
광고 그룹	campaign	String	!=, =

개체	필드	데이터 형식	지원되는 연산자
광고 그룹	cpcBidMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	targetCpaMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	cpmBidMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	cpvBidMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	targetCpmMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	effectiveTargetCpaMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	date	날짜	BETWEEN, =, <, >, <=, >=
광고 그룹	costMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	useAudienceGrouped	불	!=, =
광고 그룹	effectiveCpcBidMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	allConversionsByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹	allConversionsValueByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹	conversionsByConversionDate	배정밀도 실수	!=, =, <, >

개체	필드	데이터 형식	지원되는 연산자
광고 그룹	conversionsValueByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹	valuePerAllConversionsByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹	valuePerConversionsByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹	averageCost	배정밀도 실수	!=, =, <, >
광고 그룹	costPerAllConversions	배정밀도 실수	!=, =, <, >
광고 그룹	costPerConversion	배정밀도 실수	!=, =, <, >
광고 그룹	averagePageViews	배정밀도 실수	!=, =, <, >
광고 그룹	videoViews	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	clicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	allConversions	배정밀도 실수	!=, =, <, >
광고 그룹	averageCpc	배정밀도 실수	!=, =, <, >
광고 그룹	absoluteTopImpressionPercentage	배정밀도 실수	!=, =, <, >
광고 그룹	노출	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹	topImpressionPercentage	배정밀도 실수	!=, =, <, >

개체	필드	데이터 형식	지원되는 연산자
광고 그룹	activeViewCtr	배정밀도 실수	!=, =, <, >
광고 그룹	ctr	배정밀도 실수	!=, =, <, >
광고 그룹	searchTopImpressionShare	배정밀도 실수	!=, =, <, >
광고 그룹	searchImpressionShare	배정밀도 실수	!=, =, <, >
광고 그룹	searchAbsoluteTopImpressionShare	배정밀도 실수	!=, =, <, >
광고 그룹	relativeCtr	배정밀도 실수	!=, =, <, >
광고 그룹	conversionAction	String	!=, =
광고 그룹	conversionActionCategory	String	!=, =
광고 그룹	conversionActionName	String	!=, =, LIKE
광고 그룹	updateMask	String	
광고 그룹	생성	구조체	
광고 그룹	업데이트	구조체	
광고 그룹	primaryStatus	String	!=, =
광고 그룹	primaryStatusReasons	나열	
광고 그룹 광고	resourceName	String	!=, =
광고 그룹 광고	id	BigInteger	BETWEEN, =, !=, <, >, <=, >=

개체	필드	데이터 형식	지원되는 연산자
광고 그룹 광고	status	String	!=, =
광고 그룹 광고	labels	나열	
광고 그룹 광고	adGroup	String	!=, =
광고 그룹 광고	costMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹 광고	approvalStatus	String	!=, =
광고 그룹 광고	reviewStatus	String	!=, =
광고 그룹 광고	adStrength	String	!=, =
광고 그룹 광고	type	String	!=, =
광고 그룹 광고	businessName	String	!=, =, LIKE
광고 그룹 광고	date	날짜	BETWEEN, =, <, >, <=, >=
광고 그룹 광고	allConversionsByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹 광고	allConversionsValueByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹 광고	conversionsByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹 광고	conversionsValueByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹 광고	valuePerAllConversionsByConversionDate	배정밀도 실수	!=, =, <, >

개체	필드	데이터 형식	지원되는 연산자
광고 그룹 광고	valuePerConversionsByConversionDate	배정밀도 실수	!=, =, <, >
광고 그룹 광고	activeViewMeasurableCostMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹 광고	averageCost	배정밀도 실수	!=, =, <, >
광고 그룹 광고	costPerAllConversions	배정밀도 실수	!=, =, <, >
광고 그룹 광고	costPerConversion	배정밀도 실수	!=, =, <, >
광고 그룹 광고	clicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹 광고	averagePageViews	배정밀도 실수	!=, =, <, >
광고 그룹 광고	videoViews	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹 광고	allConversions	배정밀도 실수	!=, =, <, >
광고 그룹 광고	averageCpc	배정밀도 실수	!=, =, <, >
광고 그룹 광고	topImpressionPercentage	배정밀도 실수	!=, =, <, >
광고 그룹 광고	노출	BigInteger	BETWEEN, =, !=, <, >, <=, >=
광고 그룹 광고	absoluteTopImpressionPercentage	배정밀도 실수	!=, =, <, >
광고 그룹 광고	activeViewCtr	배정밀도 실수	!=, =, <, >
광고 그룹 광고	ctr	배정밀도 실수	!=, =, <, >
광고 그룹 광고	conversionAction	String	!=, =

개체	필드	데이터 형식	지원되는 연산자
광고 그룹 광고	conversionActionCategory	String	!=, =
광고 그룹 광고	conversionActionName	String	!=, =, LIKE
광고 그룹 광고	updateMask	String	
광고 그룹 광고	생성	구조체	
광고 그룹 광고	업데이트	구조체	
광고 그룹 광고	policyValidationParameter	구조체	
광고 그룹 광고	primaryStatus	String	!=, =
광고 그룹 광고	primaryStatusReasons	나열	
캠페인	resourceName	String	!=, =
캠페인	status	String	!=, =
캠페인	baseCampaign	String	!=, =
캠페인	name	String	!=, =, LIKE
캠페인	id	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인	campaignBudget	String	!=, =, LIKE
캠페인	startDate	날짜	BETWEEN, =, <, >, <=, >=
캠페인	endDate	날짜	BETWEEN, =, <, >, <=, >=

개체	필드	데이터 형식	지원되는 연산자
캠페인	adServingOptimizationStatus	String	!=, =
캠페인	advertisingChannelType	String	!=, =
캠페인	advertisingChannelSubType	String	!=, =
캠페인	experimentType	String	!=, =
캠페인	servingStatus	String	!=, =
캠페인	biddingStrategyType	String	!=, =
캠페인	domainName	String	!=, =, LIKE
캠페인	languageCode	String	!=, =, LIKE
캠페인	useSuppliedUrlsOnly	불	!=, =
캠페인	positiveGeoTargetType	String	!=, =
캠페인	negativeGeoTargetType	String	!=, =
캠페인	paymentMode	String	!=, =
캠페인	optimizationGoalTypes	나열	
캠페인	date	날짜	BETWEEN, =, <, >, <=, >=
캠페인	averageCost	배정밀도 실수	
캠페인	clicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=

개체	필드	데이터 형식	지원되는 연산자
캠페인	costMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인	노출	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인	useAudienceGrouped	불	!=, =
캠페인	activeViewMeasurableCostMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인	costPerAllConversions	배정밀도 실수	!=, =, <, >
캠페인	costPerConversion	배정밀도 실수	!=, =, <, >
캠페인	invalidClicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인	publisherPurchasedClicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인	averagePageViews	배정밀도 실수	!=, =, <, >
캠페인	videoViews	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인	allConversionsByConversionDate	배정밀도 실수	!=, =, <, >
캠페인	allConversionsValueByConversionDate	배정밀도 실수	!=, =, <, >
캠페인	conversionsByConversionDate	배정밀도 실수	!=, =, <, >
캠페인	conversionsValueByConversionDate	배정밀도 실수	!=, =, <, >

개체	필드	데이터 형식	지원되는 연산자
캠페인	valuePerAllConversionsByConversionDate	배정밀도 실수	!=, =, <, >
캠페인	valuePerConversionsByConversionDate	배정밀도 실수	!=, =, <, >
캠페인	allConversions	배정밀도 실수	!=, =, <, >
캠페인	absoluteTopImpressionPercentage	배정밀도 실수	!=, =, <, >
캠페인	searchAbsoluteTopImpressionShare	배정밀도 실수	!=, =, <, >
캠페인	averageCpc	배정밀도 실수	!=, =, <, >
캠페인	searchImpressionShare	배정밀도 실수	!=, =, <, >
캠페인	searchTopImpressionShare	배정밀도 실수	!=, =, <, >
캠페인	activeViewCtr	배정밀도 실수	!=, =, <, >
캠페인	ctr	배정밀도 실수	!=, =, <, >
캠페인	relativeCtr	배정밀도 실수	!=, =, <, >
캠페인	updateMask	String	
캠페인	생성	구조체	
캠페인	업데이트	구조체	
캠페인 예산	resourceName	String	!=, =
캠페인 예산	id	BigInteger	BETWEEN, =, !=, <, >, <=, >=

개체	필드	데이터 형식	지원되는 연산자
캠페인 예산	status	String	!=, =
캠페인 예산	deliveryMethod	String	!=, =
캠페인 예산	기간	String	!=, =
캠페인 예산	type	String	!=, =
캠페인 예산	name	String	!=, =, LIKE
캠페인 예산	amountMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인 예산	explicitlyShared	불	!=, =
캠페인 예산	referenceCount	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인 예산	hasRecommendedBudget	불	!=, =
캠페인 예산	date	날짜	BETWEEN, =, <, >, <=, >=
캠페인 예산	costMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인 예산	startDate	날짜	BETWEEN, =, <, >, <=, >=
캠페인 예산	endDate	날짜	BETWEEN, =, <, >, <=, >=
캠페인 예산	maximizeConversionValueTargetRoas	배정밀도 실수	!=, =, <, >
캠페인 예산	maximizeConversionsTargetCpaMicros	BigInteger	BETWEEN, =, !=, <, >, <=, >=

개체	필드	데이터 형식	지원되는 연산자
캠페인 예산	selectiveOptimizationConversionActions	String	
캠페인 예산	averageCost	배정밀도 실수	!=, =, <, >
캠페인 예산	costPerAllConversions	배정밀도 실수	!=, =, <, >
캠페인 예산	costPerConversion	배정밀도 실수	!=, =, <, >
캠페인 예산	videoViews	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인 예산	clicks	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인 예산	allConversions	배정밀도 실수	!=, =, <, >
캠페인 예산	valuePerAllConversions	배정밀도 실수	!=, =, <, >
캠페인 예산	averageCpc	배정밀도 실수	!=, =, <, >
캠페인 예산	노출	BigInteger	BETWEEN, =, !=, <, >, <=, >=
캠페인 예산	ctr	배정밀도 실수	!=, =, <, >
캠페인 예산	updateMask	String	
캠페인 예산	생성	구조체	
캠페인 예산	업데이트	구조체	

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-02-06".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준의 분할 필드 지원 세부 정보는 다음 표에 캡처되어 있습니다.

Entity Name	분할 필드	데이터 형식
광고 그룹 광고	date	날짜
광고 그룹	date	날짜
캠페인	date	날짜
캠페인 예산	date	날짜

예제

```
googleads_read = glueContext.create_dynamic_frame.from_options(
    connection_type="googleads",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "campaign-3467***",
        "API_VERSION": "v16",
        "PARTITION_FIELD": "date"
        "LOWER_BOUND": "2024-01-01"
        "UPPER_BOUND": "2024-06-05"
        "NUM_PARTITIONS": "10"
    }
)
```

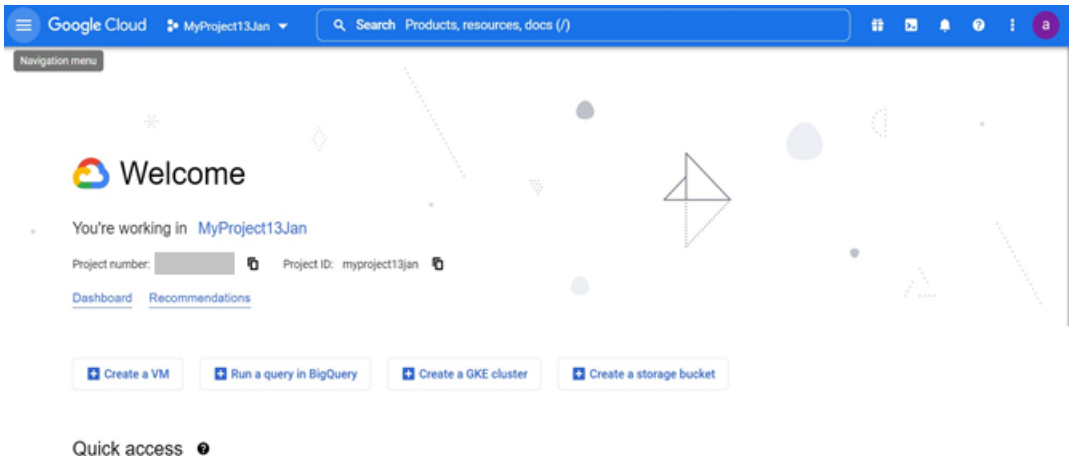
Google Ads 연결 옵션

다음은 Google Ads의 연결 옵션입니다.

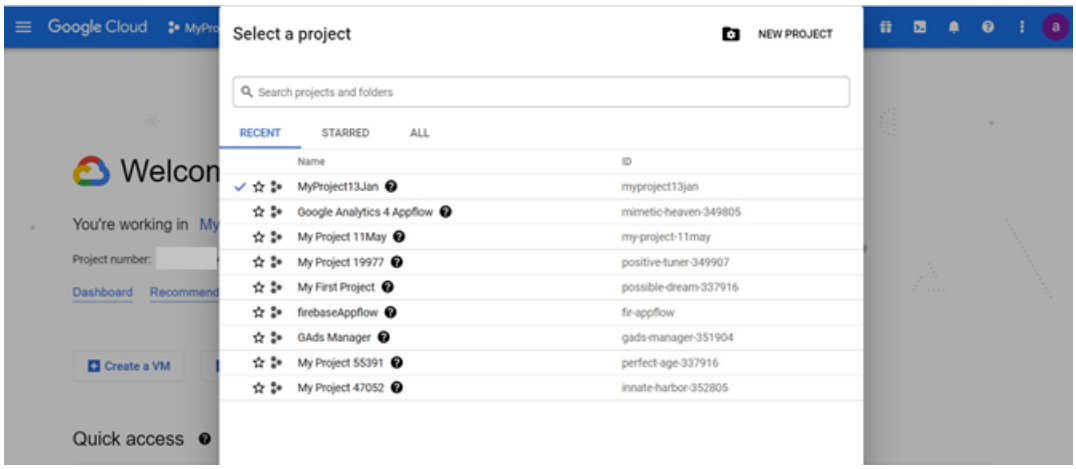
- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Google Ads에서의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용할 Snapchat Ads Rest API 버전입니다. 예: v16.
- DEVELOPER_TOKEN(문자열) - (필수) 읽기/쓰기에 사용됩니다. API에 요청하는 개발자 또는 애플리케이션을 인증하는 데 필요합니다.
- MANAGER_ID(문자열)-읽기/쓰기에 사용됩니다. 여러 Google Ads 계정을 관리할 수 있는 고유 ID입니다. 권한 있는 관리자의 고객 ID입니다. 관리자 계정을 통해 고객 계정에 액세스하는 경우 MANAGER_ID가 필요합니다. 자세한 내용은 login-customer-id를 참조하세요.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

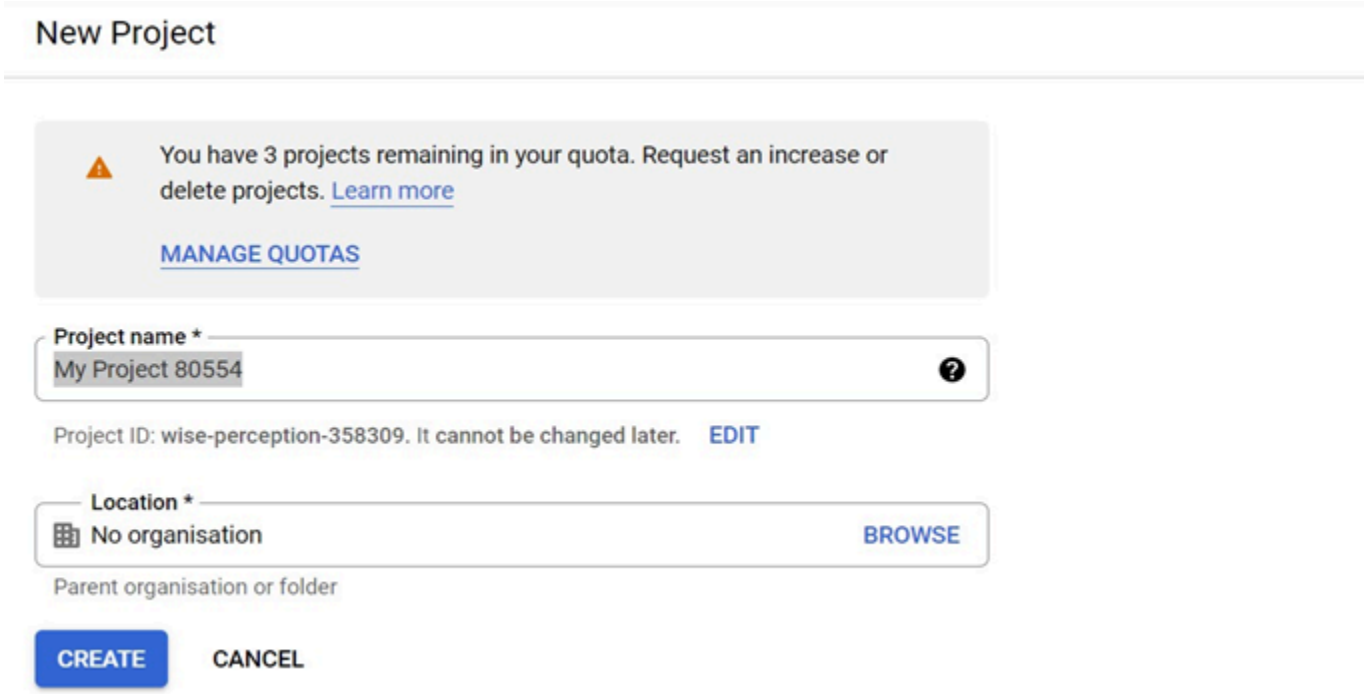
Google Ads 계정 생성

1. 자격 증명으로 Google Ads 개발자 계정에 로그인하고 *로 이동합니다MyProject.

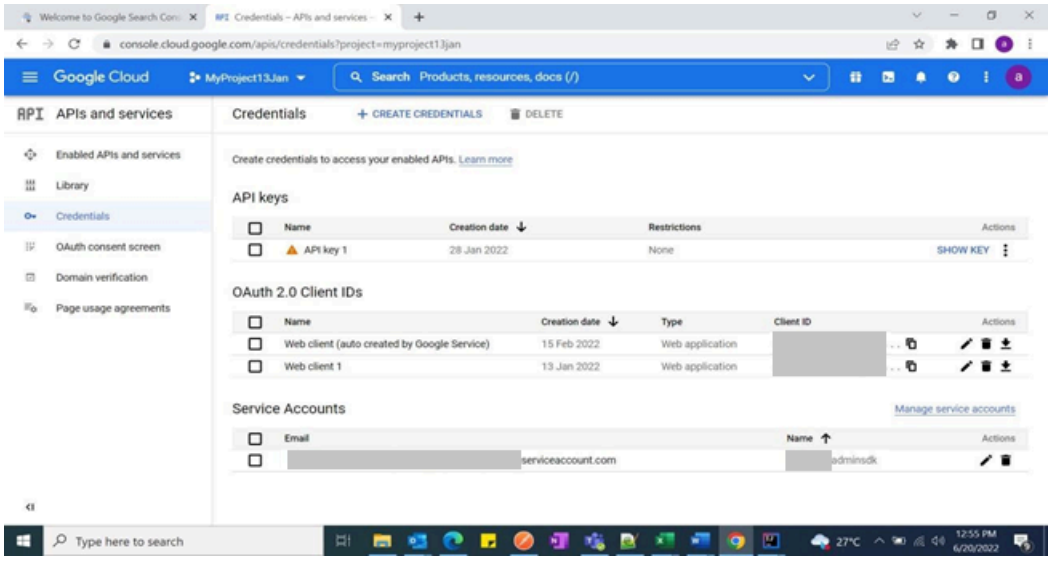


2. 등록된 애플리케이션이 없는 경우 새 프로젝트를 선택하고 Google 프로젝트를 생성하는 데 필요한 정보를 제공합니다.

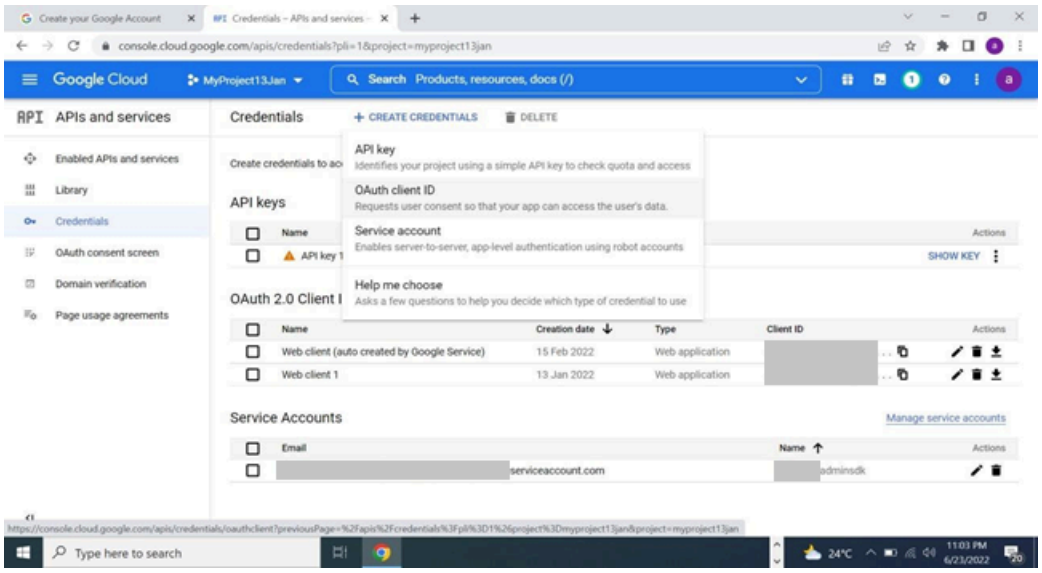




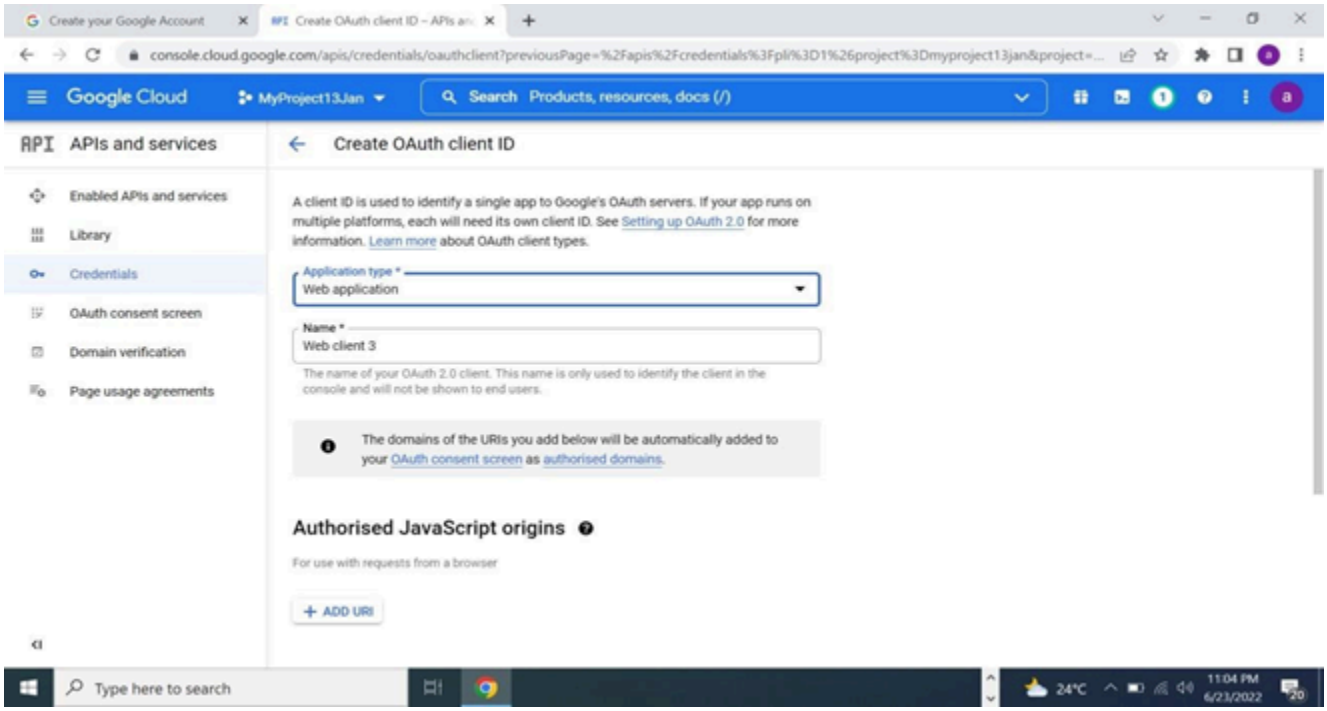
3. 탐색 탭 , API , 설정 , 클라이언트 ID 생성 및 를 선택합니다. ClientSecret 이렇게 하려면 AWS Glue 및 간의 연결을 생성하기 위한 추가 구성이 필요합니다 GoogleAds. 자세한 내용은 API 보안 인증 정보 섹션을 참조하세요.



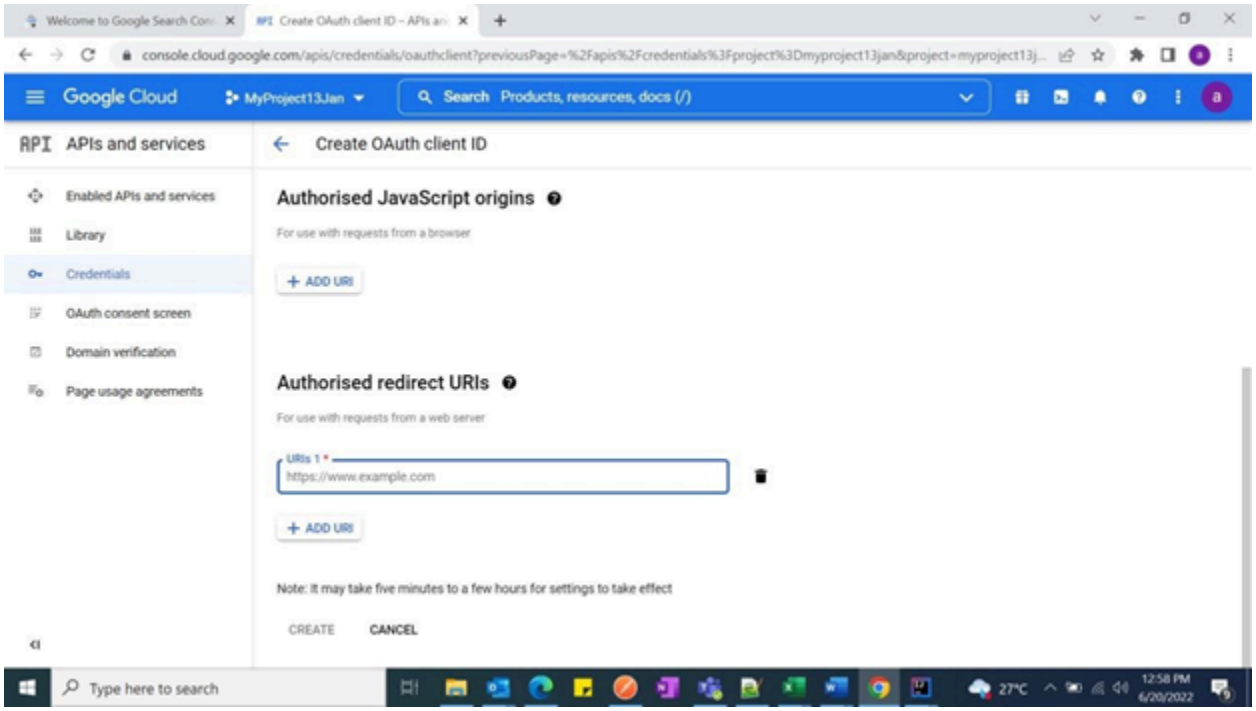
4. CREATE CREDENTIALS 를 선택하고 OAuth 클라이언트 ID 를 선택합니다.



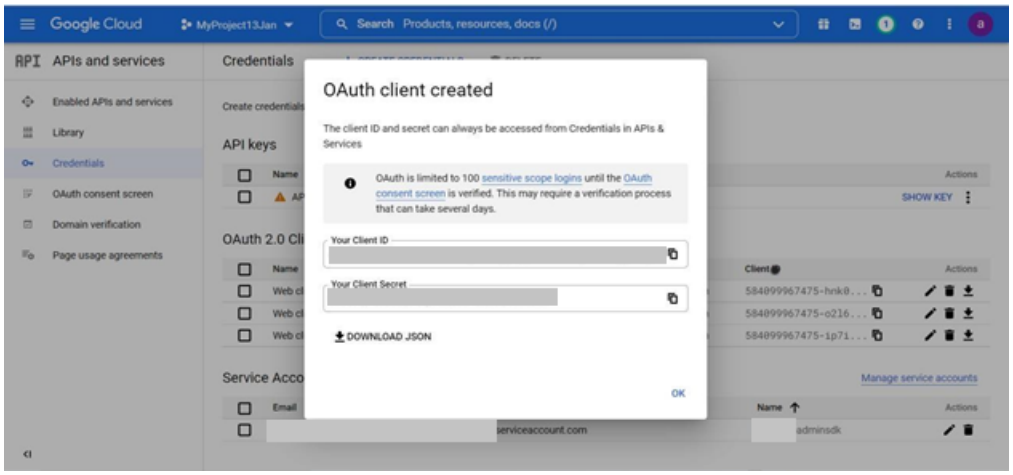
5. 애플리케이션 유형을 웹 애플리케이션 으로 선택합니다.



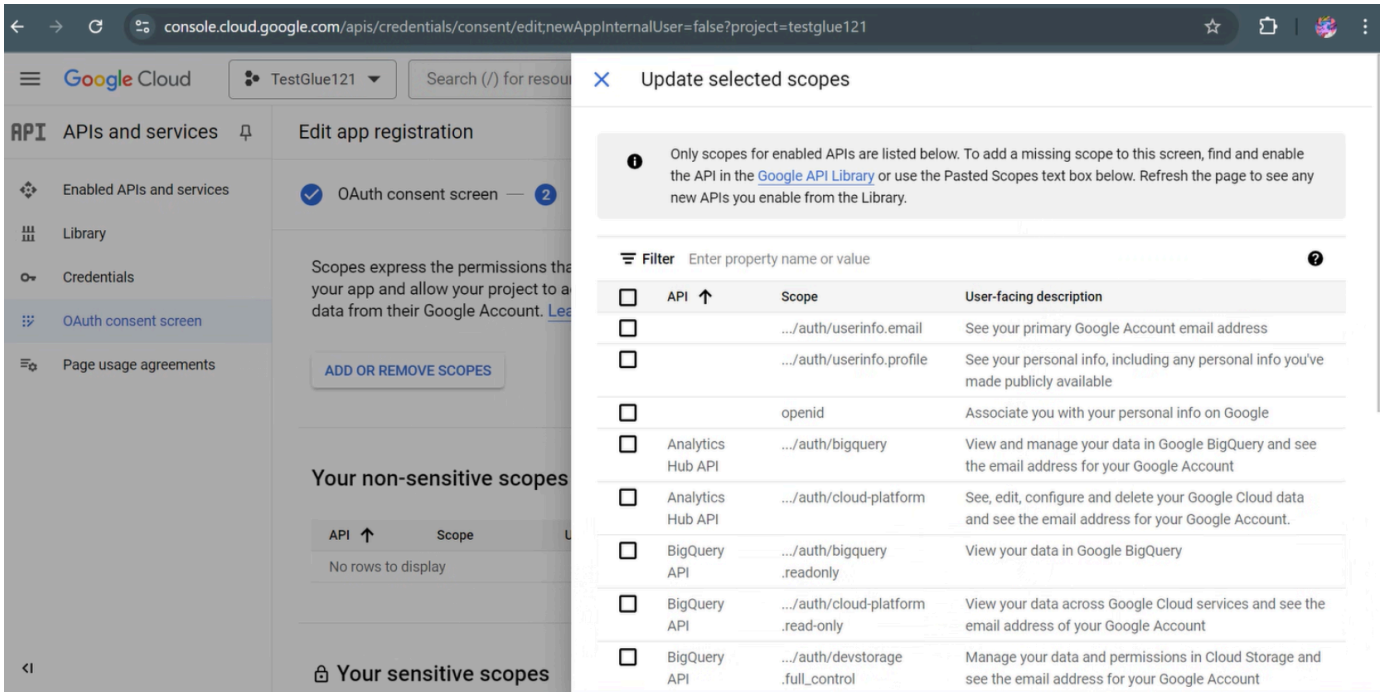
6. 승인된 리디렉션URIs에서 OAuth리디렉션을 추가URIs하고 생성을 선택합니다. 필요한 URIs 경우 여러 리디렉션을 추가할 수 있습니다.



7. 및 Google Ads 간에 연결을 생성할 때 클라이언트 ID AWS Glue 와 클라이언트 보안 암호가 생성됩니다.



8. 애플리케이션 요구 사항에 따라 범위를 추가하고, OAuth 동의 화면을 선택하고, 필요한 정보를 제공하고, 요구 사항에 따라 범위를 추가합니다.



제한 사항

다음은 Google Ads 커넥터의 제한 사항입니다.

- MANAGER_ID는 연결을 생성하는 경우 선택적 입력입니다. 그러나 특정 관리자의 기반이 되는 고객에게 액세스하려는 경우 MANAGER_ID는 필수 입력입니다. 아래 표에서는 연결에서 MANAGER_ID의 포함 여부에 따른 액세스 제한 사항을 설명합니다.

연결을 생성하는 경우 MANAGER_ID 제공 여부?	액세스 가능한 고객
예	제공된 MANAGER_ID 아래 나열된 고객 및 MANAGER_ID.
아니요	모든 고객이 나열되지만 모든 관리자의 기본 고객에는 액세스할 수 없습니다.

- 관리자 계정을 객체로 선택하면 `Account`만 하위 객체로 표시됩니다. Google Ads 커넥터에서 캠페인, 광고 등과 같은 엔터티는 관리자 계정이 아닌 개별 클라이언트 계정을 기반으로 검색됩니다.
- 관리자 계정에 대한 지표는 검색할 수 없습니다. 대신 개별 클라이언트 계정에 대한 지표를 검색할 수 있습니다.
- 각 계정에는 활성 캠페인과 일시 중지된 캠페인을 모두 포함하여 최대 10,000개의 캠페인이 있을 수 있습니다. 자세한 내용은 Campaign per account를 참조하세요.
- 보고서를 생성할 때 표시할 특정 지표를 선택하는 경우 선택한 지표가 모두 0인 행은 반환되지 않습니다. 자세한 내용은 Zero Metrics를 참조하세요.
- 다음 필드의 경우 전체 매핑 흐름은 계정, 광고 그룹 및 광고 그룹 광고 엔터티, 특히 conversionAction, conversionActionCategory, conversionActionName에 대해서는 작동하지 않습니다. 자세한 내용은 Segment and Metrics를 참조하세요.
- `segments.date` 필드를 선택하면 날짜 범위 필터가 필수입니다.

Google Analytics에 연결 4

Google Analytics 4는 앱 및 웹 사이트와의 방문자 상호 작용에 대한 지표를 추적하고 보고하는 분석 서비스입니다. 이러한 지표에는 페이지 뷰, 활성 사용자 및 이벤트가 포함됩니다. Google Analytics 4 사용자인 경우 Google Analytics 4 계정에 AWS Glue 연결할 수 있습니다. Google Analytics 4를 ETL 작업의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Google Analytics 4에서 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송합니다.

주제

- AWS Glue Google Analytics 4 지원
- 연결 생성 및 사용 API 작업이 포함된 정책
- Google Analytics 구성 4

- Google Analytics 4 연결 구성
- Google Analytics 4 엔터티에서 읽기
- Google Analytics 4 연결 옵션
- Google Analytics 4 계정 생성
- 클라이언트 앱 및 OAuth 2.0 보안 인증 생성 단계
- 제한 사항 및 고려 사항

AWS Glue Google Analytics 4 지원

AWS Glue 는 다음과 같이 Google Analytics 4를 지원합니다.

소스로 지원되나요?

예. 작업을 사용하여 AWS Glue ETL Google Analytics 4에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Google Analytics 4 API 버전

v1 베타.

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

다음 관리형 IAM 정책을 사용하여 액세스를 허용할 수도 있습니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 관리 콘솔을 사용할 때 AWS Glue 리소스에 AWS 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

Google Analytics 구성 4

AWS Glue 를 사용하여 Google Analytics 4에서 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 전송하려는 데이터를 수집하는 데이터 스트림이 하나 이상 있는 Google Analytics 계정이 있습니다.
- Google Cloud Platform 계정과 Google Cloud 프로젝트가 있습니다.
- Google Cloud 프로젝트에서 다음을 활성화했습니다APIs.
  - Google Analytics API
  - Google Analytics 관리자 API
  - Google Analytics 데이터 API
- Google Cloud 프로젝트에서 외부 사용자를 위한 OAuth 동의 화면을 구성했습니다. OAuth 동의 화면에 대한 자세한 내용은 Google Cloud Platform 콘솔 도움말의 OAuth 동의 화면 설정을 참조하세요.
- Google Cloud 프로젝트에서 OAuth 2.0 클라이언트 ID를 구성했습니다. 자세한 내용은 OAuth 2.0 설정을 참조하세요.

이러한 요구 사항을 충족하면 Google Analytics 4 계정에 AWS Glue 연결할 준비가 된 것입니다.

Google Analytics 4 연결 구성

Google Sheet 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. AuthorizationCode 권한 부여 유형의 경우:
      - 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Google Analytics 4를 선택합니다.
   b. 연결하려는 Google Analytics 4의 `INSTANCE_URL`을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

`AUTHORIZATION_CODE` 권한 부여 유형입니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Google Analytics 4로 리디렉션합니다. 사용자가 로그인하고 Google Analytics 4 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Google Analytics 4에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Google Analytics 4로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

자세한 내용은 Using Auth 2.0 to Access Google APIs 섹션을 참조하세요.

Google Analytics 4 엔터티에서 읽기

사전 조건

- 읽으려는 Google Analytics 4 객체입니다. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

개체	필터링 가능	한도 지원	주문 지원 기준	선택 지원*	파티셔닝 지원
실시간 보고서	예	예	예	예	아니요
코어 보고서	예	예	예	예	예

예

```
googleAnalytics4_read = glueContext.create_dynamic_frame.from_options(
    connection_type="GoogleAnalytics4",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v1beta"
```

```
}
```

Google Analytics 4 엔터티 및 필드 세부 정보

개체	필드	데이터 형식	지원되는 연산자
코어 보고서	동적 필드		
코어 보고서	차원 필드	String	LIKE, =
코어 보고서	차원 필드	날짜	LIKE, =
코어 보고서	지표 필드	String	>, <, >=, <=, = BETWEEN
코어 보고서	사용자 지정 차원 및 사용자 지정 지표 필드	String	NA
실시간 보고서	appVersion	String	LIKE, =
실시간 보고서	audienceId	String	LIKE, =
실시간 보고서	audienceName	String	LIKE, =
실시간 보고서	구/군/시	String	LIKE, =
실시간 보고서	cityId	String	LIKE, =
실시간 보고서	country	String	LIKE, =
실시간 보고서	countryId	String	LIKE, =
실시간 보고서	deviceCategory	String	LIKE, =
실시간 보고서	eventName	String	LIKE, =
실시간 보고서	minutesAgo	String	LIKE, =
실시간 보고서	platform	String	LIKE, =
실시간 보고서	streamId	String	LIKE, =

개체	필드	데이터 형식	지원되는 연산자
실시간 보고서	streamName	String	LIKE, =
실시간 보고서	unifiedScreenName	String	LIKE, =
실시간 보고서	activeUsers	String	>, <, >=, <=, = BETWEEN
실시간 보고서	변환	String	>, <, >=, <=, = BETWEEN
실시간 보고서	eventCount	String	>, <, >=, <=, = BETWEEN
실시간 보고서	screenPageViews	String	>, <, >=, <=, = BETWEEN

쿼리 파티셔닝

1. 필터 기반 파티션

   Spark에서 동시성을 활용하려는 경우 추가 스파크 옵션 PARTITION_FIELDLOWER_BOUND, UPPER_BOUND, 를 제공할 NUM_PARTITIONS 수 있습니다. 이러한 파라미터를 사용하면 원본 쿼리가 스파크 태스크에서 동시에 실행할 수 있는 하위 쿼리 NUM_PARTITIONS 수로 분할됩니다.

   - PARTITION_FIELD: 쿼리를 분할하는 데 사용할 필드의 이름입니다.
   - LOWER_BOUND: 선택한 파티션 필드의 포함 하한 값입니다.

     날짜의 경우 Spark SQL 쿼리에 사용되는 Spark 날짜 형식을 수락합니다. 유효한 값의 예: "2024-02-06".
   - UPPER_BOUND: 선택한 파티션 필드의 전용 상한 값입니다.
   - NUM_PARTITIONS: 파티션 수입니다.

   예

```
googleAnalytics4_read = glueContext.create_dynamic_frame.from_options(
    connection_type="GoogleAnalytics4",
    connection_options={
```

```
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v1beta",
        "PARTITION_FIELD": "date"
        "LOWER_BOUND": "2022-01-01"
        "UPPER_BOUND": "2024-01-02"
        "NUM_PARTITIONS": "10"
    }
```

2. 레코드 기반 파티션

   Spark에서 동시성을 활용하려는 경우 추가 스파크 옵션을 제공할 NUM_PARTITIONS 수 있습니다. 이러한 파라미터를 사용하면 원본 쿼리가 스파크 태스크에서 동시에 실행할 수 있는 하위 쿼리 NUM_PARTITIONS 수로 분할됩니다.

   - NUM_PARTITIONS: 파티션 수입니다.

   예

```
googleAnalytics4_read = glueContext.create_dynamic_frame.from_options(
    connection_type="GoogleAnalytics4",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v1beta",
        "NUM_PARTITIONS": "10"
    }
```

Google Analytics 4 연결 옵션

다음은 Google Analytics 4의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Google Analytics 4의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Google Analytics 4 Rest API 버전입니다.
- SELECTED_FIELDS(목록<문자열>) - 기본값: 비어 있음(SELECT*). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리를 파티셔닝하는 데 사용할 필드.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(정수) - 읽기에 사용됩니다. (선택 사항)

Google Analytics 4 계정 생성

단계에 따라 Google Analytics 4 계정을 생성합니다. https://support.google.com/analytics/answer/9304153?hl=en

클라이언트 앱 및 OAuth 2.0 보안 인증 생성 단계

자세한 내용은 Google Analytics4 API 설명서 섹션을 참조하세요.

1. 자격 증명으로 Google Analytics 계정에 로그인하여 계정을 생성하고 설정합니다. 그런 다음 관리 > 계정 생성 으로 이동합니다.
2. 속성 생성을 선택하여 생성한 계정의 속성을 생성합니다. 필수 세부 정보로 속성을 설정합니다. 제공된 모든 세부 정보가 생성되면 해당 속성 ID가 생성됩니다.
3. 드롭다운에서 데이터 스트림 > 스트림 추가 > 웹을 선택하여 생성된 속성에 대한 데이터 스트림을 추가합니다. URL 및 기타 필수 필드와 같은 웹 사이트 세부 정보를 제공합니다. 모든 세부 정보를 제공하면 해당 스트림 ID와 측정 ID가 생성됩니다.
4. 측정 ID를 복사하여 웹 사이트에서 Google Analytics를 설정하고 웹 사이트의 구성에 를 추가합니다.
5. 보고서로 이동하여 필요한 보고서를 생성하여 Google Analytics에서 보고서를 생성합니다.
6. console.cloud.google.com 이동하여 앱을 승인하고 Google Analytics 데이터 를 검색한 API다음 를 활성화합니다API.

    1. API 및 서비스 페이지로 이동하여 자격 증명 > OAuth 2.0 클라이언트 설정 IDs을 선택합니다.
    2. 리디렉션 을 URL 추가하여 AWS Glue 리디렉션을 제공합니다URL.
7. 연결을 생성하는 데 추가로 필요한 클라이언트 ID와 클라이언트 암호를 복사합니다.

제한 사항 및 고려 사항

다음은 Google Analytics 4 커넥터의 제한 사항입니다.

- 코어 보고서 엔터티의 경우 9개의 차원 필드와 10개의 지표 필드만 요청에 전송할 수 있습니다. 허용되는 필드 수를 초과하면 요청이 실패하고 커넥터에 오류 메시지가 표시됩니다.
- 실시간 보고서 엔터티의 경우 요청에서 4개의 차원 필드만 보낼 수 있습니다. 허용되는 필드 수를 초과하면 요청이 실패하고 커넥터에 오류 메시지가 표시됩니다.
- Google Analytics 4는 베타 버전 프리 도구이므로 새로운 기능, 엔터티 개선 사항, 새 필드 추가 및 기존 필드 사용 중단에 대한 정기적인 업데이트가 있을 예정입니다.
- 코어 보고서 필드는 동적으로 채워지므로 필드의 추가, 감가상각 및 이름 변경과 필드에 대한 새로운 제한 부과를 언제든지 수행할 수 있습니다.
- 기본 시작 날짜는 30일이고 종료 날짜는 어제(현재 날짜 하루 전)이며, 사용자가 값을 설정한 경우 필터 표현식 코드에서 또는 흐름이 증분인 경우 이 날짜가 재정의됩니다.
- 설명서에 따르면 Real-Time 보고서 엔터티는 요청에서 제한이 전달되지 않으면 10,000개의 레코드를 반환하고, 그렇지 않으면 는 요청 수에 관계없이 요청당 최대 250,000개의 행을 API 반환합니다. 자세한 내용은 Google Analytics 설명서의 메서드: 속성을runRealtimeReport 참조하세요.
- 실시간 보고서 엔터티는 페이지 매김을 지원하지 않으므로 레코드 기반 파티션을 지원하지 않습니다. 또한 정의된 기준을 충족하는 필드가 없으므로 필드 기반 파티션을 지원하지 않습니다.
- 요청에서 전달할 수 있는 필드 수의 제한으로 인해 지정된 한도 내에서 기본 차원 및 지표 필드를 설정하고 있습니다. '모두 선택'을 선택하면 미리 결정된 필드의 데이터만 검색됩니다.
  - 코어 보고서
    - 의 제한에 따라 SAAS 요청은 최대 9차원 및 최대 10개의 지표만 허용됩니다(즉, 요청은 최대 19개의 필드(지표 + 차원)를 포함할 수 있습니다.
    - 구현에 따라 - 사용자가 SELECT_ALL 또는 25개 이상의 선택한 필드를 사용하는 경우 요청에서 기본 필드가 전달됩니다.
    - 핵심 보고서 - 'country', 'city', 'eventName', 'cityId', 'browser', 'date', 'currencyCode', 'deviceCategory', 'transactionId', 'Active1DayUsers', 'active28DayUsers', 'active7DayUsers', 'activeUsers', 'averagePurchaseRevenue'', 'averageRevenuePerUser', 'averageSessionDuration', 'engagedSessions', 'eventCount'', 'engagementRate''에 대한 기본 필드로 간주됩니다.
  - 실시간 보고서
    - SAAS 요청에 대한 제한에 따라 최대 4차원이 허용됩니다.
    - 사용자가 SELECT_ALL 또는 선택한 필드를 15개 이상 전달하면 기본 필드가 요청에 전달됩니다.
    - 다음 필드는 RealTime 보고서 - '국가', 'deviceCategory', '도시', 'cityId', 'activeUsers', '전환', 'eventCount', 'screenPageViews'에 대한 기본 필드로 간주됩니다.

- Core-Report 엔터티에서 파티션 온 날짜 필드와 필터링 온 startDate 이 동시에 있는 경우. 이 경우 dateRange 값이 startDate 필터 값으로 재정의되지만 파티션이 항상 우선 순위여야 하므로 파티션 날짜 필드가 이미 있는 경우 startDate 필터를 삭제합니다.
- 이제 cohortSpecs 는 코어 보고서 요청 본문의 일부이므로 cohortSpec 속성에 대한 지원을 포함하도록 현재 코어 보고서 엔터티를 개선했습니다. cohortSpecs 요청 본문에서 거의 모든 필드에 사용자 입력이 필요합니다. 이를 해결하기 위해 이러한 속성/필드에 대한 기본값을 설정하고 필요한 경우 사용자가 이러한 값을 재정의할 수 있도록 프로비저닝을 제공했습니다.

FieldName	기본값	기본값을 재정의하는 filterPredicate 옵션을 전달할 샘플 쿼리
startDate	현재 날짜로부터 30일 전	“'2023-05-09'에서 “2023-05-10” startDate 사이
endDate	현재 날짜로부터 1일 전	“'2023-05-09'에서 “2023-05-10” startDate 사이
startOffset	0	startOffset=2
endOffset	1	endOffset=10
세부 수준	DAILY	세분성='WEEKLY'

- 이러한 모든 필터를 한 번에 함께 전달하거나 다른 필터와 함께 전달할 수도 있습니다.
  - 예 1 - filterPredicate: “2023-05-09”과 “2023-05-10” startDate =ANDstartOffset1 AND endOffset=2 세AND분성=” WEEKLY사이
  - 예 2 - filterPredicate: city=“xyz” AND startOffset=1 AND endOffset=2 세AND분성=”WEEKLY”
- 코호트 요청에서:
  - 요청에서 cohortNthMonth"가 전달되면 내부 세분화 값이 'MONTHLY'로 설정됩니다.
  - 마찬가지로 ‘cohortNthWeek’가 전달되면 세분화 값이 ‘WEEKLY’로 설정됩니다.
  - 또한 ‘cohortNthDay’의 경우 세분화 값은 ‘DAILY’로 설정됩니다. 자세한 내용은 다음을 참조하세요.
    - https://developers.google.com/analytics/devguides/reporting/data/v1/advanced
    - https://developers.google.com/analytics/devguides/reporting/data/v1/rest/v1beta/CohortSpec

- 프로비저닝은 사용자가 dateRange 및 세분화된 기본값을 재정의할 수 있도록 제공됩니다. 위의 표를 참조하세요.

AWS Glue Studio에서 Google BigQuery에 연결

> **Note**
>
> AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에 있는 Google BigQuery의 테이블에서 읽고 쓸 수 있습니다. 프로그래밍 방식으로 AWS Glue 작업을 통해 Google BigQuery를 구성하려면 BigQuery 연결을(를) 참조하세요.

AWS Glue Studio에는 BigQuery에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스가 있습니다.

AWS Glue Studio에서 Google BigQuery에 대한 연결을 생성할 때 통합 연결이 생성됩니다. 자세한 내용은 고려 사항 단원을 참조하십시오.

특정 형식의 자격 증명인 `{"credentials": "base64 encoded JSON"}`로 보안 암호를 만드는 대신 이제 Google BigQuery에 대한 통합 연결을 통해 Goolge BigQuery의 JSON을 직접 포함하는 보안 암호를 생성할 수 있습니다(`{"type": "service-account", ...}`).

주제

- BigQuery 연결 생성
- BigQuery 소스 노드 생성
- BigQuery 대상 노드 생성
- 고급 옵션

BigQuery 연결 생성

AWS Glue에서 Google BigQuery에 연결하려면 AWS Secrets Manager 보안 암호에서 Google Cloud Platform 보안 인증 정보를 생성하고 저장한 다음 해당 보안 암호를 Google AWS Glue BigQuery 연결에 연결해야 합니다.

BigQuery에 대한 연결 구성하기:

1. Google Cloud Platform에서 관련 리소스를 생성하고 식별합니다.

   - 연결하려는 BigQuery 테이블이 포함된 GCP 프로젝트를 생성하거나 식별합니다.
   - BigQuery API를 활성화합니다. 자세한 내용은 BigQuery Storage Read API를 사용하여 테이블 데이터 읽기를 참조하세요.

2. Google Cloud Platform에서 서비스 계정 보안 인증 정보를 생성하고 내보냅니다.

   BigQuery 보안 인증 마법사를 사용하여 보안 인증 정보 생성하기 단계를 신속하게 처리할 수 있습니다.

   GCP에서 서비스 계정을 생성하려면 서비스 계정 생성하기에서 제공되는 튜토리얼을 따르세요.

   - 프로젝트를 선택할 때 BigQuery 테이블이 포함된 프로젝트를 선택합니다.
   - 서비스 계정의 GCP IAM 역할을 선택할 때 BigQuery 작업을 실행하여 BigQuery 테이블을 읽고, 쓰고, 생성할 수 있는 적절한 권한을 부여하는 역할을 추가하거나 생성하세요.

   서비스 계정의 보안 인증 정보를 생성하려면 서비스 계정 키 생성하기에서 제공되는 튜토리얼을 따르세요.

   - 키 유형을 선택할 때 JSON을 선택합니다.

   이제 서비스 계정의 보안 인증 정보가 포함된 JSON 파일을 다운로드했어야 합니다. 예를 들면 다음과 같아야 합니다.

```
{
  "type": "service_account",
  "project_id": "*****",
  "private_key_id": "*****",
  "private_key": "*****",
  "client_email": "*****",
  "client_id": "*****",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "*****",
  "universe_domain": "googleapis.com"
}
```

3. AWS Secrets Manager에서, 다운로드한 자격 증명 파일을 사용하여 보안 암호를 생성합니다. 일반 텍스트 탭을 선택하고 JSON 형식 파일 콘텐츠를 붙여넣을 수 있습니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
4. AWS Glue 데이터 카탈로그에서 https://docs.aws.amazon.com/glue/latest/dg/console-connections.html의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때 Google BigQuery를 선택합니다.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.
5. AWS Glue 작업 권한과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
6. AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

BigQuery 소스 노드 생성

필수 전제 조건

- BigQuery 유형 AWS Glue 데이터 카탈로그 연결
- 연결에 사용되는 Google BigQuery 자격 증명의 AWS Secrets Manager 보안 암호.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 읽으려는 테이블 및 해당 Google Cloud 프로젝트의 이름 및 데이터세트.

BigQuery 데이터 소스 추가

데이터 소스 - BigQuery 노드 추가하기:

1. BigQuery 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 BigQuery 연결 생성을 선택합니다. 자세한 내용은 커넥터 및 연결 사용 개요를 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. 읽으려는 BigQuery 데이터를 식별한 다음 BigQuery 소스 옵션을 선택합니다
   - 단일 테이블 선택 — 테이블에서 모든 데이터를 가져올 수 있습니다.
   - 사용자 지정 쿼리 입력 — 쿼리를 제공하여 검색할 데이터를 사용자 지정할 수 있습니다.

3. 읽고 싶은 데이터를 설명하세요

   (필수 사항) 상위 프로젝트를 테이블이 포함된 프로젝트로 설정하거나 해당하는 경우 청구 대상 상위 프로젝트로 설정합니다.

   단일 테이블을 선택한 경우 테이블을 `[dataset].[table]` 형식의 Google BigQuery 테이블 이름으로 설정합니다

   쿼리를 선택한 경우 쿼리에 제공합니다. 쿼리에서 `[project].[dataset].[tableName]` 형식으로 정규화된 테이블 이름이 있는 테이블을 참조하세요.

4. BigQuery 속성을 제공합니다

   단일 테이블을 선택한 경우 추가 속성을 제공하지 않아도 됩니다.

   쿼리를 선택한 경우 다음 사용자 지정 Google BigQuery 속성을 제공해야 합니다.

   - `viewsEnabled`를 true로 설정합니다.
   - `materializationDataset`을(를) 데이터세트에 설정합니다. AWS Glue 연결을 통해 제공된 보안 인증 정보로 인증된 GCP 주체는 이 데이터 세트에 테이블을 생성할 수 있어야 합니다.

BigQuery 대상 노드 생성

필수 전제 조건

- BigQuery 유형 AWS Glue 데이터 카탈로그 연결
- 연결에 사용되는 Google BigQuery 자격 증명의 AWS Secrets Manager 보안 암호.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 쓰려는 테이블 및 해당 Google Cloud 프로젝트의 이름 및 데이터세트.

BigQuery 데이터 대상 추가

데이터 대상 - BigQuery 노드 추가하기:

1. BigQuery 데이터 대상의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 BigQuery 연결 생성을 선택합니다. 자세한 내용은 커넥터 및 연결 사용 개요를 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.

2. 쓰려는 BigQuery 테이블을 식별한 다음 쓰기 방법을 선택합니다.

   - 직접 — BigQuery Storage 쓰기 API를 사용하여 BigQuery에 직접 글을 씁니다.
   - 간접 — Google Cloud Storage에 쓴 다음 BigQuery에 복사합니다.

   간접적으로 쓰려는 경우 임시 GCS 버킷과 함께 대상 GCS 위치를 제공하세요. AWS Glue 연결에 추가 구성을 제공해야 합니다. 자세한 내용은 Google BigQuery를 통한 간접 쓰기 사용을 참조하세요.
3. 읽고 싶은 데이터를 설명하세요

   (필수 사항) 상위 프로젝트를 테이블이 포함된 프로젝트로 설정하거나 해당하는 경우 청구 대상 상위 프로젝트로 설정합니다.

   단일 테이블을 선택한 경우 테이블을 `[dataset].[table]` 형식의 Google BigQuery 테이블 이름으로 설정합니다

고급 옵션

BigQuery 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue을(를) 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

AWS Glue 개발자 설명서의 BigQuery 연결 옵션 참조를 참조하세요.

Google Search Console에 연결

Google Search Console은 웹사이트 소유자가 Google이 사이트를 어떻게 보는지 모니터링하고 유기적 존재감을 최적화하기 위해 사용할 수 있는 무료 플랫폼입니다. 여기에는 참조 도메인, 모바일 사이트 성능, 검색 결과, 트래픽이 가장 많은 쿼리 및 페이지 보기가 포함됩니다. Google Search Console 사용자인 경우 Google Search Console 계정에 AWS Glue를 연결할 수 있습니다. Google Search Console을 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Google Search Console에서 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송합니다.

주제

- AWS Glue의 Google Search Console 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Google Search Console 구성
- Google Search Console 연결 구성

- Google Search Console 엔터티에서 읽기
- Google Search Console 연결 옵션
- Google Search Console 제한 사항

AWS Glue의 Google Search Console 지원

AWS Glue에서는 다음과 같이 Google Search Console을 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Google Search Console에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Google Search Console API 버전

다음 Google Search Console API 버전이 지원됩니다.

- v3

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
     "Statement": [
      {
             "Effect": "Allow",
             "Action": [
                 "glue:ListConnectionTypes",
                 "glue:DescribeConnectionType",
                 "glue:RefreshOAuth2Tokens",
                 "glue:ListEntities",
                 "glue:DescribeEntity"
             ],
             "Resource": "*"
```

```
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Google Search Console 구성

AWS Glue를 사용하여 Google Search Console에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Google Search Console 계정이 있습니다.
- Google Cloud Platform 관리자 계정을 만들고 Google Cloud 프로젝트를 생성했습니다.
- Google Cloud 프로젝트에서 Google Search Console API를 활성화했습니다.
- Google Cloud 프로젝트에서 외부 사용자를 위한 OAuth 동의 화면을 구성했습니다. 자세한 내용은 Google Cloud Platform Console 도움말의 Setting up your OAuth consent screen을 참조하세요.
- Google Cloud 프로젝트에서 OAuth 2.0 클라이언트 ID를 구성했습니다. AWS Glue가 계정에 인증된 직접 호출을 할 때 데이터에 안전하게 액세스하는 데 사용하는 클라이언트 자격 증명은 OAuth 2.0 설정을 참조하세요.

이러한 요구 사항을 충족하면 Google Search Console 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Google Search Console에서 다른 작업을 수행하지 않아도 됩니다.

Google Search Console 연결 구성

Google Search Console에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다. 권한 부여 유형은 AWS Glue에서 Google Search Console과 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.
- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Google Search Console에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Google Search Console로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Google Search Console 설명서는 Using OAuth 2.0 to Access Google APIs를 참조하세요.

Google Search Console 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Google Search Console을 선택합니다.
   b. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
```

```
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

c. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

d. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Google Search Console 엔터티에서 읽기

사전 조건

읽으려는 Google Ads 객체입니다. 객체 이름이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Search Analytics	예	예	아니요	예	아니요
Sites	아니요	아니요	아니요	예	아니요
Sitemaps	아니요	아니요	아니요	예	아니요

예시:

```
googleSearchConsole_read = glueContext.create_dynamic_frame.from_options(
    connection_type="googlesearchconsole",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
```

```
        "API_VERSION": "v3"
    }
```

Google Search Console 엔터티 및 필드 세부 정보:

Google Search Console에서는 지원되는 엔터티에 대해 메타데이터를 동적으로 가져오도록 엔드포인트를 제공합니다. 따라서 운영자 지원은 데이터 유형 수준에서 캡처됩니다.

엔터티	필드	데이터 유형	지원되는 연산자	참고
Search Analytics	keys	List	N/A	
	clicks	Double	N/A	
	impressions	Double	N/A	
	ctr	BigDecimal	N/A	BigDecimal 데이터 형식의 경우 '0' 값은 '0E-18' 형식입니다.
	position	Double	N/A	
	start_end_date	Date	BETWEEN	`start_end_date` 의 기본 값은 <현재 날짜로부터 30일 전>과 <어제: 즉, 현재 날짜로부터 1일 전> 사이입니다. 참고: UTC 날짜 값을 전달할 것으로 예상합니다. 예: start_end_date가 '2022-01-01'와

엔터티	필드	데이터 유형	지원되는 연산자	참고
				'2024-09-09' 사이
	country	String	EQUAL_TO, NOT_EQUAL_TO, CONTAINS	유효한 값은 'IND', 'CAN' 등입니다.
	type	String	EQUAL_TO, NOT_EQUAL_TO	유효한 값은 'discover', 'googleNews', 'news', 'image', 'video', 'web'입니다.
	searchAppearance	String	EQUAL_TO, NOT_EQUAL_TO, CONTAINS	유효한 값 목록은 검색 표시를 참조하세요.
	device	String	EQUAL_TO, NOT_EQUAL_TO, CONTAINS	유효한 값은 'DESKTOP', 'MOBILE', 'TABLET'입니다.
	dimensions	String	EQUAL_TO	유효한 값은 'country', 'device'입니다.
	page	String	EQUAL_TO, NOT_EQUAL_TO, CONTAINS	
	query	String	EQUAL_TO, NOT_EQUAL_TO, CONTAINS	
	dataState	String	EQUAL_TO	유효한 값은 'all' 및 'final'입니다.

엔터티	필드	데이터 유형	지원되는 연산자	참고
Sites	siteUrl	String	N/A	
	permissionLevel	String	N/A	
Sitemaps	path	String	N/A	
	type	String	N/A	
	lastSubmitted	DateTime	N/A	
	isPending	Boolean	N/A	
	isSitemapsIndex	Boolean	N/A	
	lastDownloaded	DateTime	N/A	
	warnings	Long	N/A	
	errors	Long	N/A	
	contents	List	N/A	

> **Note**
>
> 필터에 대한 유효한 값의 업데이트된 목록은 Google Search Console API 문서를 참조하세요. `start_end_date` 필드는 `start_date` 및 `end_date`의 조합입니다.

분할 쿼리

필터 기반 분할 및 레코드 기반 분할은 지원되지 않습니다.

Google Search Console 연결 옵션

다음은 Google Search Console의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Google Search Console에서의 객체 이름입니다.

- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Google Search Console Rest API 버전입니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열)-기본값: 'start_end_date <현재 날짜로부터 30일 전>과 <어제: 즉, 현재 날짜로부터 1일 전> 사이'. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열)-기본값: 'start_end_date <현재 날짜로부터 30일 전>과 <어제: 즉, 현재 날짜로부터 1일 전> 사이' 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- INSTANCE_URL(문자열) - 읽기에 사용됩니다. 유효한 Google Search Console 인스턴스 URL입니다.

Google Search Console 제한 사항

다음은 Google Search Console의 제한 사항 또는 참고 사항입니다.

- Google Search Console은 API에 사용량 한도를 적용합니다. 자세한 내용은 사용 제한을 참조하세요.
- Search Analytics 엔터티에 대한 필터가 전달되지 않으면 API는 지정된 기본 날짜 범위 내에 전체 사이트의 모든 클릭, 노출, CTR 및 기타 데이터를 합산하여 단일 레코드로 표시합니다.
- 데이터를 더 작은 세그먼트로 분류하려면 쿼리에 차원을 도입해야 합니다. 차원은 API에 데이터를 분할하는 방법을 알려줍니다.
  - 예를 들어, filterPredicate: dimensions="country"를 추가하면 지정된 기간 동안 사이트가 트래픽을 수신한 각 국가에 대해 하나의 레코드를 가져옵니다.
  - 여러 차원을 전달하는 예: filterPredicate: dimensions="country" AND dimensions="device" AND dimensions="page". 이 경우 이러한 세 가지 차원의 고유한 조합마다 응답에 행이 하나씩 표시됩니다.
- 기본값은 start_end_date 및 dataState 필드에 설정됩니다.

필드 이름	기본 필터 표현식	기본값을 재정의하는 표현식
start_end_date	start_end_date between <현재 날짜로부터 30일 전> AND <어제 즉, 현재 날짜로부터 1일 전>	start_end_date between "2024-01-01" AND "2024-05-05"

필드 이름	기본 필터 표현식	기본값을 재정의하는 표현식
dateState	dataState="all"	dataState="final"

Google Sheets에 연결

Google Sheets는 대량의 데이터를 구성하고, 사용자 지정 보고서를 생성하고, 계산을 자동화하고, 다른 사용자와 협업할 수 있는 온라인 스프레드시트 소프트웨어입니다.Google Sheets 사용자인 경우 Google Sheets 계정에 AWS Glue 연결할 수 있습니다. 그런 다음 Google Sheets를 ETL 작업의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Google Sheets와 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue Google Sheets 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Google 시트 구성
- Google Sheets 연결 구성
- Google Sheets 엔터티에서 읽기
- Google Sheets 연결 옵션
- Google Sheets에 대한 권한 부여 코드 OAuth 흐름 설정
- Google Sheets 커넥터의 제한 사항

AWS Glue Google Sheets 지원

AWS Glue 는 다음과 같이 Google Sheets를 지원합니다.

소스로 지원되나요?

예. 작업을 사용하여 AWS Glue ETL Google Sheets에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Google Sheets API 버전

Google Sheets API v4 및 Google Drive API v3

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Google 시트 구성

AWS Glue 를 사용하여 Google Sheets에서 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호가 있는 Google Sheets 계정이 있습니다.
- Google Sheets 계정에 API 액세스할 수 있습니다. Google Sheets는 추가 비용 없이 사용할 API 수 있습니다.
- Google Sheets 계정을 사용하면 연결된 앱을 설치할 수 있습니다. 이 기능에 대한 액세스 권한이 없는 경우 Google Sheets 관리자에게 문의하세요.

이러한 요구 사항을 충족하면 Google Sheets 계정에 AWS Glue 연결할 준비가 된 것입니다.

Google Sheets 연결 구성

Google Sheet 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. AuthorizationCode 권한 부여 유형의 경우:
      - 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 데이터 소스를 선택할 때 Google Sheets를 선택합니다.
   b. Google Sheets 환경을 제공합니다.
      i. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
      ii. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
```

```
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

AUTHORIZATION_CODE 권한 부여 유형

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Google Sheets로 리디렉션합니다. 사용자가 로그인하고 Google Sheets 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Google Sheets에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Google Sheets로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

자세한 내용은 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성의 퍼블릭 Google Sheets 설명서를 참조하세요.

Google Sheets 엔터티에서 읽기

사전 조건

- 읽으려는 Google SpreadSheet입니다. 스프레드시트의 SpreadSheet ID 및 tabName이 필요합니다.

Google Sheets 엔터티 및 필드 세부 정보:

엔터티	데이터 형식	지원되는 연산자
스프레드시트	String	N/A(필터링이 지원되지 않음)

예

```
googleSheets_read = glueContext.create_dynamic_frame.from_options(
    connection_type="googlesheets",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "{SpreadSheetID}#{SheetTabName}",
        "API_VERSION": "v4"
    }
```

분할 쿼리

레코드 기반 분할에 한해 Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션으로 NUM_PARTITIONS를 제공할 수 있습니다. 이 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

NUM_PARTITIONS를 사용한 예

```
googlesheets_read = glueContext.create_dynamic_frame.from_options(
    connection_type="googlesheets",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "{SpreadSheetID}#{SheetTabName}",
        "API_VERSION": "v4",
        "NUM_PARTITIONS": "10"
    }
```

Google Sheets 연결 옵션

다음은 Google Sheets의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Google SheetssheetTabName의 SpreadSheet ID 및 . 예: {SpreadSheetID}#{SheetTabName}.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Google Sheets Rest API 버전입니다.
- SELECTED_FIELDS(목록<문자열>) - 기본값: 비어 있음(SELECT*). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Google Sheets에 대한 권한 부여 코드 OAuth 흐름 설정

사전 조건

- Google Sheets 앱을 사용하기 위해 로그인할 수 있는 Google 계정입니다. Google 계정에서 Google Sheets에는 전송하려는 데이터가 포함되어 있습니다.
- Google Cloud Platform 계정 및 Google Cloud 프로젝트. 자세한 내용은 Google Cloud 프로젝트 생성을 참조하세요.

Google 계정을 설정하고 OAuth 2.0 자격 증명을 가져오려면:

1. Google Cloud 프로젝트가 설정되면 프로젝트APIs에서 Google Sheets API 및 Google Drive를 활성화합니다. 활성화 단계는 Google Cloud Platform용 API 콘솔 도움말APIs의 활성화 및 비활성화를 참조하세요.
2. 그런 다음 외부 사용자를 위한 OAuth 동의 화면을 구성합니다. OAuth 동의 화면에 대한 자세한 내용은 Google Cloud Platform 콘솔 도움말의 OAuth 동의 화면 설정을 참조하세요.
3. OAuth 동의 화면에서 다음 범위를 추가합니다.

   - Google Sheets API 읽기 전용 범위
   - Google Drive API 읽기 전용 범위

   이러한 범위에 대한 자세한 내용은 Google Identity 설명서APIs의 OAuth2.0 Scopes for Google을 참조하세요.
4. OAuth 2.0 클라이언트 ID 및 보안 암호를 생성합니다. 이 클라이언트 ID를 생성하는 단계는 Google Cloud Platform 콘솔 도움말의 OAuth 2.0 설정을 참조하세요.

   2.0 클라이언트 ID에는 하나 이상의 승인된 OAuth 리디렉션이 있어야 합니다URLs.

   리디렉션 URLs 형식은 다음과 같습니다.

   - https://<aws-region>.console.aws.amazon.com/gluestudio/oauth
5. OAuth 2.0 클라이언트 ID 설정에서 클라이언트 ID와 클라이언트 암호를 기록해 둡니다.

Google Sheets 커넥터의 제한 사항

다음은 Google Sheets 커넥터의 제한 사항입니다.

- Google Sheets 커넥터는 필터를 지원하지 않습니다. 따라서 필터 기반 분할은 지원되지 않습니다.
- 레코드 기반 분할에는 SAAS별로 정확한 레코드 수를 반환하는 조항이 없습니다. 결과적으로 빈 레코드가 있는 파일이 생성되는 시나리오가 발생할 수 있습니다.
- Google Sheets 커넥터는 필터 기반 분할을 지원하지 않으므로 `partitionField`, `lowerbound` 및 `upperbound`는 유효한 연결 옵션이 아닙니다. 이러한 옵션이 제공되면 AWS Glue 작업이 실패할 것으로 예상됩니다.
- 데이터 처리 문제를 방지하려면 시트의 첫 번째 행을 헤더 행으로 지정하는 것이 중요합니다.
  - 헤더 행을 제공하지 않으면 시트에 첫 번째 행이 비어 있는 데이터가 포함된 경우 헤더 행이 `Unnamed:1`, `Unnamed:2`, `Unnamed:3` 등으로 바뀝니다.
  - 헤더 행을 제공하면 빈 열 이름이 `Unnamed:<number of column>`으로 바뀝니다. 예를 들어 헤더 행이 `['ColumnName1', 'ColumnName2', '', '', 'ColumnName5', 'ColumnName6']`이면 `['ColumnName1', 'ColumnName2', 'Unnamed:3', 'Unnamed:4', 'ColumnName5', 'ColumnName6'].`으로 됩니다.
- Google Sheets 커넥터는 증분 전송을 지원하지 않습니다.
- Google Sheets 커넥터는 문자열 데이터 유형만 지원합니다.
- 시트의 중복 헤더는 숫자 접미사를 사용하여 반복적으로 이름이 바뀝니다. 중복 헤더의 이름을 바꾸는 동안 사용자가 제공한 헤더 이름이 우선합니다. 예를 들어 헤더 행이 ["Name", "", "Name", null, "Unnamed:6", ""]인 경우 ["Name", "Unnamed:2", "Name1", "Unnamed:4", "Unnamed:6", "Unnamed:61"]로 변경됩니다.
- Google Sheets 커넥터는 tabName에 공백을 지원하지 않습니다.
- 폴더 이름에는 다음과 같은 특수 문자를 사용할 수 없습니다.
  - #
  - /

에 연결 HubSpot

HubSpot의 CRM 플랫폼에는 마케팅, 판매, 콘텐츠 관리 및 고객 서비스에 필요한 모든 도구와 통합이 있습니다.

- Marketing Hub - 트래픽을 늘리고, 더 많은 방문자를 전환하고, 대규모로 전체 인바운드 마케팅 캠페인을 실행하는 데 도움이 되는 마케팅 소프트웨어입니다.
- Sales Hub - 잠재 고객에 대한 심층적인 인사이트를 얻고, 보유한 작업을 자동화하고, 더 많은 거래를 더 빠르게 성사시키는 데 도움이 되는 영업 CRM 소프트웨어입니다.

- Service Hub - 고객과 연결하고, 기대치를 초과 달성하며, 비즈니스를 성장시키는 홍보자로 전환하는 데 도움이 되는 고객 서비스 소프트웨어입니다.
- Operations Hub - 앱을 동기화하고, 고객 데이터를 정리 및 생성하고, 프로세스를 자동화하는 운영 소프트웨어로 모든 시스템과 팀이 더 잘 협력할 수 있습니다.

주제

- HubSpot의 AWS Glue 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- HubSpot 구성
- HubSpot 연결 구성
- HubSpot 엔터티에서 읽기
- HubSpot 연결 옵션
- HubSpot 커넥터의 제한 사항 및 참고 사항

HubSpot의 AWS Glue 지원

AWS Glue는 다음과 같이 HubSpot을 지원합니다.

소스로 지원되나요?

예 - 동기화 및 비동기. AWS Glue ETL 작업을 사용하여 HubSpot에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 HubSpot API 버전

다음 HubSpot API 버전이 지원됩니다.

- v1
- v2
- v3
- v4

버전별 엔터티 지원은 동기화 소스에 대해 지원되는 엔터티 및 비동기 소스에 대해 지원되는 엔터티: 섹션을 참조하세요.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려면 다음 관리형 IAM 정책을 대신 사용합니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS 관리 콘솔을 사용할 때 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

HubSpot 구성

AWS Glue를 사용하여 HubSpot에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- HubSpot 계정이 있습니다. 자세한 내용은 HubSpot 계정 생성 섹션을 참조하세요.
- HubSpot 계정이 API 액세스에 대해 활성화되어 있습니다.
- 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하기 위해 사용하는 클라이언트 자격 증명을 제공하는 앱이 HubSpot 개발자 계정 아래에 있어야 합니다. 자세한 내용은 HubSpot 개발자 앱 생성 섹션을 참조하세요.

이러한 요구 사항을 충족하면 HubSpot 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 HubSpot에서 다른 작업을 수행하지 않아도 됩니다.

HubSpot 계정 생성

HubSpot 계정을 생성하려면 다음을 수행합니다.

1. HubSpot CRM SignUp URL로 이동합니다.
2. 이메일 주소를 입력하고 이메일 확인을 선택합니다(또는 Google, Microsoft 또는 Apple 계정으로 가입하도록 선택할 수 있음).
3. 받은 편지함에서 HubSpot의 확인 코드를 확인합니다.
4. 6자리 확인 코드를 입력하고 다음을 클릭합니다.
5. 암호를 입력하고 다음을 클릭합니다.
6. 이름과 성을 입력하고 다음을 클릭하거나 Google로 가입 링크를 사용하여 가입합니다.
7. 업종을 입력하고 다음을 클릭합니다.
8. 직무 역할을 입력하고 다음을 클릭합니다.
9. 회사 이름을 입력하고 다음을 클릭합니다.
10. 회사 규모(회사에서 근무하는 직원 수)를 선택하고 다음을 클릭합니다.
11. 회사 웹 사이트를 입력하고 다음을 클릭합니다.
12. 데이터를 호스팅할 위치(미국 또는 유럽)를 선택하고 계정 생성을 클릭합니다.
13. 계정 생성 목적을 선택하고 다음을 클릭합니다.
14. Google 계정 연결을 선택하거나 연락처를 직접 추가하여 연락처를 HubSpot 계정에 연결하도록 선택합니다.
15. Google 계정 연결 옵션을 선택하여 연락처를 연결하고 HubSpot 계정 사용을 시작한 경우 Google 계정에 로그인합니다.

HubSpot 개발자 앱 생성

앱 개발자 계정은 앱, 통합 및 개발자 테스트 계정을 생성하고 관리하기 위한 것입니다. 또한 App Marketplace 목록을 생성하고 관리할 수 있습니다. 그러나 앱 개발자 계정 및 관련 테스트 계정은 표준 HubSpot 계정에 연결되지 않습니다. 다른 HubSpot 계정과 데이터 또는 자산을 동기화할 수 없습니다. 클라이언트 ID 및 클라이언트 보안 암호를 가져오려면 개발자 계정을 생성합니다.

1. https://developers.hubspot.com/으로 이동합니다.
2. 개발자 계정 생성을 선택하고 아래로 스크롤합니다.
3. 앱 개발자 계정, 프라이빗 앱 계정 또는 CMS 개발자 샌드박스 계정을 생성할지 묻는 메시지가 표시됩니다. 앱 개발자 계정 생성을 선택합니다.
4. 이미 HubSpot으로 계정을 생성했으므로 이 사용자로 계속을 선택할 수 있습니다.
5. 가입 시작을 클릭합니다.
6. 직무 역할을 입력하고 다음을 클릭합니다.
7. 개발자 계정의 이름을 지정하고 다음을 클릭한 다음, 건너뛰기를 클릭합니다.
8. 앱 생성을 선택합니다.
9. 앱이 생성되면 인증을 선택합니다.
10. 인증에서 클라이언트 ID 및 클라이언트 암호를 메모합니다.
11. 리전별 리디렉션 URL을 https://*<aws-region>*.console.aws.amazon.com/gluestudio/oauth로 추가합니다. 예를 들어 us-east-1 리전에 https://us-east-1.console.aws.amazon.com/gluestudio/oauth를 추가합니다.
12. 아래로 스크롤하여 범위를 찾습니다. ‘CRM’ 및 ‘Standard’ 제목에서 두 가지 종류의 범위를 선택해야 합니다.
13. 다음 범위를 추가합니다.

```
content
automation
oauth
crm.objects.owners.read
forms
tickets
crm.objects.contacts.write
e-commerce
crm.schemas.custom.read
crm.objects.custom.read
sales-email-read
crm.objects.custom.write
```

```
crm.objects.companies.write
crm.lists.write
crm.objects.companies.read
crm.lists.read
crm.objects.deals.read
crm.objects.deals.write
crm.objects.contacts.read
```

14.저장을 클릭하면 이제 개발 계정을 사용할 준비가 됩니다.

15.위로 스크롤하여 클라이언트 ID를 찾습니다.

16.동일한 페이지에서 표시를 클릭하여 클라이언트 암호를 가져옵니다.

HubSpot 개발자 테스트 계정 생성

앱 개발자 계정 내에서 개발자 테스트 계정을 생성하여 실제 HubSpot 데이터에 영향을 주지 않고 앱 및 통합을 테스트할 수 있습니다. 개발자 테스트 계정은 프로덕션 계정을 미러링하지 않고 Marketing, Sales, CMS 및 Operations Hub의 엔터프라이즈 버전 90일 평가판에 액세스하여 대부분의 HubSpot 도구 및 API를 테스트할 수 있습니다.

1. 홈을 클릭하세요.
2. 테스트 계정 생성을 클릭합니다.
3. 앱 테스트 계정 생성을 클릭합니다.
4. 새 창이 나타납니다. 앱 테스트 계정 이름을 입력하고 생성을 클릭합니다.

   이제 앱 테스트 계정이 생성됩니다.

> **Note**
>
> 개발자 계정은 API 통합과 같은 개발 활동과 관련이 있으며, 앱 테스트 계정은 개발자 계정에서 생성 또는 가져오는 데이터를 보는 데 사용됩니다.

HubSpot 연결 구성

HubSpot에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.

연결을 생성하는 사용자는 HubSpot 클라이언트 애플리케이션에 대한 클라이언트 ID 및 클라이언트 보안 암호와 같은 OAuth 관련 정보를 제공해야 합니다. AWS Glue 콘솔은 사용자를 HubSpot으로 리디렉션합니다. 사용자가 로그인하고 HubSpot 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 HubSpot에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 HubSpot으로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 HubSpot 설명서는 Public apps을 참조하세요.

HubSpot 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 HubSpot을 선택합니다.
   b. HubSpot 환경을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

HubSpot 엔터티에서 읽기

사전 조건

읽으려는 HubSpot 객체. 객체 이름(예: 연락처 또는 태스크)이 필요합니다. 다음 표에는 동기화 소스에 대해 지원되는 엔터티가 나와 있습니다.

동기화 소스에 대해 지원되는 엔터티

개체	API 버전	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Campaigns	v1	아니요	예	아니요	예	아니요
회사	v3	예	예	예	예	예
Contacts	v3	예	예	예	예	예
연락처 목록	v1	아니요	예	아니요	예	아니요
거래	v3	예	예	예	예	예
CRM 파이프라인(거래 파이프라인)	v1	아니요	아니요	아니요	예	아니요

개체	API 버전	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
이메일 이벤트	v1	아니요	예	아니요	예	아니요
호출	v3	예	예	예	예	예
Notes	v3	예	예	예	예	예
이메일	v3	예	예	예	예	예
회의	v3	예	예	예	예	예
업무	v3	예	예	예	예	예
우편	v3	예	예	예	예	예
사용자 지정 객체	v3	예	예	예	예	예
양식	v2	아니요	아니요	아니요	예	아니요
소유자	v3	아니요	예	아니요	예	아니요
Products	v3	예	예	예	예	예
티켓	v3	예	예	예	예	예
워크플로	v3	아니요	아니요	아니요	예	아니요
Associations	v4	예	아니요	아니요	예	아니요
연결 레이블	v4	아니요	아니요	아니요	예	아니요

예시:

```
hubspot_read = glueContext.create_dynamic_frame.from_options(
```

```
    connection_type="hubspot",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "contact",
        "API_VERSION": "v3"
    }
```

비동기 소스에 대해 지원되는 엔터티:

개체	API 버전	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Companies	v3	예	아니요	예	예	아니요
Contacts	v3	예	아니요	예	예	아니요
거래	v3	예	아니요	예	예	아니요
호출	v3	예	아니요	예	예	아니요
Notes	v3	예	아니요	예	예	아니요
Emails	v3	예	아니요	예	예	아니요
회의	v3	예	아니요	예	예	아니요
업무	v3	예	아니요	예	예	아니요
우편	v3	예	아니요	예	예	아니요
사용자 지정 객체	v3	예	아니요	예	예	아니요
Products	v3	예	아니요	예	예	아니요
Tickets	v3	예	아니요	예	예	아니요

예시:

```
hubspot_read = glueContext.create_dynamic_frame.from_options(
    connection_type="hubspot",
```

```
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "contact",
        "API_VERSION": "v3",
        "TRANSFER_MODE": "ASYNC"
    }
```

HubSpot 엔터티 및 필드 세부 정보:

HubSpot API v4:

개체	API 버전	필드	데이터 유형	지원되는 연산자
연결 레이블	v4	category	String	해당 사항 없음
		typeId	Integer	N/A
		레이블	String	해당 사항 없음
Associations		from	Struct	N/A
		id	String	"="
		아래로 변경합니다.	나열	N/A

> **Note**
>
> `Associations` 객체의 경우 두 객체 사이의 연결을 가져오려면 AWS Glue 작업을 생성하는 동안 필수 필터를 통해 'from Id'(첫 번째 객체의 ID)를 제공해야 합니다. 이 경우 여러 ID에 대한 연결을 가져오려면 `where` 절에서 여러 ID를 제공해야 합니다. 예: 연락처 ID '1' 및 '151'에 대한 `Associations`를 가져오는 경우 필터를 `where id=1 AND id=151`로 제공해야 합니다.

HubSpot API v3:

개체	필드	데이터 유형	지원되는 연산자
소유자	firstName	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	lastName	String	해당 사항 없음
	createdAt	DateTime	N/A
	archived	불	N/A
	teams	나열	N/A
	id	String	해당 사항 없음
	userId	Integer	N/A
	이메일	String	해당 사항 없음
	updatedAt	DateTime	N/A
워크플로	name	String	해당 사항 없음
	id	Integer	N/A
	type	String	해당 사항 없음
	활성화	불	N/A
	insertedAt	Long	N/A
	updatedAt	Long	N/A
	contactListIds	Struct	N/A
	personaTagIds	나열	N/A

다음 엔터티에 대해 HubSpot에서는 메타데이터를 동적으로 가져도록 엔드포인트를 제공하므로 운영자 지원이 각 엔터티의 데이터 유형 수준에서 캡처됩니다.

> **Note**
>
> DML_STATUS는 동기화 모드에서 런타임 시 모든 레코드에 추가된 가상 필드로, 해당 상태를 결정합니다(CREATED/UPDATED). CONTAINS/LIKE 연산자는 비동기 모드에서 지원되지 않습니다.

개체	데이터 유형	지원되는 연산자
Contact	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
Company	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A

개체	데이터 유형	지원되는 연산자
거래	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
티켓	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
Product	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A

개체	데이터 유형	지원되는 연산자
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
사용자 지정 객체	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
Call	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A

개체	데이터 유형	지원되는 연산자
이메일	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
회의	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
Note	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A

개체	데이터 유형	지원되는 연산자
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
Task	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A
우편	Integer	"=, !=, <, >, >=, <="
	Long	"=, !=, <, >, >=, <="
	String	"=, !=, LIKE"
	날짜	N/A
	DateTime	"between"
	불	"="
	나열	N/A
	Struct	N/A

HubSpot API v2:

개체	필드	데이터 유형	지원되는 연산자
양식	portalId	Integer	N/A
	guid	String	해당 사항 없음
	name	String	해당 사항 없음
	method	String	해당 사항 없음
	cssClass	String	해당 사항 없음
	리디렉션	String	해당 사항 없음
	submitText	String	해당 사항 없음
	notifyRecipients	String	해당 사항 없음
	createdAt	Long	N/A
	updatedAt	Long	N/A
	ignoreCurrentValues	불	N/A
	deletable	불	N/A
	inlineMessage	불	N/A
	captchaEnabled	불	N/A
	cloneable	불	N/A
	formFieldGroups	나열	N/A
	editable	불	N/A
	deletedAt	Integer	N/A
	themeName	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	parentId	Integer	N/A
	style	String	해당 사항 없음
	isPublished	불	N/A
	publishAt	Integer	N/A
	unpublishAt	Integer	N/A
	publishedAt	Integer	N/A
	kickbackEmailWorkflowId	String	해당 사항 없음
	kickbackEmailsJson	Integer	N/A
	customUid	String	해당 사항 없음
	createMarketableContact	불	N/A
	editVersion	Integer	N/A
	thankYouMessageJson	String	해당 사항 없음
	themeColor	String	해당 사항 없음
	alwaysCreateNewCompany	불	N/A
	internalUpdatedAt	Long	N/A
	businessUnitId	Integer	N/A
	portableKey	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	paymentSessionTemplateIds	나열	N/A
	selectedExternalOptions	나열	N/A

HubSpot API v1:

개체	필드	데이터 유형	지원되는 연산자
캠페인	id	Integer	N/A
	appId	Integer	N/A
	appName	String	해당 사항 없음
	lastUpdatedTime	Long	N/A
Contact_List	dynamic	불	N/A
	name	String	해당 사항 없음
	portalId	Integer	N/A
	createdAt	Long	N/A
	listId	Integer	N/A
	updatedAt	Long	N/A
	ListType	String	해당 사항 없음
	filters	나열	N/A
	authorId	Integer	N/A
	metaData	Struct	N/A

개체	필드	데이터 유형	지원되는 연산자
	archived	불	N/A
	ilsFilterBranch	String	해당 사항 없음
	filterIds	나열	N/A
	limitExempt	불	N/A
	internal	불	N/A
	readOnly	불	N/A
	parentId	Integer	N/A
Email_Event	id	String	해당 사항 없음
	type	String	해당 사항 없음
	수신자	String	해당 사항 없음
	portalId	Integer	N/A
	appId	Integer	N/A
	appName	String	해당 사항 없음
	emailCampaignId	Long	N/A
	attempt	Integer	N/A
	created	Long	N/A
	sentBy	Struct	N/A
	smtpId	String	해당 사항 없음
	응답	String	해당 사항 없음
	subject	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	cc	나열	N/A
	bcc	나열	N/A
	replyTo	나열	N/A
	from	String	해당 사항 없음
	dropReason	String	해당 사항 없음
	dropMessage	String	해당 사항 없음
	브라우저	Struct	N/A
	userAgent	String	해당 사항 없음
	기간	Long	N/A
	location	Struct	N/A
	filteredEvent	불	N/A
	deviceType	String	해당 사항 없음
	suppressedReason	String	해당 사항 없음
	suppressedMessage	String	해당 사항 없음
CRM_Pipeline	pipelineId	String	해당 사항 없음
	createdAt	Long	N/A
	updatedAt	Long	N/A
	objectType	String	해당 사항 없음
	레이블	String	해당 사항 없음
	displayOrder	Integer	N/A

개체	필드	데이터 유형	지원되는 연산자
	활성화	불	N/A
	스테이지	나열	N/A
	objectTypeId	String	해당 사항 없음
	기본값	불	N/A

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  DateTime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  “2024-01-01T10:00:00.115Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

다음 표에서는 엔터티 분할 필드 지원 세부 정보를 설명합니다.

엔터티 이름	분할 필드	데이터 유형
contact	hs_object_id	Long
	createdate, lastmodifieddate	DateTime
company	hs_object_id	Long

엔터티 이름	분할 필드	데이터 유형
	createdate, hs_lastmodifieddate	DateTime
deal	hs_object_id	Long
	createdate, hs_createdate, hs_lastmodifieddate	DateTime
ticket	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime
product	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime
custom_object	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime
call	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime
이메일	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime
회의	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime
note	hs_object_id	Long

엔터티 이름	분할 필드	데이터 유형
	createdate, hs_lastmodifieddate	DateTime
task	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime
postal_mail	hs_object_id	Long
	createdate, hs_lastmodifieddate	DateTime

예시:

```
hubspot_read = glueContext.create_dynamic_frame.from_options(
    connection_type="hubspot",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "company",
        "API_VERSION": "v3",
        "PARTITION_FIELD": "hs_object_id"
        "LOWER_BOUND": "50"
        "UPPER_BOUND": "16726619290"
        "NUM_PARTITIONS": "10"
    }
```

HubSpot 연결 옵션

다음은 HubSpot의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. HubSpot에서의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 HubSpot Rest API 버전입니다. 예: v1,v2,v3,v4.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.

- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- TRANSFER_MODE(문자열) - 쿼리를 비동기 모드에서 실행해야 하는지 여부를 나타내는 데 사용됩니다.

HubSpot 커넥터의 제한 사항 및 참고 사항

다음은 HubSpot 커넥터의 제한 사항입니다.

- 검색 엔드포인트는 지정된 쿼리에 대해 총 10,000개의 결과로 제한됩니다. 레코드가 10,000개를 초과하는 파티션은 400 오류가 발생합니다.
- 커넥터에 대한 기타 중요한 제한 사항은 제한 사항에 설명되어 있습니다.
- HubSpot은 최대 3개의 필터링 문을 허용합니다.
- 현재 HubSpot은 표준 HubSpot 객체(예: 연락처, 회사, 거래 또는 티켓)와 사용자 지정 객체 간의 연결을 지원합니다.
  - 무료 계정의 경우: 각 객체 페어링(예: 연락처 및 회사) 간에 최대 10개의 연결 유형만 생성할 수 있습니다.
  - 슈퍼 관리자 계정의 경우: 각 객체 페어링 간에 최대 50개의 연결 유형만 생성할 수 있습니다.
  - 자세한 내용은 연결 v4 및 연결 레이블 생성과 사용을 참조하세요.
- 'Quote' 및 'Communications' 객체는 현재 커넥터에서 지원되지 않으므로 연결에 없습니다.
- 비동기의 경우 SaaS는 값을 오름차순으로만 정렬합니다.
- Ticket 엔터티의 경우 SaaS는 비동기 모드에서 hs_object_id 필드를 반환하지 않습니다.

Instagram Ads에 연결

Instagram은 브랜드, 유명 인사, 사고 방식의 리더, 친구, 가족 등과 연결할 수 있는 인기 있는 사진 공유 앱입니다. 이는 사진 공유 및 소셜 네트워킹 서비스입니다. 사용자는 사진이나 짧은 동영상을 찍어 팔

로워와 공유할 수 있습니다. Instagram Ads는 Instagram 사용자에게 제공하기 위해 비즈니스에서 비용을 지불할 수 있는 게시물입니다.

주제

- Instagram Ads에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Instagram Ads 구성
- Instagram Ads 연결 구성
- Instagram Ads 엔터티에서 읽기
- Instagram Ads 연결 옵션
- Instagram Ads 커넥터에 대한 제한 사항 및 참고 사항

Instagram Ads에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Instagram Ads를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Instagram Ads에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Instagram Ads API 버전

다음 Instagram Ads API 버전이 지원됩니다.

- v17.0
- v18.0
- v19.0
- v20.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Instagram Ads 구성

AWS Glue를 사용하여 Instagram Ads에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Instagram Standard 계정에는 Facebook을 통해 간접적으로 액세스됩니다.
- 액세스 토큰을 생성하려면 사용자 인증이 필요합니다.

- Instagram Ads SDK 커넥터는 사용자 액세스 토큰 OAuth 흐름을 구현합니다.
- OAuth2.0을 사용하여 Instagram Ads에 대한 API 요청을 인증합니다. 이 웹 기반 인증은 2FA의 상위 세트인 다중 인증(MFA) 아키텍처에 속합니다.
- 사용자에게 엔드포인트에 액세스할 수 있는 권한을 부여해야 합니다. 사용자의 데이터에 액세스하기 위해 엔드포인트 권한 부여는 권한 및 기능을 통해 처리됩니다.

OAuth 2.0 자격 증명 가져오기

인스턴스에 대해 인증된 직접 호출을 수행할 수 있도록 API 자격 증명을 확보하려면 그래프 API를 참조하세요.

Instagram Ads 연결 구성

Instagram Ads에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 3각 OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.
- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Instagram Ads에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Instagram Ads로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 액세스 토큰을 생성합니다. 만료되는 시스템 사용자 토큰은 생성 날짜 또는 새로 고친 날짜로부터 60일 동안 유효합니다. 연속성을 생성하려면 개발자가 60일 이내에 액세스 토큰을 새로 고쳐야 합니다. 그렇지 않으면 액세스 토큰이 몰수되고 개발자가 API 액세스 권한을 다시 획득하기 위해 새 토큰을 확보해야 합니다. 액세스 토큰 새로 고침을 참조하세요.

Instagram Ads 연결을 구성하는 방법:

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Instagram Ads를 선택하세요.
   b. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

c. 사용자 관리형 클라이언트 애플리케이션 클라이언트 ID를 제공합니다.

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다. 선택한 보안 암호에는 값이 연결된 앱의 클라이언트 보안 암호인 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET` 키가 있어야 합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Instagram Ads 엔터티에서 읽기

사전 조건

읽으려는 Instagram Ads 객체. 객체 이름이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
캠페인	예	예	아니요	예	예
광고 세트	예	예	아니요	예	예
광고	예	예	아니요	예	예
광고 크리에이티브	아니요	예	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
인사이트 - 계정	아니요	예	아니요	예	아니요
광고 이미지	예	예	아니요	예	아니요
인사이트 - 광고	예	예	아니요	예	예
인사이트 - 광고 세트	예	예	아니요	예	예
인사이트 - 캠페인	예	예	아니요	예	예

예제:

```
instagramAds_read = glueContext.create_dynamic_frame.from_options(
    connection_type="instagramads",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v20.0"
    }
```

Instagram Ads 엔터티 및 필드 세부 정보

엔터티 및 필드 세부 정보에 대한 자세한 내용은 다음을 참조하세요.

- 캠페인
- 광고 세트
- 광고
- 광고 크리에이티브
- 광고 계정 인사이트
- 광고 이미지
- 광고 인사이트

- 광고 세트 인사이트
- 캠페인 인사이트

자세한 내용은 마케팅 API를 참조하세요.

> **Note**
>
> 구조체 및 목록 데이터 유형은 커넥터의 응답에서 문자열 데이터 유형으로 변환됩니다.

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리를 파티셔닝하는 데 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  DateTime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  "2022-01-01T00:00:00.000Z"
  ```
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).

  유효한 값의 예제:

  ```
  "2024-01-02T00:00:00.000Z"
  ```
- NUM_PARTITIONS: 파티션 수.

예제:

```
instagramADs_read = glueContext.create_dynamic_frame.from_options(
    connection_type="instagramads",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
```

```
        "API_VERSION": "v20.0",
        "PARTITION_FIELD": "created_time"
        "LOWER_BOUND": "2022-01-01T00:00:00.000Z"
        "UPPER_BOUND": "2024-01-02T00:00:00.000Z"
        "NUM_PARTITIONS": "10"
    }
```

Instagram Ads 연결 옵션

다음은 Instagram Ads에 대한 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Instagram Ads에서의 객체 이름.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용할 Instagram Ads 그래프 API 버전. 예: v21.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- `PARTITION_FIELD`(문자열) - 읽기에 사용됩니다. 쿼리를 파티셔닝하는 데 사용할 필드.
- `LOWER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- `UPPER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- `NUM_PARTITIONS`(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Instagram Ads 커넥터에 대한 제한 사항 및 참고 사항

다음은 Instagram Ads 커넥터에 대한 제한 사항 또는 참고 사항입니다.

- 앱의 직접 호출 수는 롤링 1시간 기간에 사용자가 수행할 수 있는 직접 호출 수(200에 사용자 수를 곱한 값). 요금 제한에 대한 자세한 내용은 사용 제한 및 비즈니스 사용 사례 사용 제한을 참조하세요.

AWS Glue Studio에서 Intercom에 연결

Intercom은 제품, 현재, 고객 측면에서 비즈니스와 고객 간의 열린 채널을 제공하는 참여 OS로, 고객 여정 전반에 걸쳐 모든 참여를 최대한 활용할 수 있는 지속적인 대화를 생성합니다.

주제

- Intercom에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Intercom 구성
- Intercom 연결 구성
- Intercom 엔터티에서 읽기
- Intercom 연결 옵션
- 제한 사항
- 새 Intercom 계정 생성 및 클라이언트 앱 구성

Intercom에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Intercom을 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Intercom에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Intercom API 버전

v2.5. 버전별 엔터티 지원은 Intercom 엔터티에서 읽기 섹션을 참조하세요.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, Amazon CloudWatch Logs, Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Intercom 구성

AWS Glue를 사용하여 Intercom에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- Intercom 계정이 있습니다. 자세한 내용은 새 Intercom 계정 생성 및 클라이언트 앱 구성 단원을 참조하십시오.
- Intercom 계정이 API 액세스에 대해 활성화되어 있습니다.
- 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하기 위해 사용하는 클라이언트 자격 증명을 제공하는 앱이 Intercom 개발자 계정 아래에 있어야 합니다. 자세한 내용은 새 Intercom 계정 생성 및 클라이언트 앱 생성 단계를 참조하세요.

이러한 요구 사항을 충족하면 Intercom 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Intercom 연결 구성

Intercom에서는 OAuth 2에 대한 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS

Glue 콘솔은 사용자를 Google Ads로 리디렉션합니다. 사용자가 로그인하고 Intercom 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때 고유한 클라이언트 ID 및 클라이언트 보안 암호를 제공해야 합니다. 이 시나리오에서는 여전히 Intercom으로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 자세한 내용은 Ads API를 참조하세요.

Intercom 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. 고객 관리형 연결 앱의 경우 - 보안 암호에는 연결된 앱 액세스 토큰, 새로 고침 토큰, client_id 및 client_secret이 포함되어야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Intercom을 선택합니다.
   b. Intercom 환경을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Intercom 엔터티에서 읽기

사전 조건

- 읽으려는 Intercom 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

개체	API_Version	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Admins	v2.5	No	아니요	아니요	예	No
회사	v2.5	No	예	아니요	예	No
대화	v2.5	예	예	예	예	예
데이터 속성	v2.5	No	아니요	아니요	예	No
연락처	v2.5	예	예	예	예	예
세그먼트	v2.5	No	아니요	아니요	예	No
Tags	v2.5	No	아니요	아니요	예	No
팀	v2.5	No	아니요	아니요	예	No

예

```
Intercom_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Intercom",
    connection_options={
```

```
        "connectionName": "connectionName",
        "ENTITY_NAME": "company",
        "API_VERSION": "V2.5"
    }
)
```

Intercom 엔터티 및 필드 세부 정보

개체	필드	데이터 형식	지원되는 연산자
Admins	type	String	NA
Admins	id	String	NA
Admins	avatar	구조체	NA
Admins	name	String	NA
Admins	이메일	String	NA
Admins	away_mode_enabled	불	NA
Admins	away_mode_reassign	불	NA
Admins	has_inbox_seat	불	NA
Admins	teams_ids	나열	NA
Admins	job_title	String	NA
회사	type	String	NA
회사	id	String	NA
회사	app_id	String	NA
회사	created_at	DateTime	NA
회사	remote_created_at	DateTime	NA
회사	updated_at	DateTime	NA

개체	필드	데이터 형식	지원되는 연산자
회사	last_request_at	DateTime	NA
회사	계획	구조체	NA
회사	company_id	String	NA
회사	name	String	NA
회사	custom_attributes	구조체	NA
회사	session_count	Integer	NA
회사	monthly_spend	Integer	NA
회사	user_count	Integer	NA
회사	industry	String	NA
회사	size	Integer	NA
회사	웹 사이트	String	NA
회사	tags	구조체	NA
회사	segments	구조체	NA
연락처	id	String	EQUAL_TO. NOT_EQUAL_TO
연락처	type	String	NA
연락처	workspace_id	String	NA
연락처	external_id	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	역할	String	EQUAL_TO. NOT_EQUAL_TO

개체	필드	데이터 형식	지원되는 연산자
연락처	이메일	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	phone	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	name	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	avatar	String	NA
연락처	owner_id	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	social_profiles	구조체	NA
연락처	has_hard_bounced	불	EQUAL_TO
연락처	marked_email_as_spam	불	EQUAL_TO
연락처	unsubscribed_from_emails	불	EQUAL_TO
연락처	created_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	updated_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN

개체	필드	데이터 형식	지원되는 연산자
연락처	signed_up_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	last_seen_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	last_replied_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	last_contacted_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	last_email_opened_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	last_email_clicked_at	DateTime	EQUAL_TO, GREATER_THAN, LESS_THAN
연락처	language_override	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	브라우저	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	browser_version	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO

개체	필드	데이터 형식	지원되는 연산자
연락처	browser_language	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	os	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	location	구조체	NA
연락처	location_country	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	location_region	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	location_city	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	android_app_name	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	android_app_version	String	NA
연락처	android_device	String	NA
연락처	android_os_version	String	NA
연락처	android_sdk_version	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	android_last_seen_at	날짜	NA

개체	필드	데이터 형식	지원되는 연산자
연락처	ios_app_name	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	ios_app_version	String	NA
연락처	ios_device	String	NA
연락처	ios_os_version	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	ios_sdk_version	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
연락처	ios_last_seen_at	DateTime	NA
연락처	custom_attributes	구조체	NA
연락처	tags	구조체	NA
연락처	notes	구조체	NA
연락처	companies	구조체	NA
연락처	unsubscribed_from_sms	불	NA
연락처	sms_consent	불	NA
연락처	opted_out_subscription_types	구조체	NA
연락처	referrer	String	NA
연락처	utm_campaign	String	NA
연락처	utm_content	String	NA

개체	필드	데이터 형식	지원되는 연산자
연락처	utm_medium	String	NA
연락처	utm_source	String	NA
연락처	utm_term	String	NA
대화	type	String	NA
대화	id	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	created_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	updated_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	source	구조체	NA
대화	source_id	String	EQUAL_TO, NOT_EQUAL_TO
대화	source_type	String	EQUAL_TO, NOT_EQUAL_TO,
대화	source_delivered_as	String	EQUAL_TO, NOT_EQUAL_TO,
대화	source_subject	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO

개체	필드	데이터 형식	지원되는 연산자
대화	source_body	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	source_author_id	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	source_author_type	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	source_author_name	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	source_author_email	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	source_url	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	contacts	구조체	NA
대화	teammates	구조체	NA
대화	제목	String	NA
대화	admin_assignee_id	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN

개체	필드	데이터 형식	지원되는 연산자
대화	team_assignee_id	Integer	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	custom_attributes	구조체	NA
대화	open	불	EQUAL_TO
대화	state	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	읽기	불	EQUAL_TO
대화	waiting_since	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	snoozed_until	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	tags	구조체	NA
대화	첫 번째_연락처_회신	구조체	NA
대화	priority	String	EQUAL_TO, NOT_EQUAL_TO
대화	topics	구조체	NA
대화	sla_applied	구조체	NA
대화	conversation_rating	구조체	NA

개체	필드	데이터 형식	지원되는 연산자
대화	conversation_rating_requested_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	conversation_rating_replied_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	conversation_rating_score	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	conversation_rating_remark	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	conversation_rating_contact_id	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	conversation_rating_admin_id	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO
대화	통계	구조체	NA
대화	statistics_time_to_assignment	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN

개체	필드	데이터 형식	지원되는 연산자
대화	statistics_time_to_admin_reply	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_time_to_first_close	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_time_to_last_close	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_median_time_to_reply	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_first_contact_reply_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_first_assignment_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_first_admin_reply_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN

개체	필드	데이터 형식	지원되는 연산자
대화	statistics_first_close_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_last_assignment_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_last_assignment_admin_reply_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_last_contact_reply_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_last_admin_reply_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_last_close_at	DateTime	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_last_closed_by_id	String	CONTAINS, EQUAL_TO, NOT_EQUAL_TO

개체	필드	데이터 형식	지원되는 연산자
대화	statistics_count_reopens	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_count_assignments	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	statistics_count_conversation_parts	Integer	EQUAL_TO, NOT_EQUAL_TO, GREATER_THAN, LESS_THAN
대화	conversation_parts	나열	NA
데이터 속성	id	Integer	NA
데이터 속성	type	String	NA
데이터 속성	모델	String	NA
데이터 속성	name	String	NA
데이터 속성	full_name	String	NA
데이터 속성	레이블	String	NA
데이터 속성	설명	String	NA
데이터 속성	data_type	String	NA
데이터 속성	옵션	나열	NA
데이터 속성	api_writable	불	NA
데이터 속성	ui_writable	불	NA

개체	필드	데이터 형식	지원되는 연산자
데이터 속성	사용자 지정	불	NA
데이터 속성	archived	불	NA
데이터 속성	created_at	불	NA
데이터 속성	updated_at	DateTime	NA
데이터 속성	admin_id	String	NA
세그먼트	type	String	NA
세그먼트	id	String	NA
세그먼트	name	String	NA
세그먼트	created_at	DateTime	NA
세그먼트	updated_at	DateTime	NA
세그먼트	person_type	String	NA
세그먼트	count	Integer	NA
Tags	type	String	NA
Tags	id	String	NA
Tags	name	String	NA
팀	type	String	NA
팀	id	String	NA
팀	name	String	NA
팀	admin_ids	나열	NA

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-02-06".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준의 분할 필드 지원 세부 정보는 다음 표에 캡처되어 있습니다.

Entity Name	분할 필드	데이터 형식
연락처	created_at, updated_at,last_seen_at	DateTime
대화	id	Integer
대화	created_at, updated_at	DateTime

예제

```
Intercom_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Intercom",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "conversation",
        "API_VERSION": "V2.5",
        "PARTITION_FIELD": "created_at"
        "LOWER_BOUND": "2022-07-13T07:55:27.065Z"
        "UPPER_BOUND": "2022-08-12T07:55:27.065Z"
        "NUM_PARTITIONS": "2"
    }
)
```

Intercom 연결 옵션

다음은 Intercom에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Intercom에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Intercom Rest API 버전. 예: v2.5.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리를 파티셔닝하는 데 사용할 필드.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - 사용자가 작업을 실행하려는 인스턴스의 URL. 예: https://api.intercom.io.

제한 사항

다음은 Intercom 커넥터에 대한 제한 사항입니다.

- 회사 엔터티를 사용하는 경우 반환할 수 있는 회사 수는 10,000곳으로 제한됩니다. 자세한 내용은 List all companies API를 참조하세요.
- 정렬 기준을 적용하는 동안 필터는 연락 엔터티와 대화 엔터티 모두에 필수입니다.
- MCA는 SaaS 제공업체에서 지원합니다. 그러나 설명서에 언급된 API 사용 제한에 따라 AWS Glue에 MCA를 호스팅하지 않습니다. 이 경우 다른 워크로드에 영향을 미치고 리소스 경합으로 인해 잠재적으로 성능 문제가 발생할 수 있기 때문입니다.

새 Intercom 계정 생성 및 클라이언트 앱 구성

Intercom 계정 생성

1. Intercom URL을 선택하고 페이지 오른쪽 상단에서 Start my free trial을 선택하세요.
2. 페이지 오른쪽 상단에 있는 Try for free 버튼을 선택하세요.
3. 필요한 비즈니스 유형을 선택하세요.

4. 페이지에서 요청된 정보를 입력하세요.
5. 모든 정보를 입력한 후 Register를 선택하세요.

Intercom 개발자 앱 생성

클라이언트 ID 및 클라이언트 보안 암호를 가져오려면 개발자 계정을 생성합니다.

1. https://app.intercom.com/으로 이동하세요.
2. 이메일 ID 및 암호를 입력하거나 Sign In Using Google을 선택하고 로그인하세요.
3. 왼쪽 하단에서 사용자 프로파일을 선택하고 설정을 선택하세요.
4. Apps & Integration을 선택하세요.
5. Apps & Integration 아래 Developer Hub 탭을 선택하세요.
6. New app을 선택하고 여기에서 앱을 생성하세요.
7. 앱 이름을 입력하고 Create 앱을 선택하세요.
8. 앱 내에서 Authentication 섹션으로 이동하세요.
9. Edit을 선택하고 리디렉션 URI를 추가하세요. 리전별 리디렉션 URL을 `https://<aws-region>.console.aws.amazon.com/gluestudio/oauth`로 추가하세요. 예를 들어 `https://us-east-1.console.aws.amazon.com/gluestudio/oauth for the us-east-1 region`을 추가하세요.
10. Basic Information 섹션에서 생성된 클라이언트 ID 및 클라이언트 보안 암호를 가져오세요.

Jira Cloud에 연결

Jira Cloud는 Atlassian에서 개발한 플랫폼입니다. 플랫폼에는 팀이 Agile 프로젝트를 계획하고 추적하는 데 도움이 되는 문제 추적 제품이 포함되어 있습니다. Jira Cloud 사용자는 계정에 문제, 워크플로 및 이벤트와 같은 프로젝트에 대한 데이터가 포함됩니다. AWS Glue 를 사용하여 Jira Cloud 데이터를 특정 AWS 서비스 또는 지원되는 다른 애플리케이션으로 전송할 수 있습니다.

주제

- AWS Glue Jira Cloud에 대한 지원
- 연결 생성 및 사용 API 작업이 포함된 정책
- Jira Cloud 구성
- Jira Cloud 연결 구성

- Jira Cloud 엔터티에서 읽기
- Jira Cloud 연결 옵션
- Jira Cloud 커넥터에 대한 제한 사항 및 참고 사항

AWS Glue Jira Cloud에 대한 지원

AWS Glue 는 다음과 같이 Jira Cloud를 지원합니다.

소스로 지원되나요?

예. 작업을 사용하여 AWS Glue ETL Jira Cloud에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Jira Cloud API 버전

지원되는 Jira Cloud API 버전은 다음과 같습니다.

- v2

버전별 엔터티 지원은 지원되는 소스 엔터티를 참조하세요.

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려면 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS 관리 콘솔을 사용할 때 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

Jira Cloud 구성

AWS Glue 를 사용하여 Jira Cloud에서 지원되는 대상으로 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Jira Cloud에서 Jira 소프트웨어 제품을 사용하는 Atlassian 계정이 있습니다. 자세한 내용은 Jira Cloud 계정 생성 단원을 참조하십시오.
- 에 대한 서비스 액세스 권한이 있는 AWS 계정이 생성되어 있어야 합니다 AWS Glue.
- 이 앱은 가 계정에 인증된 호출을 할 때 데이터에 안전하게 액세스하는 데 AWS Glue 사용하는 클라이언트 자격 증명을 제공합니다. 자세한 내용은 Atlassian 개발자 설명서의 OAuth 2.0(3LO) 활성화를 참조하세요.

이러한 요구 사항을 충족하면 Jira Cloud 계정에 AWS Glue 연결할 준비가 된 것입니다.

Jira Cloud 계정 생성

Jira Cloud 계정을 생성하려면:

1. Atlassian 가입 URL으로 이동합니다.
2. 작업 이메일과 이름을 입력하고 동의를 선택합니다. 확인 이메일을 받습니다.
3. 이메일을 확인한 후 암호를 생성하고 가입을 선택할 수 있습니다.
4. 이름과 암호를 입력하고 가입을 선택합니다.
5. 사이트에 들어가야 하는 페이지로 리디렉션됩니다. 사이트 이름을 입력하고 동의를 선택합니다.

Atlassian Cloud 사이트가 시작되면 프로젝트 유형 기본 설정에 따라 몇 가지 질문에 답하여 Jira를 설정할 수 있습니다.

기존 계정에 로그인하려면:

1. Atlassian 로그인URL으로 이동하여 자격 증명을 입력합니다.
2. 이메일과 암호를 입력하고 로그인을 클릭합니다. Jira 대시보드로 리디렉션됩니다.

Jira Cloud에서 앱 생성

Jira Cloud에서 앱을 생성하고 관리형 클라이언트 앱에서 클라이언트 ID 및 클라이언트 보안 암호를 가져오려면:

1. Jira CloudURL로 이동하여 보안 인증을 입력합니다.
2. 생성을 선택하고 OAuth 2.0 통합 옵션을 선택합니다.
3. 앱 이름을 입력하고 T&C를 확인한 다음 생성을 선택합니다.
4. 왼쪽 메뉴의 배포 섹션으로 이동하여 편집을 선택합니다.
5. 배포 제어 편집 섹션에서 다음을 수행합니다.
   a. 공유로 DISTRIBUTIONSTATUS를 선택합니다.
   b. 공급업체 이름을 입력합니다.
   c. 개인 정보 보호 정책 에 URL 대한 를 입력합니다. 예: https://docs.aws.amazon.com/glue/latest/dg/security-iam-awsmanpol.html
   d. 서비스 약관의 URL를 입력합니다(선택 사항).
   e. 고객 지원 담당자의 URL를 입력합니다(선택 사항).
   f. 에서 예/아니요를 PERSONAL DATA DECLARATION 선택하고 변경 사항 저장을 선택합니다.
6. 각 앱의 왼쪽 메뉴에서 권한으로 이동합니다.
7. Jira API에서 추가를 선택합니다. 추가되면 구성 옵션을 선택합니다.

8. 클래식 범위 > Jira 플랫폼 REST API 섹션에서 범위 편집을 선택하고 모든 범위를 확인합니다. 저장을 클릭합니다.
9. 세분화된 범위 아래에서 범위 편집을 선택하고 다음 범위를 선택합니다.
10. 아래로 스크롤하여 범위를 찾습니다. “” 및 “표준CRM”이라는 제목에서 두 가지 종류의 범위를 선택해야 합니다.
11. 다음 범위를 추가합니다.

```
read:application-role:jira
read:audit-log:jira
read:avatar:jira
read:field:jira
read:group:jira
read:instance-configuration:jira
read:issue-details:jira
read:issue-event:jira
read:issue-link-type:jira
read:issue-meta:jira
read:issue-security-level:jira
read:issue-security-scheme:jira
read:issue-type-scheme:jira
read:issue-type-screen-scheme:jira
read:issue-type:jira
read:issue.time-tracking:jira
read:label:jira
read:notification-scheme:jira
read:permission:jira
read:priority:jira
read:project:jira
read:project-category:jira
read:project-role:jira
read:project-type:jira
read:project-version:jira
read:project.component:jir
read:project.property:jira
read:resolution:jira
read:screen:jira
read:status:jira
read:user:jira
read:workflow-scheme:jira
read:workflow:jira
read:field-configuration:jira
read:issue-type-hierarchy:jira
```

```
read:webhook:jira
```

12.왼쪽 메뉴의 인증으로 이동하여 추가를 선택합니다.

13https://us-east-1.console.aws.amazon.com/gluestudio/oauth와 같은 콜백 URL 입력

14.왼쪽 메뉴의 설정으로 이동하여 아래로 스크롤하여 인증 세부 정보를 확인합니다. 클라이언트 ID와 보안 암호를 기록해 둡니다.

Jira Cloud 연결 구성

Jira Cloud에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Jira Cloud로 리디렉션합니다. 사용자가 로그인하고 Jira Cloud 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Jira Cloud에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Jira Cloud로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Jira Cloud 설명서는 Enabling OAuth 2.0(3LO)을 참조하세요.

Jira Cloud 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Jira Cloud를 선택합니다.
   b. Jira Cloud 환경을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Jira Cloud 엔터티에서 읽기

사전 조건

읽으려는 Jira Cloud 객체. 객체 이름(예: 감사 레코드 또는 문제)이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
감사 레코드	예	예	아니요	예	예
문제	예	예	아니요	예	예
문제 필드	아니요	아니요	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
문제 필드 구성	예	예	아니요	예	예
문제 링크 유형	아니요	아니요	아니요	예	아니요
문제 알림 체계	예	예	아니요	예	예
문제 보안 체계	아니요	아니요	아니요	예	아니요
문제 유형 체계	예	예	예	예	예
문제 유형 화면 체계	예	예	예	예	예
문제 유형	아니요	아니요	아니요	예	아니요
Jira 설정	예	아니요	아니요	예	아니요
Jira 설정 고급	아니요	아니요	아니요	예	아니요
Jira 설정 전역	아니요	아니요	아니요	예	아니요
레이블	아니요	아니요	아니요	예	예
사용자 본인	예	아니요	아니요	예	아니요
권한	아니요	아니요	아니요	예	아니요.
프로젝트	예	예	예	예	예
프로젝트 카테고리	아니요	아니요	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
프로젝트 유형	아니요	아니요	아니요	예	아니요
서버 정보	아니요	아니요	아니요	예	아니요
사용자	아니요	아니요	아니요.	예	아니요
워크플로	예	예	예	예	예
워크플로 체계	아니요	예	아니요	예	예
워크플로 체계 프로젝트 연결	예	아니요	아니요	예	아니요
워크플로 상태	아니요	아니요	아니요	예	아니요
워크플로 상태 카테고리	아니요	아니요	아니요	예	아니요

예제:

```
jiracloud_read = glueContext.create_dynamic_frame.from_options(
    connection_type="JiraCloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "audit-record",
        "API_VERSION": "v2"
    }
```

Jira Cloud 엔터티 및 필드 세부 정보:

객체	필드	데이터 유형	지원되는 필터 연산자
감사 레코드	필터	String	"="
	from	DateTime	"="
	아래로 변경합니다.	DateTime	"="
	id	Integer	N/A
	요약	String	해당 사항 없음
	remoteAddress	String	해당 사항 없음
	authorAccountId	String	해당 사항 없음
	생성 완료	String	해당 사항 없음
	category	String	해당 사항 없음
	eventSource	String	해당 사항 없음
	설명	String	해당 사항 없음
	objectItem	구조체	N/A
	changedValues	나열	N/A
	associatedItems	나열	N/A
그룹	groupName	나열	"="
	name	String	해당 사항 없음
	groupId	String	"="
문제	affectedVersion	String	"=, !="
	assignee	String	"=, !="
	category	String	"=, !="

객체	필드	데이터 유형	지원되는 필터 연산자
	component	String	"=, !="
	creator	String	"=, !="
	due	DateTime	N/A
	epic_link	String	"=, !="
	필터	String	"=, !="
	fixVersion	String	"=, !="
	hierarchyLevel	Integer	"=, !="
	issueKey	String	"=, !=, >, <, >=, <="
	issueLink	String	"=, !="
	issueLinkType	String	"=, !="
	labels	String	"=, !="
	lastViewed	DateTime	"=, >, <, >=, <=, between"
	level	String	"=, !="
	parent	String	"=, !="
	우선순위	String	"=, !="
	project	String	"=, !="
	projectType	String	"=, !="
	reporter	String	"=, !="
	resolution	String	"=, !="

객체	필드	데이터 유형	지원되는 필터 연산자
	resolved	DateTime	"=, >, <, >=, <=, between"
	Sprint	String	"=, !="
	status	String	"=, !="
	type	String	"=, !="
	updated	DateTime	"=, >, <, >=, <=, between"
	voter	String	"=, !="
	votes	Integer	"=, !=, <, >, <=, >=, between"
	watcher	String	"=, !="
	watchers	Integer	"=, !=, <, >, <=, >=, between"
	workRatio	Integer	"=, !=, <, >, <=, >=, between"
	validateQuery	String	"="
	expand	String	"="
	fieldByKeys	불	"="
	id	String	해당 사항 없음
	self	String	해당 사항 없음
	키	String	해당 사항 없음
	renderedFields	구조체	N/A

객체	필드	데이터 유형	지원되는 필터 연산자
	속성	나열	"="
	names	구조체	N/A
	스키마	구조체	N/A
	transitions	나열	N/A
	작업	구조체	N/A
	editmeta	구조체	N/A
	변경 로그	구조체	N/A
	versionedRepresentations	구조체	N/A
	필드	나열	"="
	fieldsToInclude	구조체	N/A
	warningMessages	나열	N/A
	생성 완료	DateTime	N/A
	worklogDate	DateTime	N/A
IssueEvents	id	Integer	N/A
	name	String	해당 사항 없음
문제 필드	id	String	해당 사항 없음
	키	String	해당 사항 없음
	name	String	해당 사항 없음
	사용자 지정	불	N/A
	orderable	불	N/A

객체	필드	데이터 유형	지원되는 필터 연산자
	navigable	불	N/A
	searchable	불	N/A
	clauseNames	나열	N/A
	scope	구조체	N/A
	스키마	구조체	N/A
문제 필드 구성	isDefault	불	"="
	쿼리	String	"="
	id	Integer	"="
	name	String	해당 사항 없음
	설명	String	해당 사항 없음
문제 링크 유형	id	String	해당 사항 없음
	name	String	해당 사항 없음
	inward	String	해당 사항 없음
	outward	String	해당 사항 없음
	self	String	해당 사항 없음
문제 알림 체계	expand	String	"="
	self	String	해당 사항 없음
	id	Integer	N/A
	name	String	해당 사항 없음
	설명	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
	notificationSchemeEvents	나열	N/A
	scope	구조체	N/A
문제 우선순위	self	String	해당 사항 없음
	statusColor	String	해당 사항 없음
	설명	String	해당 사항 없음
	iconUrl	String	해당 사항 없음
	name	String	해당 사항 없음
	id	String	해당 사항 없음
	isDefault	불	N/A
문제 해결	self	String	해당 사항 없음
	id	String	해당 사항 없음
	설명	String	해당 사항 없음
	name	String	해당 사항 없음
문제 보안 체계	self	String	해당 사항 없음
	id	Integer	N/A
	name	String	해당 사항 없음
	설명	String	해당 사항 없음
	defaultSecurityLevelId	Integer	N/A
	levels	나열	N/A
문제 유형	self	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
	id	String	해당 사항 없음
	설명	String	해당 사항 없음
	iconUrl	String	해당 사항 없음
	name	String	해당 사항 없음
	subtask	불	N/A
	avatarId	Integer	N/A
	entityId	String	해당 사항 없음
	hierarchyLevel	Integer	N/A
	scope	구조체	N/A
문제 유형 체계	orderBy	String	"="
	expand	String	"="
	queryString	String	"="
	id	String	해당 사항 없음
	name	String	해당 사항 없음
	설명	String	해당 사항 없음
	defaultIssueTypeId	String	해당 사항 없음
	isDefault	불	N/A
문제 유형 화면 체계	queryString	String	"="
	orderBy	String	"="
	expand	String	"="

객체	필드	데이터 유형	지원되는 필터 연산자
	id	String	"="
	name	String	해당 사항 없음
	설명	String	해당 사항 없음
Jira 설정	키	String	해당 사항 없음
	keyFilter	String	"="
	id	String	해당 사항 없음
	값	String	해당 사항 없음
	name	String	해당 사항 없음
	desc	String	해당 사항 없음
	type	String	해당 사항 없음
	defaultValue	String	해당 사항 없음
	example	String	해당 사항 없음
	allowedValues	나열	N/A
Jira 설정 고급	id	String	해당 사항 없음
	키	String	해당 사항 없음
	값	String	해당 사항 없음
	name	String	해당 사항 없음
	desc	String	해당 사항 없음
	type	String	해당 사항 없음
	defaultValue	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
	example	String	해당 사항 없음
	allowedValues	나열	N/A
Jira 설정 전역	votingEnabled	불	N/A
	watchingEnabled	불	N/A
	unassignedIssuesAllowed	불	N/A
	subTasksEnabled	불	N/A
	issueLinkingEnabled	불	N/A
	timeTrackingEnabled	불	N/A
	attachmentsEnabled	불	N/A
	timeTrackingConfiguration	구조체	N/A
레이블	values	나열	N/A
사용자 본인	expand	String	"="
	self	String	해당 사항 없음
	accountId	String	해당 사항 없음
	accountType	String	해당 사항 없음
	emailAddress	String	해당 사항 없음
	avatarUrls	String	해당 사항 없음
	displayName	String	해당 사항 없음
	활성화	불	N/A

객체	필드	데이터 유형	지원되는 필터 연산자
	timeZone	String	해당 사항 없음
	locale	String	해당 사항 없음
	그룹	구조체	N/A
	applicationRoles	구조체	N/A
권한	id	String	해당 사항 없음
	키	String	해당 사항 없음
	name	String	해당 사항 없음
	type	String	해당 사항 없음
	설명	String	해당 사항 없음
	havePermission	불	N/A
	deprecatedKey	불	N/A
프로젝트	orderBy	String	"="
	키	나열	"="
	쿼리	String	"="
	typeKey	String	"="
	categoryId	Integer	"="
	작업	String	"="
	expand	String	"="
	status	나열	"="
	self	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
	id	Integer	"="
	키	String	해당 사항 없음
	설명	String	해당 사항 없음
	lead	구조체	N/A
	구성 요소	나열	N/A
	issueTypes	나열	N/A
	url	String	해당 사항 없음
	이메일	String	해당 사항 없음
	assigneeType	String	해당 사항 없음
	versions	나열	N/A
	name	String	해당 사항 없음
	역할	구조체	N/A
	avatarUrls	구조체	N/A
	projectCategory	구조체	N/A
	projectTypeKey	String	해당 사항 없음
	simplified	불	N/A
	style	String	해당 사항 없음
	favourite	불	N/A
	isPrivate	불	N/A
	issueTypeHierarchy	구조체	N/A

객체	필드	데이터 유형	지원되는 필터 연산자
	권한	구조체	N/A
	속성	나열	"="
	uuid	String	해당 사항 없음
	insight	구조체	N/A
	삭제됨	불	N/A
	retentionTillDate	String	해당 사항 없음
	deletedDate	String	해당 사항 없음
	deletedBy	구조체	N/A
	archived	불	N/A
	archivedDate	String	해당 사항 없음
	archivedBy	구조체	N/A
	landedPageInfo	구조체	N/A
프로젝트 카테고리	self	String	해당 사항 없음
	id	String	해당 사항 없음
	name	String	해당 사항 없음
	설명	String	해당 사항 없음
프로젝트 유형	키	String	해당 사항 없음
	formattedKey	String	해당 사항 없음
	설명	String	해당 사항 없음
	descriptionI18nKey	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
	icon	String	해당 사항 없음
	color	String	해당 사항 없음
서버 정보	baseUrl	String	해당 사항 없음
	version	String	해당 사항 없음
	versionNumbers	나열	N/A
	deploymentType	String	해당 사항 없음
	buildNumber	Integer	N/A
	buildDate	DateTime	N/A
	serverTime	DateTime	N/A
	scmInfo	String	해당 사항 없음
	serverTitle	String	해당 사항 없음
	healthChecks	나열	N/A
사용자	self	String	해당 사항 없음
	accountId	String	해당 사항 없음
	accountType	String	해당 사항 없음
	emailAddress	String	해당 사항 없음
	avatarUrls	구조체	N/A
	displayName	String	해당 사항 없음
	활성화	불	N/A
	timeZone	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
	locale	String	해당 사항 없음
	그룹	구조체	N/A
	applicationRoles	구조체	N/A
	expand	String	해당 사항 없음
워크플로	workflowName	String	"="
	expand	String	"="
	queryString	String	"="
	orderBy	String	"="
	isActive	불	"="
	id	구조체	N/A
	설명	String	해당 사항 없음
	transitions	나열	N/A
	statuses	나열	N/A
	isDefault	불	N/A
	schemes	나열	N/A
	projects	나열	N/A
	hasDraftWorkflow	불	N/A
	작업	구조체	N/A
	생성 완료	String	해당 사항 없음
	updated	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
워크플로 체계	self	String	해당 사항 없음
	id	Integer	N/A
	name	String	해당 사항 없음
	설명	String	해당 사항 없음
	defaultWorkflow	String	해당 사항 없음
	issueTypeMappings	구조체	N/A
	originalDefaultWorkflow	String	해당 사항 없음
	originalIssueTypeMappings	구조체	N/A
	draft	불	N/A
	lastModifiedUser	구조체	N/A
	lastModified	String	해당 사항 없음
	updateDraftIfNeeded	불	N/A
	issueTypes	구조체	N/A
워크플로 체계 프로젝트 연결	projectId	Integer	"="
	projectIds	나열	N/A
	workflowScheme	구조체	N/A
워크플로 상태	self	String	해당 사항 없음
	설명	String	해당 사항 없음
	iconUrl	String	해당 사항 없음

객체	필드	데이터 유형	지원되는 필터 연산자
	name	String	해당 사항 없음
	id	String	해당 사항 없음
	StatusCategory	구조체	N/A
워크플로 상태 카테고리	self	String	해당 사항 없음
	id	String	해당 사항 없음
	키	String	해당 사항 없음
	colorName	String	해당 사항 없음
	name	String	해당 사항 없음

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(NUM_PARTITIONS)을 제공할 수 있습니다. 이 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- NUM_PARTITIONS: 파티션 수.

예제:

```
jiraCloud_read = glueContext.create_dynamic_frame.from_options(
    connection_type="JiraCloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "issue",
        "API_VERSION": "v2",
        "NUM_PARTITIONS": "10"
    }
```

Jira Cloud 연결 옵션

다음은 Jira Cloud의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Jira Cloud에서 객체의 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Jira Cloud Rest API 버전입니다. 예: v2.
- DOMAIN_URL(문자열) - (필수) 사용하려는 Jira Cloud ID입니다.
- SELECTED_FIELDS(목록<문자열>) - 기본값: 비어 있음(SELECT*). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Jira Cloud 커넥터에 대한 제한 사항 및 참고 사항

다음은 Jira Cloud 커넥터에 대한 제한 사항 또는 참고 사항입니다.

- Contains 연산자는 문자열 데이터 형식인 resourceName 필드를 사용하지 않습니다.

Kustomer에 연결

Kustomer는 사용하기 쉬운 단일 도구로 고객에게 더 나은 서비스를 제공하는 데 필요한 모든 것을 하나로 모으는 강력한 고객 경험 플랫폼입니다.

주제

- AWS Glue의 Kustomer 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Kustomer 구성
- Kustomer 연결 구성
- Kustomer 엔터티에서 읽기
- Kustomer 연결 옵션
- Kustomer 제한 사항

AWS Glue의 Kustomer 지원

AWS Glue는 다음과 같이 Kustomer를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Kustomer에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Kustomer API 버전

다음 Kustomer API 버전이 지원됩니다.

- v1

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
     "Statement": [
      {
             "Effect": "Allow",
             "Action": [
                 "glue:ListConnectionTypes",
                 "glue:DescribeConnectionType",
                 "glue:RefreshOAuth2Tokens",
                 "glue:ListEntities",
                 "glue:DescribeEntity"
             ],
             "Resource": "*"
         }
     ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절

차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Kustomer 구성

AWS Glue를 사용하여 Kustomer에서 지원되는 대상으로 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- 전송하려는 데이터가 포함된 Kustomer 계정이 있습니다.
- 계정 설정에서 API 키를 생성했습니다. 자세한 내용은 API 키 생성 단원을 참조하십시오.
- 연결을 생성하는 동안 AWS Glue에 API 키를 제공합니다.

이러한 요구 사항을 충족하면 Kustomer 계정에 AWS Glue를 연결할 준비가 된 것입니다.

API 키 생성

다음 단계를 따라 AWS Glue Studio에서 Kustomer 커넥터에 대한 연결을 생성하는 데 사용할 API 키를 생성합니다.

1. 자격 증명을 사용하여 Kustomer 대시보드에 로그인합니다.
2. 왼쪽 메뉴에서 Settings 아이콘을 선택합니다.
3. Security 드롭다운을 확장하고 API Keys를 선택합니다.
4. API Key 생성 페이지의 오른쪽 상단 모서리에서 Add an API Key를 선택합니다.
5. 생성 중인 API 키의 필수 사항을 입력합니다.
   - Name: API 키의 이름.
   - Roles: Kustomer API가 작동하려면 'org'를 선택해야 합니다.
   - Expires (in days): API 키가 유효한 기간(일). 사용 사례에 적합한 경우 Never expires로 유지할 수 있습니다.

6. 생성(Create)을 선택합니다.
7. API 키(토큰) 값을 저장하면 AWS Glue Studio에서 Kustomer 커넥터에 대한 연결을 생성할 때 사용할 수 있습니다.

Kustomer 연결 구성

다음 단계를 따라 Kustomer 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `apiKey`와 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Kustomer를 선택합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Kustomer 엔터티에서 읽기

사전 조건

읽으려는 Kustomer 객체. 객체 이름(예: Brands 또는 Cards)이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Brands	아니요	예	아니요	예	아니요
카드	아니요	예	아니요	예	아니요
Chat Settings	아니요	아니요	아니요	예	아니요
회사	예	예	예	예	예
Conversations	예	예	예	예	예
고객	예	예	예	예	예
Customer Searches Pinned	아니요	예	아니요	예	아니요
Customer Searches Position	아니요	아니요	아니요	예	아니요
Email Hooks	아니요	예	아니요	예	아니요
Web Hooks	아니요	예	아니요	예	아니요
KB Articles	아니요	예	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
KB Categories	아니요	예	아니요	예	아니요
KB Forms	아니요	예	아니요	예	아니요
KB Routes	아니요	예	아니요	예	아니요
KB Tags	아니요	예	아니요	예	아니요
KB Templates	아니요	예	아니요	예	아니요
KB Themes	아니요	예	아니요	예	아니요
Klasses	아니요	예	아니요	예	아니요
KViews	아니요	예	아니요	예	아니요
메시지	예	예	예	예	예
Notes	예	예	예	예	예
알림	아니요	예	아니요	예	아니요

예시:

```
Kustomer_read = glueContext.create_dynamic_frame.from_options(
    connection_type="kustomer",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "brands",
        "API_VERSION": "v1"
    }
```

Kustomer 엔터티 및 필드 세부 정보

엔터티 및 필드 세부 정보에 대한 자세한 내용은 다음을 참조하세요.

- Brands
- Cards
- Chat Settings
- Companies
- Conversations
- Customers
- Customers Searches Pinned
- Customer Searches Positions
- Hooks Email
- Hooks Web
- KB Articles
- KB Categories
- KB Forms
- KB Routes
- KB Tags
- KB Templates
- KB Themes
- Klasses
- Kviews
- 메시지
- 참고
- 알림

Kustomer API v1

개체	필드	데이터 유형	지원되는 연산자
Brands	id	String	해당 사항 없음
	name	String	해당 사항 없음
	iconUrl	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	기본값	불	N/A
카드	id	String	해당 사항 없음
	name	String	해당 사항 없음
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	설명	String	해당 사항 없음
	url	String	해당 사항 없음
	contexts	나열	N/A
Chat Settings	id	String	해당 사항 없음
	settingsVersion	Integer	N/A
	widgetType	String	해당 사항 없음
	version	Integer	N/A
	teamName	String	해당 사항 없음
	greeting	String	해당 사항 없음
	autoreply	String	해당 사항 없음
	embedIconUrl	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	embedIconColor	String	해당 사항 없음
	fallbackEmailSubject	String	해당 사항 없음
	fallbackEmailIntroduction	String	해당 사항 없음
	활성화	불	N/A
	outboundChatEnabled	불	N/A
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	offhoursMessage	String	해당 사항 없음
	offhoursImageUrl	String	해당 사항 없음
	closableChat	불	N/A
	noHistory	불	N/A
	disableAttachments	불	N/A
	volumeControl	Struct	N/A
	singleSessonChat	불	N/A
	showTypingIndicatorWeb	불	N/A
회사	id	String	해당 사항 없음
	name	String	=, !=, CONTAINS
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A

개체	필드	데이터 유형	지원되는 연산자
	modifiedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	tags	나열	N/A
	domains	나열	N/A
	emails	나열	N/A
	phones	나열	N/A
	whatsapps	나열	N/A
	socials	나열	N/A
	urls	나열	N/A
	위치	나열	N/A
	roleGroupVersions	나열	N/A
	rev	Integer	N/A
Conversations	id	String	해당 사항 없음
	name	String	=, !=, CONTAINS
	preview	String	해당 사항 없음
	channels	나열	N/A
	status	String	=, !=, CONTAINS
	messageCount	Integer	=, !=, >, >=, <, <=
	noteCount	Integer	=, !=, >, >=, <, <=
	satisfaction	Integer	=, !=, >, >=, <, <=
	satisfactionLevel	Struct	N/A

개체	필드	데이터 유형	지원되는 연산자
	createdAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	updatedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	modifiedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	lastActivityAt	DateTime	N/A
	스팸	불	N/A
	ended	불	=, !=
	endedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	endedReason	String	CONTAINS
	endedByType	String	해당 사항 없음
	importedAt	String	해당 사항 없음
	tags	나열	N/A
	suggestedTags	나열	N/A
	감성	String	해당 사항 없음
	예측	나열	N/A
	suggestedShortcuts	나열	N/A
	firstMessageIn	Struct	N/A
	firstMessageOut	Struct	N/A
	lastMessageIn	Struct	N/A

개체	필드	데이터 유형	지원되는 연산자
	lastMessageOut	Struct	N/A
	lastMessageAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	lastMessageUnrespondedTo	Struct	N/A
	lastMessageUnrespondedToSinceLastDone	Struct	N/A
	assignedUsers	나열	N/A
	assignedTeams	나열	N/A
	firstResponse	Struct	N/A
	firstResponseSinceLastDone	Struct	N/A
	lastResponse	Struct	N/A
	firstDone	Struct	N/A
	lastDone	Struct	N/A
	direction	String	=, !=, CONTAINS
	lastMessageDirection	String	해당 사항 없음
	outboundMessageCount	Integer	N/A
	inboundMessageCount	Integer	N/A
	rev	Integer	N/A

개체	필드	데이터 유형	지원되는 연산자
	priority	Integer	=, !=, >, >=, <, <=
	roleGroupVersions	나열	N/A
	accessOverride	나열	N/A
	어시스턴트	Struct	N/A
	phase	String	해당 사항 없음
	Skills	나열	N/A
	matchedTi meBasedRules	나열	N/A
고객	id	String	해당 사항 없음
	name	String	=, !=, CONTAINS
	displayName	String	해당 사항 없음
	displayColor	String	해당 사항 없음
	displayIcon	String	해당 사항 없음
	externalId	String	=, !=, CONTAINS
	externalIds	나열	N/A
	sharedExternalIds	나열	N/A
	emails	나열	N/A
	sharedEmails	나열	N/A
	phones	나열	N/A
	sharedPhones	나열	N/A
	whatsapps	나열	N/A

개체	필드	데이터 유형	지원되는 연산자
	facebookIds	나열	N/A
	instagramIds	나열	N/A
	socials	나열	N/A
	sharedSocials	나열	N/A
	urls	나열	N/A
	위치	나열	N/A
	activeUsers	나열	N/A
	watchers	나열	N/A
	recentLocation	Struct	N/A
	locale	String	=, !=, CONTAINS
	timeZone	String	해당 사항 없음
	gender	String	=, !=, CONTAINS
	createdAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	updatedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	modifiedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	lastActivityAt	DateTime	N/A
	deleted	불	N/A
	lastConversation	Struct	N/A
	conversationCounts	Struct	N/A

개체	필드	데이터 유형	지원되는 연산자
	preview	Struct	N/A
	tags	나열	N/A
	progressiveStatus	String	=, !=, CONTAINS
	verified	불	N/A
	rev	Integer	N/A
	recentItems	나열	N/A
	defaultLang	String	=, !=, CONTAINS
	satisfactionLevel	Struct	N/A
	roleGroupVersions	나열	N/A
	accessOverride	나열	N/A
	companyName	String	해당 사항 없음
	firstName	String	해당 사항 없음
	lastName	String	해당 사항 없음
Customer Searches Pinned	id	String	해당 사항 없음
	search	String	해당 사항 없음
	createdAt	DateTime	N/A
Customer Searches Positions	id	String	해당 사항 없음
	positions	나열	N/A
	children	나열	N/A
	createdAt	DateTime	N/A

개체	필드	데이터 유형	지원되는 연산자
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	rev	Integer	N/A
Email Hooks	id	String	해당 사항 없음
	설명	String	해당 사항 없음
	debug	불	N/A
	이메일	String	해당 사항 없음
	eventName	String	해당 사항 없음
	제목	String	해당 사항 없음
	hash	String	해당 사항 없음
	키	String	해당 사항 없음
	createdAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	updatedAt	DateTime	N/A
Web Hooks	id	String	해당 사항 없음
	설명	String	해당 사항 없음
	eventName	String	해당 사항 없음
	hash	String	해당 사항 없음
	url	String	해당 사항 없음
	createdAt	DateTime	N/A

개체	필드	데이터 유형	지원되는 연산자
	modifiedAt	DateTime	N/A
	updatedAt	DateTime	N/A
	제목	String	해당 사항 없음
	version	Integer	N/A
	debug	불	N/A
KB Articles	id	String	해당 사항 없음
	hash	String	해당 사항 없음
	제목	String	해당 사항 없음
	source	String	해당 사항 없음
	status	String	해당 사항 없음
	scope	String	해당 사항 없음
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
	deleted	불	N/A
	deletedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	publishedAt	DateTime	N/A
	tags	나열	N/A
	categories	나열	N/A
	knowledgeBases	나열	N/A

개체	필드	데이터 유형	지원되는 연산자
	metaTitle	String	해당 사항 없음
	metaDescription	String	해당 사항 없음
	metaKeywords	나열	N/A
	langVersions	Struct	N/A
	latestLangs	Struct	N/A
KB Categories	id	String	해당 사항 없음
	hash	String	해당 사항 없음
	createdAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	updatedAt	DateTime	N/A
	published	불	N/A
	positions	나열	N/A
	categoryPositions	나열	N/A
	root	불	N/A
	langs	Struct	N/A
KB Forms	id	String	해당 사항 없음
	name	String	해당 사항 없음
	slug	String	해당 사항 없음
	hash	String	해당 사항 없음
	본문	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	layout	나열	N/A
	layoutV2	나열	N/A
	componentsV2	Struct	N/A
	조건	Struct	N/A
	advanced	불	N/A
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
	publishedAt	DateTime	N/A
	modifiedAt	String	해당 사항 없음
	published	불	N/A
	snippets	나열	N/A
	recaptcha	불	N/A
	klass	String	해당 사항 없음
	채널	String	해당 사항 없음
	deflection	불	N/A
	formHookEnabled	불	N/A
	replyFrom	String	해당 사항 없음
	wcag	불	N/A
KB Routes	id	String	해당 사항 없음
	url	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	routableType	String	해당 사항 없음
	routableId	String	해당 사항 없음
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
KB Tags	id	String	해당 사항 없음
	name	String	해당 사항 없음
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
KB Templates	id	String	해당 사항 없음
	제목	String	해당 사항 없음
	설명	String	해당 사항 없음
	beta	불	N/A
	매니페스트	Struct	N/A
	jsxSnippets	나열	N/A
	이미지	나열	N/A
	version	String	해당 사항 없음
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A

개체	필드	데이터 유형	지원되는 연산자
KB Themes	id	String	해당 사항 없음
	name	String	해당 사항 없음
	활성화	불	N/A
	기본값	불	N/A
	lastfileUpdatedAt	DateTime	N/A
	사용자 지정	불	N/A
	status	String	해당 사항 없음
	templateVersionId	String	해당 사항 없음
	templateTitle	String	해당 사항 없음
	templateVersion	String	해당 사항 없음
	매니페스트	Struct	N/A
	configSnippets	나열	N/A
	jsxSnippets	나열	N/A
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	rev	Integer	N/A
Klasses	id	String	해당 사항 없음
	name	String	해당 사항 없음
	icon	String	해당 사항 없음

개체	필드	데이터 유형	지원되는 연산자
	color	String	해당 사항 없음
	appDisabled	불	N/A
	status	String	해당 사항 없음
	updatedAt	DateTime	N/A
	createdAt	DateTime	N/A
	s3DataUrl	String	해당 사항 없음
KViews	id	String	해당 사항 없음
	리소스	String	해당 사항 없음
	템플릿	String	해당 사항 없음
	context	String	해당 사항 없음
	meta	Struct	N/A
	appDisabled	불	N/A
	활성화	불	N/A
	advanced	불	N/A
	layout	나열	N/A
	구성 요소	Struct	N/A
	조건	Struct	N/A
	rev	Integer	N/A
	createdAt	DateTime	N/A
	modifiedAt	DateTime	N/A

개체	필드	데이터 유형	지원되는 연산자
	updatedAt	DateTime	N/A
알림	id	String	해당 사항 없음
	name	String	해당 사항 없음
	status	String	해당 사항 없음
	이벤트	Struct	N/A
	createdAt	DateTime	N/A
	updatedAt	DateTime	N/A
메시지	id	String	해당 사항 없음
	externalId	String	해당 사항 없음
	채널	String	=, !=, CONTAINS
	앱	String	해당 사항 없음
	size	Integer	=, !=, >, >=, <, <=
	direction	String	=, !=, CONTAINS
	preview	String	해당 사항 없음
	subject	String	해당 사항 없음
	meta	Struct	N/A
	status	String	=, !=, CONTAINS
	directionType	String	=, !=, CONTAINS
	assignedTeams	나열	N/A
	assignedUsers	나열	N/A

개체	필드	데이터 유형	지원되는 연산자
	errorAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	auto	불	=, !=
	sentAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	createdAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	updatedAt	DateTime	N/A
	modifiedAt	DateTime	N/A
	redacted	불	N/A
	createdByTeams	나열	N/A
	rev	Integer	N/A
	reactions	나열	N/A
	intentDetections	나열	N/A
Notes	id	String	해당 사항 없음
	본문	String	CONTAINS
	createdAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	updatedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	modifiedAt	DateTime	=, !=, <, <=, >, >=, BETWEEN
	createdByTeams	나열	N/A

분할 쿼리

필드 기반 분할

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  DateTime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2023-01-15T11:18:39.205Z"
  ```
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준 분할 필드 지원 세부 정보는 다음 표에 나와 있습니다.

엔터티 이름	분할 필드	데이터 유형
회사	modifiedAt	DateTime
Conversations	createdAt, updatedAt , modifiedAt, endedAt, lastMessageAt	DateTime
	messageCount, noteCount	BigInteger
	priority	Integer
고객	createdAt, updatedAt, modifiedAt	DateTime
메시지	errorAt, sentAt, createdAt	DateTime
	size	BigInteger

엔터티 이름	분할 필드	데이터 유형
Notes	createdAt, updatedAt, modifiedAt	DateTime

예시:

```
Kustomer_read = glueContext.create_dynamic_frame.from_options(
    connection_type="kustomer",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "conversation",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "createdAt"
        "LOWER_BOUND": "2023-01-15T11:18:39.205Z"
        "UPPER_BOUND": "2023-02-15T11:18:39.205Z"
        "NUM_PARTITIONS": "2"
    }
```

Kustomer 연결 옵션

다음은 Kustomer에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Stripe에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Kustomer Rest API 버전.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - (필수) 읽기에 사용됩니다. Kustomer 인스턴스 URL.

Kustomer 제한 사항

다음은 Kustomer의 제한 사항 또는 참고 사항입니다.

- Kustomer API 설명서에서 어떠한 엔드포인트도 선언하지 않았으므로 `Customer Searches` 엔터티는 지원되지 않습니다.
- `Klasses` 엔터티에 대한 필터링 및 증분 전송은 지원되지 않습니다.
- 순서 기준은 단일 요청의 여러 적용 가능한 필드에서 지원될 수 있습니다.

  하지만 여러 필드의 기능별 순서는 일부 조합에 대해 SaaS 종단에서 일관되지 않게 작동하는 것으로 관찰되었습니다. 잘못된 정렬 결과를 표시할 수 있는 'n'개의 조합이 있을 수 있으므로 예측할 수 없습니다. 예시:

  `Customers` 엔터티의 경우 `progressiveStatus desc, name asc`를 기준으로 정렬해도 정렬된 결과가 정확하지 않습니다. `progressiveStatus` 순서만을 기반으로 하여 정렬됩니다. 이러한 동작이 관찰되면 단일 필드를 사용하여 순서를 정할 수 있습니다.
- 'id' 필드의 순서 기준은 `Conversations` 및 `Messages` 엔터티에서만 쿼리 파라미터로 지원됩니다. 예: https://api.kustomerapp.com/v1/conversations?sort=desc('id'를 기준으로 결과를 내림차순으로 정렬)

  추가로 다른 필드의 다른 필터 또는 순서는 API 엔드포인트가 POST인 POST 요청 본문으로 변환됩니다(https://api.kustomerapp.com/v1/customers/search). `Conversations` 및 `Messages`에서 'id'에 의한 순서 지정 지원을 허용하려면 id에 의한 순서만 존재하거나 다른 적용 가능한 필드에 대한에 의한 다른 필터 및/또는 순서가 있어야 합니다.
- Kustomer는 필터링된 요청 또는 필터링되지 않은 요청과 무관하게 최대 10,000개의 레코드를 가져올 수 있습니다. 이러한 제한으로 인해 10,000개 이상의 레코드를 보유한 엔터티에 대한 데이터 손실이 발생합니다. 이 문제를 어느 정도 완화하기 위해 수행할 수 있는 두 가지 해결 방법이 있습니다.
  - 필터를 적용하여 특정 레코드 세트를 가져옵니다.
  - 필터가 적용된 레코드가 10,000개를 초과하는 경우 후속 신규 요청에 연속 필터 값을 적용하거나 필터에 범위를 적용합니다. 예시:

    첫 번째 요청의 filterExpression: `modifiedAt >= 2022-03-15T05:26:23.000Z and modifiedAt < 2023-03-15T05:26:23.000Z`

    이로 인해 10,000개 레코드 제한이 소진된다고 가정합니다.

    filterExpression `modifiedAt >= 2023-03-15T05:26:23.000Z`를 사용하여 다른 요청을 트리거할 수 있습니다.

- SaaS 동작으로 Kustomer의 CONTAINS 연산자는 단어 내에서 부분 일치가 아닌 전체 단어 일치만 지원합니다. 예: “body CONTAINS ‘test record’”는 ‘body’ 필드에 ‘test’가 있는 레코드와 일치합니다. 하지만 “body CONTAINS ‘test’”는 ‘body’ 필드에 ‘testAnotherRecord’가 있는 레코드와는 일치하지 않습니다.

LinkedIn에 연결

LinkedIn은 다양한 후원 게시물 및 기타 방법을 통해 LinkedIn 소셜 네트워크에 대한 액세스를 제공하는 유료 마케팅 도구입니다. LinkedIn은 B2B 회사가 리드, 온라인 인식, 콘텐츠 공유 등을 구축할 수 있는 강력한 마케팅 도구입니다.

주제

- AWS Glue의 LinkedIn 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- LinkedIn 구성
- LinkedIn 연결 구성
- LinkedIn 엔터티에서 읽기
- LinkedIn 연결 옵션
- LinkedIn 계정 생성
- 제한 사항

AWS Glue의 LinkedIn 지원

AWS Glue에서는 다음과 같이 LinkedIn을 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 LinkedIn에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 LinkedIn API 버전

202406(2024년 6월)

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

LinkedIn 구성

AWS Glue를 사용하여 LinkedIn에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- LinkedIn 계정이 있습니다. 계정 생성에 대한 자세한 내용은 LinkedIn 계정 생성을 참조하세요.
- LinkedIn 계정이 API 액세스에 대해 활성화되어 있습니다.
- LinkedIn 계정에 OAuth2 API 통합을 생성했습니다. 이 통합에서는 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하는 데 사용하는 클라이언트 자격 증명을 제공합니다. 자세한 내용은 the section called “LinkedIn 계정 생성” 단원을 참조하십시오.

이러한 요구 사항을 충족하면 LinkedIn 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 LinkedIn에서 다른 작업을 수행하지 않아도 됩니다.

LinkedIn 연결 구성

LinkedIn은 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 LinkedIn에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 LinkedIn으로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰 둘 다 모두 생성합니다. 액세스 토큰은 생성 후 60일 후에 만료됩니다. 새로 고침 토큰을 사용하여 새 액세스 토큰을 가져올 수 있습니다.

Authorization Code OAuth 흐름을 위해 연결된 앱을 생성하는 방법에 대한 퍼블릭 LinkedIn 설명서는 Authorization Code Flow (3-legged OAuth)를 참조하세요.

LinkedIn 연결 구성

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   - AWS 관리형 연결된 앱의 경우-비어 있는 보안 암호 또는 임시 값이 지정된 보안 암호입니다.

     > **Note**
     >
     > AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.

    1. 연결 유형을 선택할 때 LinkedIn을 선택합니다.
    2. LinkedIn 환경을 제공합니다.
    3. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

    4. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
    5. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

LinkedIn 엔터티에서 읽기

사전 조건

읽으려는 LinkedIn 객체입니다. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Ad Accounts	예	예	예	예	아니요
Campaigns	예	예	예	예	아니요
Campaign Groups	예	예	예	예	아니요
Creatives	예	예	예	예	아니요
Ad Analytics	예	아니요	아니요	예	아니요
Ad Analytics All AdAcocunts	예	아니요	아니요	예	아니요
Ad Analytics All Campaigns	예	아니요	아니요	예	아니요
Ad Analytics All CampaignGroups	예	아니요	아니요	예	아니요
Ad Analytics All AdCreatives	예	아니요	아니요	예	아니요
Share Statistics	예	아니요	아니요	예	아니요
Page Statistics	예	아니요	아니요	예	아니요
Follower Statistics	예	아니요	아니요	예	아니요

예

```
netsuiteerp_read = glueContext.create_dynamic_frame.from_options(
    connection_type="linkedin",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "adaccounts",
        "API_VERSION": "202406"
    }
)
```

LinkedIn 엔터티 및 필드 세부 정보

필드 데이터 유형	지원되는 필터 연산자
String	=
DateTime	BETWEEN, =
Numeric	=
Boolean	=

LinkedIn 연결 옵션

다음은 LinkedIn의 연결 옵션입니다.

- ENTITY_NAME(문자열)-(필수) 읽기/쓰기에 사용됩니다. LinkedIn에서의 객체 이름입니다. 예: adAccounts.
- API_VERSION(문자열)-(필수) 읽기/쓰기에 사용됩니다. 사용할 LinkedIn Rest API 버전. LinkedIn에서는 현재 버전 202406만 지원하므로 값은 202406입니다.
- SELECTED_FIELDS(List<String>)-기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 선택한 엔티티에 대해 선택하려는 열입니다.
- FILTER_PREDICATE(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

LinkedIn 계정 생성

LinkedIn 앱 및 OAuth 자격 증명 생성

1. LinkedIn 개발자 네트워크 페이지로 이동하여 LinkedIn 계정 자격 증명으로 로그인합니다.
2. 내 앱 페이지로 이동하여 애플리케이션 생성을 선택해 새 LinkedIn 앱을 생성합니다.
3. 앱 등록 양식에 다음 세부 정보를 입력합니다.
   - 회사 이름-기존 회사를 선택하거나 새 회사를 생성합니다.
   - 이름-애플리케이션 이름을 입력합니다.
   - 설명-애플리케이션 설명을 입력합니다.
   - 애플리케이션 로고-애플리케이션 로고로 이미지 파일을 선택합니다.
   - 애플리케이션 사용-애플리케이션 사용을 선택합니다.
   - 웹 사이트 URL-애플리케이션에 대한 자세한 정보가 포함된 웹 사이트 URL을 입력합니다.
   - 비즈니스 이메일-비즈니스 이메일 주소를 입력합니다.
   - 회사 전화-회사 전화 번호를 입력합니다.
   - LinkedIn API 사용 약관-읽고 동의하세요.
4. 앱 등록 양식이 완료되면 제출을 선택합니다.

   인증 키(클라이언트 ID와 클라이언트 보안 암호) 및 기타 관련 세부 정보가 표시되는 인증 페이지로 리디렉션됩니다.
5. 웹 애플리케이션에서 사용자의 LinkedIn 계정에서 사용자의 이메일 주소에 액세스해야 하는 경우 `r_emailaddress` 권한을 선택합니다. 또한 LinkedIn 애플리케이션에 대해 승인된 리디렉션 URL 지정할 수 있습니다.

LinkedIn 계정에서 페이지 생성

1. LinkedIn 개발자 제품으로 이동합니다.
2. LinkedIn 개발자 제품 페이지의 오른쪽 상단에서 내 앱을 선택합니다.
3. 내 앱 페이지의 오른쪽 상단에서 앱 생성을 선택합니다.
4. 앱 생성 페이지의 앱 이름 필드에 앱 이름을 입력합니다.
5. LinkedIn 페이지 필드에 회사 페이지 이름 또는 URL을 입력합니다.

> **Note**
>
> LinkedIn 페이지가 없는 경우 새 LinkedIn 생성을 선택하여 만들 수 있습니다.

6. 개인 정보 보호 정책 URL 필드에 개인 정보 보호 정책 URL을 입력합니다.
7. 로고 업로드를 선택하여 사용자가 앱으로 권한을 부여할 때 표시할 이미지를 업로드합니다.
8. 법적 계약 섹션에서 이 약관을 읽었으며 이에 동의합니다를 선택합니다.
9. 앱 생성을 선택합니다.

   새 앱이 생성되고 내 앱 탭에서 사용할 수 있습니다.

LinkedIn에 캠페인 광고 게시

1. Campaign Manager에 로그인합니다.
2. 기존 캠페인 그룹을 선택하거나 생성을 선택해 새로 만듭니다.
3. 목표를 선택합니다.
4. 그룹, 예산 및 일정을 선택합니다.
5. 대상 고객을 빌드합니다.
6. 광고 형식을 선택합니다.
7. 예산과 일정을 선택합니다.
8. 광고를 설정합니다.
9. 검토하고 실행합니다.

제한 사항

분석 필드 `ad_analytics_all_adAccounts`, `ad_analytics_all_campaigns`, `ad_analytics_all_campaign_groups`, `ad_analytics_all_adCreatives`의 경우 레코드를 검색하려면 필터가 필수입니다.

Mailchimp에 연결

Mailchimp는 클라이언트, 고객 및 기타 이해 당사자를 관리하고 이들과 대화하는 데 도움이 되는 올인원 마케팅 플랫폼입니다. 마케팅에 대한 Mailchimp의 접근 방식은 건전한 고객 응대 관리 관행, 정교하게 설계된 이메일, 고유한 자동 워크플로 및 강력한 데이터 분석에 중점을 둡니다. Mailchimp 사용자인

경우 Mailchimp 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, Mailchimp를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Mailchimp 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 Mailchimp 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Mailchimp 구성
- Mailchimp 연결 구성
- Mailchimp 엔터티에서 읽기
- Mailchimp 연결 옵션
- Mailchimp 계정 생성
- 제한 사항

AWS Glue의 Mailchimp 지원

AWS Glue는 다음과 같이 Mailchimp를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Mailchimp에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Mailchimp API 버전

3.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management 콘솔을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Mailchimp 구성

AWS Glue를 사용하여 Mailchimp에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호를 사용하는 Mailchimp 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Mailchimp 계정 생성을 참조하세요.
- AWS Glue에 대한 서비스 액세스 권한으로 AWS 계정을 생성해야 합니다.
- 다음 리소스 중 하나를 생성했는지 확인합니다. 이러한 리소스는 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하는 데 사용하는 자격 증명을 제공합니다.
  - OAuth 2.0 인증을 지원하는 개발자 앱. 개발자 앱 생성에 대한 자세한 내용은 Mailchimp 계정 생성을 참조하세요.

이러한 요구 사항을 충족하면 Mailchimp 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Mailchimp에서 다른 작업을 수행하지 않아도 됩니다.

Mailchimp 연결 구성

Mailchimp는 인증 메커니즘에 대해 다음 두 가지 유형을 지원합니다.

- Mailchimp는 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.
  - 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' `OAuth`로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Mailchimp Client ID 및 Client Secret을 제외한 `OAuth` 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Mailchimp로 리디렉션합니다. 사용자가 로그인하고 Mailchimp 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
  - 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Mailchimp에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Mailchimp로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
  - `AUTHORIZATION_CODE` `OAuth` 흐름에 대한 연결된 앱을 생성하는 방법에 대한 공개 Mailchimp 설명서는 Access Data on Behalf of Other Users with OAuth 2를 참조하세요.
- 사용자 지정 인증 - 사용자 지정 권한 부여에 필요한 API 키 생성에 대한 공개 Mailchimp 설명서는 About API Keys를 참조하세요.

다음 단계를 따라 Mailchimp 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - `OAuth` 인증 - 고객 관리형 연결된 앱: 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   - 사용자 지정 인증 - 고객 관리형 연결된 앱: 보안 암호는 키 역할을 하는 'api_key'와 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

> **Note**
>
> AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Mailchimp를 선택합니다.
   c. Mailchimp `instanceUrl`을 입력합니다.
   d. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
    "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "secretsmanager:DescribeSecret",
                    "secretsmanager:GetSecretValue",
                    "secretsmanager:PutSecretValue",
                    "ec2:CreateNetworkInterface",
                    "ec2:DescribeNetworkInterface",
                    "ec2:DeleteNetworkInterface",
                ],
                "Resource": "*"
            }
        ]
    }
```

   e. 인증 유형을 선택하여 Mailchimp에 연결합니다.
      - `0Auth` 인증 - 연결하려는 Mailchimp의 토큰 URL, 사용자 관리형 클라이언트 애플리케이션 ClientId를 입력합니다.
      - 사용자 지정 인증 - 인증 유형 사용자 지정을 선택하여 Mailchimp에 연결합니다.
   f. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   g. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 입력합니다.

Mailchimp 엔터티에서 읽기

사전 조건

읽으려는 Mailchimp 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

- Abuse-reports
- 자동화
- 캠페인
- Click-details
- Lists
- 회원
- Open-details
- Segments
- Stores
- Unsubscribed

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
자동화	예	예	예	예	예
Campaigns	아니요	아니요	아니요	아니요	아니요
Lists	예	예	아니요	예	예
Reports Abuse	아니요	예	아니요	예	예
Reports Open	아니요	예	아니요	예	예
Reports Click	예	예	아니요	예	예
Reports Unsubscribe	아니요	예	아니요	예	예
세그먼트	아니요	예	아니요	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Segment Members	예	예	아니요	예	아니요
Stores	예	예	예	예	아니요

예

```
mailchimp_read = glueContext.create_dynamic_frame.from_options(
            connection_type="mailchimp",
            connection_options={

                  "connectionName": "connectionName",

                  "ENTITY_NAME": "stores",
"INSTANCE_URL": "https://us14.api.mailchimp.com",
                  "API_VERSION": "3.0"
               })
```

Mailchimp 엔터티 및 필드 세부 정보

- Abuse-reports
- 자동화
- 캠페인
- Click-details
- Lists
- 회원
- Open-details
- Segments
- Stores
- Unsubscribed

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  DateTime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2024-07-01T00:00:00.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

다음 표에서는 엔터티 분할 필드 지원 세부 정보를 설명합니다.

엔터티 이름	분할 필드	데이터 유형

예시:

```
read_read = glueContext.create_dynamic_frame.from_options(
    connection_type="mailchimp",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "automations",
        "API_VERSION": "3.0",
        "INSTANCE_URL": "https://us14.api.mailchimp.com",
        "PARTITION_FIELD": "create_time",
        "LOWER_BOUND": "2024-02-05T14:09:30.115Z",
```

```
        "UPPER_BOUND": "2024-06-07T13:30:00.134Z",
        "NUM_PARTITIONS": "3"
    }
```

Mailchimp 연결 옵션

다음은 Mailchimp의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Mailchimp에서의 객체 이름.
- INSTANCE_URL(문자열) - (필수) 유효한 Mailchimp 인스턴스 URL.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Mailchimp Engage Rest API 버전. 예: 3.0.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Mailchimp 계정 생성

1. Mailchimp 로그인 페이지로 이동하여 이메일 ID와 암호를 입력한 다음 Sign up을 선택합니다.
2. Mailchimp의 확인 이메일을 열고 확인 링크를 선택하여 계정을 확인합니다.

> **Note**
>
> 활성화 이메일을 수신하는 데 걸리는 시간은 다를 수 있습니다. 활성화 이메일을 받지 못한 경우 스팸 폴더를 확인하고 활성화 이메일 문제 해결 팁을 읽어보세요.. Mailchimp는 admin@pottedplanter.com 또는 security@example.com과 같은 역할 기반 이메일 주소의 가입을 차단합니다.

계정에 처음 로그인할 때 Mailchimp에서 필수 정보 입력을 요청합니다. Mailchimp는 이 정보를 사용하여 계정이 사용 약관을 준수하는지 확인하고 사용자 및 회사의 요구 사항과 관련된 안내를 제공합니다.

3. 정보를 입력하고 프롬프트에 따라 활성화 프로세스를 완료한 다음 새 Mailchimp 계정을 시작합니다.

`OAuth2.0` 애플리케이션 등록

1. Mailchimp login 페이지로 이동하여 이메일 ID와 암호를 입력한 다음 Log in을 선택합니다.
2. 오른쪽 상단 모서리에서 User 아이콘을 선택한 다음 드롭다운 메뉴에서 Account and billing을 선택합니다.
3. Extras를 선택하고 드롭다운 메뉴에서 Registered apps를 선택합니다.
4. Register An App을 찾아 선택합니다.
5. 다음 세부 정보를 입력합니다.
   - App name – 앱의 이름.
   - Company / Organization – 회사 또는 조직의 이름.
   - App website – 앱의 웹사이트.
   - Redirect URI - 리디렉션 URI 패턴은 로그인 흐름이 완료될 때 Mailchimp가 리디렉션(요청된 경우)할 수 있는 URI 경로(또는 쉼표로 구분된 경로 목록)입니다. 예제: `https://ap-southeast-2\\.console\\.aws\\.amazon\\.com`
6. 생성(Create)을 선택합니다.
7. Client ID와 Client Secret이 표시됩니다. 복사하여 안전한 위치에 저장합니다. 그런 다음 완료를 선택합니다.

> **Note**
>
> Client ID 및 Client Secret 문자열은 AppFlow 또는 AWS Glue를 사용할 때 이 커넥터와의 연결을 설정하는 데 사용되는 자격 증명입니다.

API 키 생성

1. Mailchimp login 페이지로 이동하여 이메일 ID와 암호를 입력한 다음 Log in을 선택합니다.

2. 오른쪽 상단 모서리에서 User 아이콘을 선택한 다음 드롭다운 메뉴에서 Account and billing을 선택합니다.
3. Extras를 선택하고 드롭다운 메뉴에서 API keys를 선택합니다.
4. Create A Key를 선택합니다.
5. 키 이름을 입력하고 Generate Key를 선택합니다.

   다음 페이지에는 생성된 API 키가 표시됩니다.
6. 키를 복사하고 안전하게 저장한 다음 Done을 선택합니다.

제한 사항

다음은 Mailchimp 커넥터의 제한 사항입니다.

- 필터링은 `Campaigns`, `Automations`, `Lists`, `Open Details`, `Members` 및 `Segments` 엔터티에서만 지원됩니다.
- `DateTime` 데이터 유형 필드에 필터를 사용하는 동안 `yyyy-mm-ddThh:MM:ssZ` 형식으로 값을 전달해야 합니다.

Microsoft Teams에 연결

Microsoft Teams는 Microsoft 365 내의 협업형 워크스페이스로, 업무 관련 대화, 협업, 화상 채팅 및 문서 공유를 위한 중앙 허브 역할을 하며, 통합된 도구 모음을 통해 작업자 생산성을 지원하도록 설계되었습니다.

주제

- AWS Glue의 Microsoft Teams 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Microsoft Teams 구성
- Microsoft Teams 연결 구성
- Microsoft Teams 엔터티에서 읽기
- Microsoft Teams 연결 옵션 참조
- 제한 사항
- 새 Microsoft Teams 계정을 생성합니다.

AWS Glue의 Microsoft Teams 지원

AWS Glue는 다음과 같이 Microsoft Teams를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Microsoft Teams에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Microsoft Teams API 버전

v1. 버전별 엔터티 지원은 소스에 대해 지원되는 엔터티를 참조하세요.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절

차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Microsoft Teams 구성

AWS Glue를 사용하여 Microsoft Teams에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호가 있는 Microsoft Teams 개발자 계정이 있습니다. 자세한 내용은 새 Microsoft Teams 계정을 생성합니다. 단원을 참조하십시오.
- 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하기 위해 사용하는 클라이언트 ID와 클라이언트 자격 증명을 제공하는 Microsoft Teams 계정에 OAuth2 앱을 설정해야 합니다. 자세한 내용은 새 Microsoft Teams 계정을 생성합니다. 단원을 참조하십시오.

이러한 요구 사항을 충족하면 Microsoft Teams 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Microsoft Teams에서 다른 작업을 수행하지 않아도 됩니다.

Microsoft Teams 연결 구성

Microsoft Teams는 인증 메커니즘에 대해 다음 두 가지 유형을 지원합니다.

1. OAuth 인증: Microsoft Teams는 OAuth2에 대해 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.
   - 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Microsoft Teams instanceurl을 제외한 OAuth 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Microsoft Teams로 리디렉션합니다. 사용자가 로그인하고 Microsoft Teams 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
   - 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Microsoft Teams에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나

리오에서는 여전히 Microsoft Teams로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 1시간 동안 활성 상태가 되고 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 공개 Microsoft Teams 설명서는 Microsoft Learn의 Register an application with the Microsoft identity platform - Microsoft Graph를 참조하세요.

다음 단계를 따라 Microsoft Teams 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보를 사용하여 보안 암호를 생성합니다. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. OAuth 인증:
      - 고객 관리형 연결된 앱 - 보안 암호는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 데이터 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Microsoft Teams를 선택합니다.
   c. Microsoft Teams 테넌트 ID를 입력합니다.
   d. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

e. Microsoft Teams 앱의 사용자 관리형 클라이언트 애플리케이션 ClientId를 입력합니다.

f. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

g. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다. Next(다음)를 선택합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

Microsoft Teams 엔터티에서 읽기

사전 조건

- 읽으려는 Microsoft Teams 객체. 객체 이름(예: team 또는 channel-message)이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티

모든 개체는 API 버전 1.0에서 지원됩니다.

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
팀	아니요	아니요	아니요	예	아니요
Team Members	예	예	아니요	예	예
Groups	예	예	예	예	예
Group Member	예	예	아니요	예	아니요
채널	예	아니요	아니요	예	예
Channel Messages	아니요	예	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Channel Message Replies	아니요	예	아니요	예	아니요
Channel Tabs	예	아니요	아니요	예	아니요
Chats	예	예	예	예	예
Calendar Events	예	예	예	예	예

예

```
MicrosoftTeams_read = glueContext.create_dynamic_frame.from_options(
    connection_type="MicrosoftTeams",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "company",
        "API_VERSION": "v1.0"
    }
```

Microsoft Teams 엔터티 및 필드 세부 정보

엔터티 목록:

- Team: https://docs.microsoft.com/en-us/graph/api/user-list-joinedteams?view=graph-rest-1.0
- Team-Member: https://docs.microsoft.com/en-us/graph/api/team-list-members?view=graph-rest-1.0
- Group: https://docs.microsoft.com/en-us/graph/api/group-list?view=graph-rest-1.0
- Group-Member: https://docs.microsoft.com/en-us/graph/api/group-list-members?view=graph-rest-1.0
- Channel: https://docs.microsoft.com/en-us/graph/api/channel-list?view=graph-rest-1.0
- Channel-Message: https://docs.microsoft.com/en-us/graph/api/channel-list-messages?view=graph-rest-1.0

- Channel-Message-Reply: https://docs.microsoft.com/en-us/graph/api/chatmessage-list-replies?view=graph-rest-1.0
- Channel-Tab: https://docs.microsoft.com/en-us/graph/api/channel-list-tabs?view=graph-rest-1.0
- Chat: https://docs.microsoft.com/en-us/graph/api/chat-list?view=graph-rest-1.0
- Calendar-Event: https://docs.microsoft.com/en-us/graph/api/group-list-events?view=graph-rest-1.0

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-02-06".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준 분할 필드 지원 세부 정보는 아래 표에 나와 있습니다.

Entity Name	Partitioning Fields	데이터 형식
Team Members	visibleHistoryStartDateTime	DateTime
Groups	createdDateTime	DateTime
채널	createdDateTime	DateTime
Chats	createdDateTime, lastModifiedDateTime	DateTime
Calendar Events	createdDateTime, lastModifiedDateTime, originalStart	DateTime

예

```
microsoftteams_read = glueContext.create_dynamic_frame.from_options(
    connection_type="MicrosoftTeams",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "group",
        "API_VERSION": "v1.0",
        "PARTITION_FIELD": "createdDateTime"
        "LOWER_BOUND": "2022-07-13T07:55:27.065Z"
        "UPPER_BOUND": "2022-08-12T07:55:27.065Z"
        "NUM_PARTITIONS": "2"
    }
```

Microsoft Teams 연결 옵션 참조

다음은 Microsoft Teams에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Microsoft Teams의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Microsoft Teams Rest API 버전. 예: v1.0.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

제한 사항

다음은 Microsoft Teams 커넥터에 대한 제한 사항입니다.

- Microsoft Teams API는 Chat 및 Team Member 엔터티에 지정된 것보다 적은 수의 레코드를 반환합니다. 이 문제는 Microsoft Teams Support에 보고되었으며, 현재 조사 중입니다.

새 Microsoft Teams 계정을 생성합니다.

1. Microsoft Teams 홈페이지인 https://account.microsoft.com/account/로 이동하여 로그인을 선택합니다.
2. 계정을 만드세요!를 선택합니다.
3. 계정 생성에 필요한 정보를 입력하고 새 계정을 생성합니다.
4. Microsoft Teams 웹사이트인 https://www.microsoft.com/en-in/microsoft-teams/log-in으로 이동합니다.
5. 방금 생성한 Microsoft 계정을 사용하여 가입합니다.
6. Teams에 성공적으로 가입한 후 https://account.microsft.com/services로 이동합니다.
7. Microsoft 365 평가판 사용을 선택합니다.
8. 아래 Microsoft 365 또는 Microsoft Teams 구독 중 하나를 활성화하여 Microsoft Teams 커넥터의 모든 필수 기능에 액세스합니다.

   - Microsoft Teams Essentials
   - Microsoft 365 Business
   - Microsoft 365 Business Basic
   - Microsoft 365 Business Standard
   - Microsoft 365 Business Premium

관리형 클라이언트 앱을 생성합니다.

1. 관리형 애플리케이션을 생성하려면 Microsoft Entra(이전 Azure Active Directory)에 새 OAuth 앱을 등록해야 합니다.
2. Microsoft Entra 관리 센터에 로그인합니다.
3. 여러 테넌트에 액세스할 수 있는 경우 상단 메뉴의 설정 아이콘을 사용하여 디렉터리 + 구독 메뉴에서 애플리케이션을 등록하려는 테넌트로 전환합니다.
4. ID > 애플리케이션 > 앱 등록으로 이동하여 새 등록을 선택합니다.
5. 새 애플리케이션의 표시 이름을 입력합니다.
6. 지원되는 계정 유형 섹션에서 애플리케이션을 사용할 수 있는 사용자를 지정합니다. 이 앱을 전역적으로 설정하려면 '조직 디렉터리의 계정' 또는 '조직 디렉터리 및 개인 Microsoft 계정의 계정'을 선택합니다.
7. 리디렉션 URI `https://{region}.console.aws.amazon.com/appflow/oauth`를 입력합니다. 예를 들어 `us-west-2 region`의 경우 `https://us-`

west-2.console.aws.amazon.com/appflow/oauth를 추가합니다. 사용하려는 여러 리전에 대해 여러 URL을 추가할 수 있습니다.

8. 앱을 등록합니다.
9. 나중에 사용할 수 있도록 클라이언트 ID를 기록해 둡니다.
10. Essentials 섹션에서 인증서 또는 암호 추가를 선택합니다.
11. 새 클라이언트 암호를 선택합니다.
12. 설명 및 만료 기간을 입력합니다.
13. 나중에 사용할 수 있도록 클라이언트 보안 암호를 복사하고 저장합니다.
14. 왼쪽 메뉴 목록에서 API 권한을 선택합니다.
15. 권한 추가를 선택합니다.
16. 'Microsoft Graph'를 선택합니다.
17. '위임된 권한'을 선택합니다.
18. 다음 권한을 모두 확인합니다.

    - User.Read
    - Offline_access
    - User.Read.All
    - User.ReadWrite.All
    - TeamsTab.ReadWriteForTeam
    - TeamsTab.ReadWriteForChat
    - TeamsTab.ReadWrite.All
    - TeamsTab.Read.All
    - TeamSettings.ReadWrite.All
    - TeamSettings.Read.All
    - TeamMember.ReadWrite.All
    - TeamMember.Read.All
    - Team.ReadBasic.All
    - GroupMember.ReadWrite.All
    - GroupMember.Read.All
    - Group.ReadWrite.All
    - Group.Read.All

- Directory.ReadWrite.All
- Directory.Read.All
- Directory.AccessAsUser.All
- Chat.ReadWrite
- Chat.ReadBasic
- Chat.Read
- ChannelSettings.ReadWrite.All
- ChannelSettings.Read.All
- ChannelMessage.Read.All
- Channel.ReadBasic.All

19. 권한 추가를 선택합니다. 이제 앱이 성공적으로 설정되었습니다. 클라이언트 ID와 클라이언트 보안 암호를 사용하여 새 연결을 생성할 수 있습니다. 자세한 내용은 https://learn.microsoft.com/en-us/graph/auth-register-app-v2를 참조하세요.

Mixpanel에 연결

Mixpanel은 기업이 사용자 참여를 측정하고 최적화하는 데 도움이 되는 강력한 실시간 분석 플랫폼입니다. Mixpanel은 고객 행동 추적에 사용되는 앱입니다. 이를 통해 사용자가 제품에 참여하는 방식을 추적하고 클릭 몇 번으로 결과를 쿼리하며 시각화할 수 있는 대화형 보고서를 통해 이 데이터를 분석할 수 있습니다. Mixpanel 사용자는 Mixpanel 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, Mixpanel을 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Mixpanel 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 Mixpanel 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Mixpanel 구성
- Mixpanel 연결 구성
- Mixpanel 엔터티에서 읽기
- Mixpanel 연결 옵션
- Mixpanel 계정 생성 및 클라이언트 앱 구성
- 제한 사항

AWS Glue의 Mixpanel 지원

AWS Glue에서는 다음과 같이 Mixpanel을 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Mixpanel에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Mixpanel API 버전

2.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절

차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Mixpanel 구성

AWS Glue를 사용하여 Mixpanel에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- Mixpanel 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Mixpanel 계정 생성을 참조하세요.
- Mixpanel 계정이 API 액세스에 대해 활성화되어 있습니다. API 액세스는 기본적으로 Enterprise, Unlimited, Developer, Performance 에디션에 대해 활성화됩니다.

이러한 요구 사항을 충족하면 Mixpanel 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Mixpanel에서 다른 작업을 수행하지 않아도 됩니다.

Mixpanel 연결 구성

Mixpanel은 `BasicAuth`의 사용자 이름과 암호를 지원합니다. 기본 인증은 클라이언트가 보호된 리소스에 액세스하기 위해 자격 증명을 직접 제공하는 간단한 인증 방법입니다. AWS Glue는 사용자 이름과 암호를 사용하여 Mixpanel API를 인증합니다.

`BasicAuth` 흐름에 대한 퍼블릭 Mixpanel 설명서는 Mixpanel 서비스 계정 섹션을 참조하세요.

Mixpanel 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - 기본 인증의 경우 보안 암호에는 `USERNAME`와 `PASSWORD`를 키로 사용하여 연결된 앱 소비자 보안 암호가 포함되어야 합니다.

     > **Note**
     >
     > AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.

a. 연결 유형을 선택할 때 Mixpanel을 선택합니다.

b. 연결하려는 Mixpanel의 `INSTANCE_URL`을 제공합니다.

c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려면 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Mixpanel 엔터티에서 읽기

사전 조건

데이터를 읽으려는 `Funnels`, `Retention` 또는 `Retention  Funnels`와 같은 Mixpanel 객체가 있어야 합니다. 또한 객체 이름을 알아야 합니다.

지원되는 엔터티

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Funnels	예	아니요	아니요	예	아니요

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Retention	예	아니요	아니요	예	아니요
Segmentation	예	아니요	아니요	예	아니요
Segmentation Sum	예	아니요	아니요	예	아니요
Segmentation Average	예	아니요	아니요	예	아니요
Cohorts	예	아니요	아니요	예	아니요
Engage	아니요	예	아니요	예	아니요
Events	예	아니요	아니요	예	아니요
Events Top	예	아니요	아니요	예	아니요
Events Names	예	아니요	아니요	예	아니요
Events Properties	예	아니요	아니요	예	아니요
Events Properties Top	예	아니요	아니요	예	아니요
Events Properties Values	예	아니요	아니요	예	아니요
Annotations	예	아니요	아니요	예	아니요
Profile Event Activity	예	아니요	아니요	예	아니요

예

```
mixpanel_read = glueContext.create_dynamic_frame.from_options(
    connection_type="mixpanel",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "/cohorts/list?project_id=2603353",
        "API_VERSION": "2.0",
        "INSTANCE_URL": "https://www.mixpanel.com/api/app/me"
    }
```

Mixpanel 엔터티 및 필드 세부 정보

엔터티	필드	데이터 형식	지원되는 연산자
Funnel	funnel_id	Integer	'='
	workspace_id	Integer	'='
	from_date	Date	'='
	to_date	Date	'=’
	length	Integer	'='
	length_unit	String	'='
	interval	Integer	'='
	unit	String	'='
	limit	Integer	'='
	data	Struct	
	meta	Struct	
Retention	workspace_id	Integer	'='
	unit	String	'='
	addiction_unit	String	'='

엔터티	필드	데이터 형식	지원되는 연산자
	from_date	Date	'='
	to_date	Date	'='
	event	String	'='
	limit	Integer	'='
	data	Struct	
Segmentation	workspace_id	Integer	'='
	event	String	'='
	from_date	Date	'='
	to_date	Date	'='
	unit	String	'='
	interval	Integer	'='
	limit	Integer	'='
	type	String	'='
	series	List	
	values	Struct	
	data	Struct	
Segmentation Numeric	workspace_id	Integer	'='
	event	String	'='
	on	String	'='
	from_date	Date	'='

엔터티	필드	데이터 형식	지원되는 연산자
	to_date	Date	'='
	unit	String	'='
	type	String	'='
	series	List	
	values	Struct	
Segmentation Sum	workspace_id	Integer	'='
	event	String	'='
	on	String	'='
	from_date	Date	'='
	to_date	Date	'='
	unit	String	'='
	metadata	Struct	
	results	Struct	
Segmentation Average	workspace_id	Integer	'='
	event	String	'='
	on	String	'='
	from_date	Date	'='
	to_date	Date	'='
	unit	String	'='
	metadata	Struct	

엔터티	필드	데이터 형식	지원되는 연산자
	results	Struct	
Cohorts	count	Integer	
	is_visible	Integer	
	description	String	
	created	DateTime	
	project_id	Integer	
	id	BigInteger	
	name	String	
	data_group_id	String	
Engage	distinct_id	String	
		properties	Struct
Event	WorkSpace	Integer	'='
	event	String	'='
	type	String	'='
	unit	String	'='
	interval	Integer	'='
	from_date	Date	'='
	to_date	Date	'='
	series	List	
	values	Struct	

엔터티	필드	데이터 형식	지원되는 연산자
Events Top	type	String	'='
	workspace_id	Integer	'='
	limit	Integer	'='
	amount	Integer	
	event	String	
	percent_change	Float	
Event Name	data	List	
	workspace_id	Integer	'='
	type	String	'='
	limit	Integer	'='
Event Properties	workspace_id	Integer	'='
	event	String	'='
	name	String	'='
	type	String	'='
	unit	String	'='
	interval	Integer	'='
	from_date	Date	'='
	to_date	Date	'='
	limit	Integer	'='
	data	Struct	

엔터티	필드	데이터 형식	지원되는 연산자
	series	List	
	values	Struct	
Event Properties Top	workspace_id	Integer	'='
	event	String	'='
	limit	Integer	'='
	data	Struct	
Event Properties Value	workspace_id	Integer	'='
	event	String	'='
	limit	Integer	'='
	name	String	'='
	data	List	
Annotation	workspace_id	Integer	
	date	DateTime	
	project_id	Integer	
	id	BigInteger	
	description	String	
	from_date	Date	BETWEEN
Profile Event Activity	workspace_id	Integer	'='
	distinct_ids	String	'='
	from_date	Date	'='

엔터티	필드	데이터 형식	지원되는 연산자
	to_date	Date	'='
	event	String	
	properties	Struct	

Mixpanel 연결 옵션

다음은 Mixpanel의 연결 옵션입니다.

- `ENTITY_NAME`(문자열)-(필수) 읽기/쓰기에 사용됩니다. Mixpanel에서의 객체 이름입니다.
- `API_VERSION`(문자열)-(필수) 읽기/쓰기에 사용됩니다. 사용할 Mixpanel Rest API 버전. 예: v2.0.
- `SELECTED_FIELDS`(List<String>)-기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Mixpanel 계정 생성 및 클라이언트 앱 구성

Mixpanel 계정 생성

1. Mixpanel 홈 페이지/로 이동합니다.
2. Mixpanel 홈 페이지의 오른쪽 상단 모서리에서 가입을 선택합니다.
3. 시작하기 페이지에서 다음 작업을 완료합니다.

   - 지정된 필드에 이메일 주소를 입력합니다.
   - 필수 확인란을 선택하여 약관에 동의합니다.
   - 시작하려면 시작하기를 선택합니다.

   완료하면 확인 이메일을 받게 됩니다.
4. 이메일 받은 편지함에서 확인 메시지를 확인하고 이메일을 연 다음 지침에 따라 이메일 주소를 확인합니다.
5. 확인 페이지에서 이메일 확인을 선택하여 이메일 확인을 완료합니다.

6. 조직 이름 지정 페이지에서 조직 이름을 입력하고 다음을 선택합니다.
7. 첫 번째 프로젝트 페이지에서 프로젝트 세부 정보를 입력하고 생성을 선택합니다.
8. 다음 페이지에서 시작하기를 선택하여 계정 생성을 완료합니다.

Mixpanel 계정에 로그인

1. Mixpanel 로그인 페이지/로 이동합니다.
2. 이메일 주소를 입력하고 계속을 선택합니다.
3. 이메일 받은 편지함에서 확인 메시지를 확인하고 이메일을 연 다음 지침에 따라 이메일 주소를 확인합니다.
4. 다음 페이지에서 로그인 버튼을 선택하여 계정에 로그인합니다.

Mixpanel 요금제 구매

1. Mixpanel 페이지에서 페이지 오른쪽 상단에 있는 설정 아이콘을 선택합니다.
2. 옵션 목록에서 요금제 세부 정보 및 결제를 선택합니다.
3. 요금제 세부 정보 및 결제 페이지에서 업그레이드 또는 수정을 선택합니다.
4. 다음 페이지에서 구매하려는 요금제를 선택합니다.

   이렇게 하면 계정 생성 및 요금제 구매 프로세스가 완료됩니다.

사용자 이름 및 클라이언트 보안 암호 생성(앱 등록)

1. Mixpanel 페이지에서 페이지 오른쪽 상단에 있는 설정 아이콘을 선택합니다."
2. 옵션 목록에서 프로젝트 설정을 선택합니다.
3. 프로젝트 설정 페이지에서 서비스 계정을 선택한 다음 서비스 계정 추가를 선택합니다.
4. 서비스 계정 드롭다운 목록에서 서비스 계정을 선택하거나 생성할 이름을 입력하고 프로젝트 역할을 추가한 다음 만료를 지정 후 추가를 선택합니다.

> **Important**
>
> 이전 단계를 완료한 후 다음 페이지에 서비스 계정의 보안 암호 키가 표시됩니다. 서비스 계정의 보안 암호 키를 저장해야 합니다. 이 시점 이후에는 다시 액세스할 수 없습니다.

제한 사항

다음은 Mixpanel 커넥터의 제한 사항입니다.

- `Segmentation Numeric` 엔터티의 경우 필수 필터에 대한 숫자 데이터를 찾을 수 없는 경우 Mixpanel API에서 `400 (Bad Request)` 오류가 발생합니다. 흐름 실패를 방지하기 위해 `OK` 응답으로 이것을 처리하고 있습니다.
- 다음과 같은 이유로 쿼리 가능한 `limit` 필드가 지원되는 엔터티에서 제거되었습니다.
  - SDK의 제한 특성으로 해석되어 오류 발생
  - 필터는 실질적인 용도가 없음
  - 이제 제한 특성 구현에서 동등한 기능을 다룸
- SaaS 플랫폼에서 분할하는 데 필요한 연산자(>=, <=, <, >, `between`)가 없으므로 필드 기반 분할을 지원할 수 없습니다. `between` 연산자를 지원하지만 이 연산자를 지원하는 필드는 검색할 수 없습니다. 따라서 필드 기반 분할의 기준이 충족되지 않습니다.
- 페이지 매김을 지원하는 엔터티에 대해 '오프셋' 값을 가져오는 프로비저닝이 없으므로 Mixpanel에 대한 레코드 기반 분할을 지원할 수 없습니다.
- `Cohorts` 엔터티는 `CreatedDate/Time` 필드만 지원하며 `UpdatedDate/Time`을 식별할 수 있는 필드가 없기 때문에 결과적으로 `DML_Status`를 식별할 수 없습니다. 또한 삭제된 레코드를 식별할 엔드포인트가 없습니다. 따라서 CDC는 지원되지 않습니다.
- 아래 언급된 엔터티에 대해 AWS Glue 작업을 실행하려면 필수 필터가 필요합니다. 엔터티 이름 및 필요한 필터는 아래 표를 참조하세요.

  엔터티 이름 및 필수 필터

엔터티 이름	필수 필터
Annotations	from_date, to_date
Cohorts	없음
Engage	없음
Event	event, type, unit, from_date, to_date, on
Events Name	type
Events Properties	event, name, type, unit, from_date, to_date

엔터티 이름	필수 필터
Events Properties Top	event
Events Properties Values	event, name
Events Top	type
Funnels	funnel_id, from_date, to_date
Profile Event Activity	distinct_ids, from_date, to_date
Retention	from_date, to_date, unit, addiction_unit
Segmentation	event, from_date, to_date
Segmentation Average	event, from_date, to_date, on
Segmentation Numeric	event, from_date, to_date, on
Segmentation Sum	event, from_date, to_date, on

Monday에 연결

Monday.com은 프로젝트 관리 및 팀 협업을 간소화하는 다목적 업무 운영 체제입니다. 사용자 지정 가능한 워크플로, 시각적 대시보드 및 자동화 도구로 생산성을 높입니다. 사용자는 하나의 통합 플랫폼에서 작업을 추적하고, 리소스를 관리하며, 효과적으로 커뮤니케이션할 수 있습니다.

주제

- AWS Glue의 Monday 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Monday 구성
- Monday 연결 구성
- Monday 엔터티에서 읽기
- Monday 연결 옵션 참조
- 제한 사항
- 새 Monday 계정을 생성합니다.

AWS Glue의 Monday 지원

AWS Glue에서는 다음과 같이 Monday를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Monday에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Monday API 버전

v2.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절

차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Monday 구성

AWS Glue를 사용하여 Monday에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호가 있는 Monday 개발자 계정이 있습니다. 자세한 내용은 새 Monday 계정을 생성합니다. 단원을 참조하십시오.
- Monday 개발자 계정이 API 액세스에 대해 활성화되어 있습니다. 평가판 기간 내에 추가 비용 없이 모든 Monday API를 사용할 수 있습니다. 평가판 기간이 끝나면 구독을 구입하여 데이터를 생성하고 액세스해야 합니다. 자세한 내용은 Monday 라이선싱 페이지를 참조하세요.

이러한 요구 사항을 충족하면 Monday 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Monday에서 다른 작업을 수행하지 않아도 됩니다.

Monday 연결 구성

Monday는 인증 메커니즘에 대해 다음 두 가지 유형을 지원합니다.

1. OAuth 인증: Monday는 OAuth2에 대해 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.
   - 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Monday instanceurl을 제외한 OAuth 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Monday로 리디렉션합니다. 사용자가 로그인하고 Monday 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
   - 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Monday에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택해야 합니다. 이 시나리오에서는 여전히 Monday로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 1시간 동안 활성 상태가 되고 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 자세한 내용은 AUTHORIZATION_CODE OAuth 흐름을 위한 연결된 앱 생성 관련 설명서를 참조하세요.

2. 사용자 지정 인증:
   - 사용자 지정 권한 부여에 필요한 API 키 생성에 대한 공개 Monday 설명서는 https://developer.monday.com/api-reference/docs/authentication#api-token-permissions 페이지를 참조하세요.

다음 단계를 따라 Monday 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보를 사용하여 보안 암호를 생성합니다. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. OAuth 인증:
      - 고객 관리형 연결된 앱 - 보안 암호는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 사용자 지정 인증:
      - 고객 관리형 연결된 앱 - 보안 암호는 키 역할을 하는 `personalAccessToken`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 데이터 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Monday를 선택합니다.
   c. Monday instanceURL을 입력합니다.
   d. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
```

```
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

e. 인증 유형을 선택하여 Monday에 연결

- OAuth 인증: 연결하려는 Monday의 토큰 URL 및 사용자 관리형 클라이언트 애플리케이션 ClientId를 입력합니다.
- 사용자 지정 인증: 인증 유형으로 사용자 지정을 선택하여 Monday에 연결합니다.

f. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

g. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다. Next(다음)를 선택합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

Monday 엔터티에서 읽기

사전 조건

- 읽으려는 Monday 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

소스에 대해 지원되는 엔터티

엔터티 목록:

- Account: https://developer.monday.com/api-reference/docs/account#queries
- Board: https://developer.monday.com/api-reference/docs/boards#queries
- Column: https://developer.monday.com/api-reference/docs/columns#queries
- Docs: https://developer.monday.com/api-reference/docs/docs#queries
- Document Block: https://developer.monday.com/api-reference/docs/blocks#queries
- Files: https://developer.monday.com/api-reference/docs/files#queries
- Folders: https://developer.monday.com/api-reference/docs/folders#queries

- Groups: https://developer.monday.com/api-reference/docs/groups#queries
- Item: https://developer.monday.com/api-reference/docs/items#queries
- Subitems: https://developer.monday.com/api-reference/docs/subitems#queries
- Tags: https://developer.monday.com/api-reference/docs/tags-queries#queries
- Teams: https://developer.monday.com/api-reference/docs/teams#queries
- Updates: https://developer.monday.com/api-reference/docs/updates#queries
- Users: https://developer.monday.com/api-reference/docs/users#queries
- Workspaces: https://developer.monday.com/api-reference/docs/workspaces#queries

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Account	아니요	아니요	아니요	예	아니요
Boards	예	예	아니요	예	아니요
열	아니요	아니요	아니요	예	아니요
Docs	예	예	아니요	예	아니요
Document Blocks	아니요	예	아니요	예	아니요
파일	예	아니요	아니요	예	아니요
Groups	아니요	아니요	아니요	예	아니요
Item	예	예	아니요	예	아니요
Subitems	아니요	아니요	아니요	예	아니요
Tags	예	아니요	아니요	예	예
팀	예	아니요	아니요	예	아니요
업데이트	아니요	예	아니요	예	아니요
Users	예	예	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
WorkSpaces	예	예	아니요	예	아니요
폴더	예	예	아니요	예	아니요

예

```
monday_read = glueContext.create_dynamic_frame.from_options(
     connection_type="monday",
     connection_options={
         "connectionName": "connectionName",
         "ENTITY_NAME": "account",
         "API_VERSION": "v2"
     }
```

Monday 연결 옵션 참조

다음은 Monday의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Monday에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용하려는 Monday Rest API 버전입니다. 예: v2.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

제한 사항

다음은 Monday 커넥터의 제한 사항입니다.

- 동적 메타데이터 응답은 아래에 언급된 설명서와 일부 충돌합니다.
  - Group, Column 엔터티는 필터 작업을 지원하지만 동적 메타데이터 엔드포인트에 존재하지 않으므로 필터링할 수 없는 개체로 유지됩니다.

- 동적 엔드포인트는 약 15,000개 이상의 라인으로 구성되고 단일 응답으로 모든 엔터티의 메타데이터를 반환합니다. 따라서 필드가 로드되는 데 평균 10초가 걸리므로 작업 실행에 더 많은 시간이 필요합니다.
- Monday 속도 제한은 아래 표를 참조하세요. 동적 엔터티의 응답 데이터 크기가 크면 상당한 지연이 발생하며, 필드 로드에 평균 10초가 필요합니다.

복잡성 제한	5,000,000(5M)개의 복잡성 지점
일일 호출 제한	Pro 플랜의 경우 10,000개
분 제한	분당 500개 쿼리
동시성 제한	Pro 플랜의 경우 최대 동시 요청 100개

새 Monday 계정을 생성합니다.

1. Monday 홈페이지인 https://monday.com/으로 이동하고 Login을 선택합니다.
2. 로그인 페이지로 리디렉션됩니다. 페이지 하단에서 Sign up을 선택합니다.
3. 이메일 주소를 입력하고 계속을 선택합니다. 또는 Google을 통해 로그인할 수 있습니다.
4. 필요한 세부 정보를 입력하고 Continue를 선택합니다.
5. 설문 조사 질문을 완료하고 단계에 따라 계정 생성 프로세스를 완료합니다.

OAuth 애플리케이션을 등록합니다.

1. monday.com 계정에 로그인합니다. 화면 왼쪽 하단 모서리에 있는 아바타(사진 아이콘)를 클릭합니다.
2. Developer를 선택합니다.
3. 앱 생성을 선택합니다.
4. 이름 및 설명에 관한 필수 필드를 입력합니다.
5. 오른쪽에 있는 'OAuth' 섹션으로 이동하여 범위를 추가하고 'Save Feature'를 선택합니다.
6. 범위 옆의 'Redirect URLS' 탭으로 이동하여 리디렉션 URL을 추가하고 'Save Feature'를 선택합니다.
7. Redirect URLs 탭에서 앱의 URL을 입력합니다. 이 값은 https://{region-code}.console.aws.amazon.com/appflow/oauth여야 합니다. 예를 들어 `us-east-1` 을 사용하는

경우 `https://us-east-1.console.aws.amazon.com/appflow/oauth`를 추가할 수 있습니다.

8. 이제 애플리케이션을 사용할 준비가 되었습니다. 자격 증명은 'Basic Information' 섹션에서 찾을 수 있습니다. 클라이언트 ID와 클라이언트 보안 암호 문자열을 기록해 둡니다. 이 문자열은 AppFlow 커넥터를 사용하여 이 앱과 연결하는 데 사용됩니다.

개인 액세스 토큰 생성:

현재 monday.com은 개인 토큰인 V2 API 토큰만 제공합니다. API 토큰에 액세스하려면 사용자 수준에 따라 두 가지 방법 중 하나를 사용할 수 있습니다. 관리자 사용자는 두 방법을 모두 활용하여 API 토큰을 획득할 수 있습니다. 멤버 사용자는 Developer 탭에서 API 토큰에 액세스할 수 있습니다.

관리자 - monday.com 계정의 관리자 사용자인 경우 다음 단계에 따라 'Admins' 탭에서 API 토큰에 액세스할 수 있습니다.

1. monday.com 계정에 로그인합니다. 화면 왼쪽 하단 모서리에 있는 아바타(사진 아이콘)를 클릭합니다.
2. 이후 표시되는 메뉴에서 'Administration'을 선택합니다(관리자 권한이 있어야 함).
3. 'API' 섹션으로 이동하여 'API V2 토큰'을 생성합니다. 토큰을 복사하고 사용할 수 있습니다.

개발자 - monday.com 계정의 멤버 사용자인 경우 다음 단계를 통해 Developer 탭에서 API 토큰에 액세스할 수 있습니다.

1. monday.com 계정에 로그인합니다. 화면 왼쪽 하단 모서리에 있는 아바타(사진 아이콘)를 클릭합니다.
2. 이후 표시되는 메뉴에서 'Developers'를 선택합니다.
3. 상단 메뉴에서 'Developer' 드롭다운 메뉴를 선택합니다. 'My Access Tokens' 드롭다운 메뉴에서 첫 번째 옵션을 선택합니다.

AWS Glue Studio에서 MongoDB에 연결

AWS Glue에서는 MongoDB를 기본으로 지원합니다. AWS Glue Studio에서는 MongoDB에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스를 제공합니다.

주제

- MongoDB 연결 생성
- MongoDB 소스 노드 생성
- MongoDB 대상 노드 생성
- 고급 옵션

MongoDB 연결 생성

사전 조건:

- MongoDB 인스턴스가 Amazon VPC에 있는 경우, 퍼블릭 인터넷을 통과하는 트래픽 없이 AWS Glue 작업이 MongoDB 인스턴스와 통신할 수 있도록 Amazon VPC를 구성하십시오.

  Amazon VPC에서 작업을 실행하는 동안 AWS Glue가 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 MongoDB 인스턴스와 이 위치 간의 네트워크 트래픽을 허용하도록 Amazon VPC를 구성해야 합니다. 네트워크 레이아웃에 따라 보안 그룹 규칙, 네트워크 ACL, NAT 게이트웨이 및 피어링 연결을 변경해야 할 수도 있습니다.

MongoDB 연결 구성 방법:

1. 대안으로 AWS Secrets Manager에서 MongoDB 보안 인증을 사용하여 보안 암호를 생성할 수 있습니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

   - 키/값 페어를 선택하면 값 *mongodbUser*이 포함된 키 username에 대한 페어를 생성합니다.

     키/값 페어를 선택하면 값 *mongodbPass*가 포함된 키 password에 대한 페어를 생성합니다.
2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때에는 MongoDB 또는 MongoDB Atlas를 선택합니다.
   - MongoDB URL 또는 MongoDB Atlas URL을 선택할 때에는 MongoDB 인스턴스의 호스트 이름을 제공하십시오.

     MongoDB URL은 mongodb://*mongoHost*:*mongoPort*/*mongoDBname* 형식으로 제공됩니다.

MongoDB Atlas URL은 `mongodb+srv://`*`mongoHost`*`:`*`mongoPort`*`/`*`mongoDBname`* 형식으로 제공됩니다.

연결을 위한 기본 데이터베이스를 제공하는 *`mongoDBName`*은 선택 사항입니다.

- Secrets Manager 암호를 생성하기로 선택한 경우 AWS Secrets Manager 보안 인증 정보 유형을 선택합니다.

  그런 다음 AWS 암호에 *`secretName`*을 입력합니다.

- 사용자 이름과 비밀번호를 제공하기로 선택한 경우 *`mongodbUser`*와 *`mongodbPass`*를 제공하십시오.

3. 다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.

   - Amazon VPC에서 AWS에 호스팅된 MongoDB 인스턴스의 경우
     - MongoDB 보안 보안 인증 정보를 정의하는 AWS Glue 연결에 Amazon VPC 연결 정보를 제공해야 합니다. 연결을 만들거나 업데이트할 때 네트워크 옵션에서 VPC, 서브넷 및 보안 그룹을 설정합니다.

AWS Glue MongoDB 연결을 생성한 후에는 AWS Glue 작업을 실행하기 전에 다음 단계를 수행해야 합니다.

- 비주얼 에디터에서 작업을 수행할 때는 AWS Glue 작업에 대한 Amazon VPC 연결 정보를 제공해야 MongoDB에 연결할 수 있습니다. Amazon VPC에서 적합한 위치를 식별하여 MongoDB AWS Glue 연결에 제공하십시오.
- Secrets Manager 암호을 생성하기로 선택한 경우, AWS Glue 작업과 연결된 IAM 역할에 *`secretName`*을 읽을 수 있는 권한을 부여하십시오.

MongoDB 소스 노드 생성

필수 전제 조건

- 이전 섹션 the section called “MongoDB 연결 생성”에서 설명한 AWS Glue MongoDB 연결입니다.
- Secrets Manager 암호를 생성하기로 선택한 경우 연결에 사용되는 암호를 읽을 수 있는 적절한 작업 권한이 있어야 합니다.
- 읽으려는 MongoDB 컬렉션. 컬렉션에 대한 식별 정보가 필요합니다.

MongoDB 컬렉션은 데이터베이스 이름 *mongodbName* 및 컬렉션 이름 *mongodbCollection*으로 식별됩니다.

MongoDB 데이터 소스 추가

데이터 소스 - MongoDB 노드 추가 방법:

1. MongoDB 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 MongoDB 연결 생성을 선택합니다. 자세한 내용은 이전the section called “MongoDB 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. 데이터베이스를 선택합니다. *mongodbName*을 입력합니다.
3. 컬렉션을 선택하십시오. *mongodbCollection*을 입력하십시오.
4. 파티셔너, 파티션 크기(MB), 파티션 키를 선택합니다. 파티션 파라미터에 대한 자세한 내용은 the section called “소스로서의 "connectionType": "mongodb"” 섹션을 참조하십시오.
5. 사용자 지정 MongoDB 속성에서 필요한 경우 파라미터와 값을 입력합니다.

MongoDB 대상 노드 생성

필수 전제 조건

- 이전 섹션 the section called “MongoDB 연결 생성”에서 설명한 대로 AWS Secrets Manager 암호로 구성된 AWS Glue MongoDB 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 쓰고 싶은 MongoDB 테이블, *tableName*입니다.

MongoDB 데이터 대상 추가

데이터 대상 - MongoDB 노드 추가 방법:

1. MongoDB 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 MongoDB 연결 생성을 선택합니다. 자세한 내용은 이전the section called “MongoDB 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.

2. 데이터베이스를 선택합니다. *mongodbName*을 입력합니다.
3. 컬렉션을 선택하십시오. *mongodbCollection*을 입력하십시오.
4. 파티셔너, 파티션 크기(MB), 파티션 키를 선택합니다. 파티션 파라미터에 대한 자세한 내용은 the section called “소스로서의 "connectionType": "mongodb"” 섹션을 참조하십시오.
5. 원하는 경우 쓰기 재시도를 선택합니다.
6. 사용자 지정 MongoDB 속성에서 필요한 경우 파라미터와 값을 입력합니다.

고급 옵션

MongoDB 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “MongoDB 연결”를 참조하세요.

Oracle에 연결 NetSuite

Oracle NetSuite 은 all-in-one 핵심 프로세스를 자동화하고 운영 및 재무 성과에 대한 실시간 가시성을 제공하여 조직이 더 효과적으로 운영할 수 있도록 지원하는 클라우드 비즈니스 관리 솔루션입니다. 회계, 주문 처리, 재고 관리, 생산, 공급망 및 창고 운영을 관리하기 위한 통합된 단일 애플리케이션 제품군을 통해 Oracle NetSuite 은 기업에 데이터에 대한 명확한 가시성과 비즈니스에 대한 더 엄격한 제어를 제공합니다.

주제

- AWS Glue Oracle 지원 NetSuite
- 연결 생성 및 사용 API 작업이 포함된 정책
- Oracle 구성 NetSuite
- Oracle NetSuite 연결 구성
- Oracle NetSuite 엔터티에서 읽기
- Oracle NetSuite 연결 옵션
- Oracle NetSuite 커넥터에 대한 제한 사항 및 참고 사항

AWS Glue Oracle 지원 NetSuite

AWS Glue 는 다음과 NetSuite 같이 Oracle을 지원합니다.

소스로 지원되나요?

예. 작업을 사용하여 AWS Glue ETL Oracle 에서 데이터를 쿼리할 수 있습니다 NetSuite.

대상으로서 지원되나요?

아니요.

지원되는 Oracle NetSuite API 버전

지원되는 Oracle NetSuite API 버전은 다음과 같습니다.

- v1

버전별 엔터티 지원은 지원되는 소스 엔터티를 참조하세요.

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려면 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch

Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS 관리 콘솔을 사용할 때 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

Oracle 구성 NetSuite

AWS Glue 를 사용하여 Oracle 에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 NetSuite합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Oracle NetSuite 계정이 있습니다. 자세한 내용은 Oracle NetSuite 계정 생성 단원을 참조하십시오.
- Oracle NetSuite 계정에 API 액세스할 수 있습니다.
- Oracle NetSuite 개발자 계정에 OAuth 2.0 API 통합을 생성했습니다. 이 통합은 가 계정에 인증된 호출을 할 때 데이터에 안전하게 액세스하는 데 AWS Glue 사용하는 클라이언트 자격 증명을 제공합니다. 자세한 내용은 Oracle NetSuite 클라이언트 앱 및 OAuth 2.0 보안 인증 생성 단원을 참조하십시오.

이러한 요구 사항을 충족하면 Oracle NetSuite 계정에 AWS Glue 연결할 준비가 된 것입니다.

Oracle NetSuite 계정 생성

Oracle NetSuite로 이동하여 무료 제품 둘러보기 를 선택합니다. 필요한 세부 정보를 입력하여 공급업체에 문의할 수 있는 무료 제품 둘러보기를 제공합니다. 계정 조달 프로세스는 다음과 같습니다.

- NetSuite 계정 조달은 공급업체를 통해 이루어지며, 공급업체는 법적 검토가 필요한 양식/견적을 제공합니다.
- Oracle NetSuite 커넥터에 대해 조달할 계정은 Standard Cloud Service 입니다.
- 이 계정은 공급업체에서 생성하며 임시 자격 증명은 공급업체에서 공유합니다. NetSuite <billing@notification.netsuite.com> <system@sent-via.netsuite.com>에서 사용자 이름과 같은 세부 정보와 암호를 설정하는 링크가 포함된 환영 메일을 받게 됩니다.
- 암호 설정 링크를 사용하여 공급업체에서 제공한 사용자 이름의 암호를 설정합니다.

Oracle NetSuite 클라이언트 앱 및 OAuth 2.0 보안 인증 생성

클라이언트 ID와 클라이언트 보안 암호를 가져오려면 Oracle NetSuite 클라이언트 앱을 생성합니다.

1. 고객 로그인 을 통해 NetSuite 계정에 로그인합니다. NetSuite
2. 설정 > 회사 > 기능 활성화를 선택합니다.
3. SuiteCloud 섹션으로 이동하여 SuiteTalk (웹 서비스) 아래의 REST WEB SERVICES 확인란을 선택합니다.
4. 인증 관리에서 OAUTH 2.0 확인란을 선택합니다. 저장을 클릭합니다.
5. 설정 > 통합 > 통합 관리로 이동하여 새로 만들기를 선택하여 OAuth2.0 애플리케이션을 생성합니다.
6. 선택한 이름을 입력하고 를 활성화됨STATE으로 유지합니다.
7. 선택하면 토큰 기반 인증 아래에 표시된 TBA: AUTHORIZATION FLOW 및 TOKENBASED-AUTHENTICATION 확인란의 선택을 취소합니다.
8. OAuth 2.0에서 AUTHORIZATION CODE GRANT 및 PUBLIC CLIENT 확인란을 선택합니다.
9. 인증에서 클라이언트 ID와 클라이언트 보안 암호를 기록해 둡니다.
10. 를 입력합니다REDIRECTURI. 예: https://us-east-1.console.aws.amazon.com/gluestudio/oauth
11. 아래의 REST WEB SERVICES 확인란을 선택합니다SCOPE.
12. 사용자 자격 증명 아래의 USER CREDENTIALS 확인란을 선택합니다. 저장(Save)을 선택합니다.
13. 클라이언트 자격 증명 CONSUMER KEY/CLIENT ID and CONSUMER SECRET/CLIENT SECRET 아래의 에 유의하세요. 이러한 값은 한 번만 표시됩니다.
14. 필요한 경우 사용자/역할 > ADMINISTRATOR 역할 관리 > 새로 만들기로 이동하여 역할을 생성합니다.
15. 사용자 지정 역할을 생성하는 동안 권한 탭에서 다음 엔터티/기능에 대한 전체 액세스를 추가합니다.
    - “예치”, “항목”, “항목 이행”, “일지 항목 작성”, “구매 주문”, “자회사”, “벤더”, “청구서”, “벤더 반환 승인”, “추적 시간”, “고객 결제”, “사용자 지정 레코드 항목”, “사용자 지정 레코드 유형”, “REST웹 서비스”, “OAuth2.0 승인 애플리케이션 관리”, “사용자 지정 엔터티 필드”, “OAuth2.0 액세스 토큰을 사용하여 로그인”.

자세한 내용은 NetSuite Applications Suite 설명서의 OAuth 2.0을 참조하세요.

Oracle NetSuite 연결 구성

Oracle NetSuite에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다. 권한 부여 유형은 AWS Glue에서 Oracle NetSuite와 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Oracle NetSuite 인스턴스 URL을 제외한 OAuth 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱(AWS Glue 관리형 클라이언트 애플리케이션)에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Oracle NetSuite로 리디렉션합니다. 사용자가 Oracle NetSuite에 로그인하고 Salesforce 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Oracle NetSuite에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Oracle NetSuite로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Oracle NetSuite 설명서는 Public apps를 참조하세요.

Oracle NetSuite 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결에 대한 보안 암호를 생성해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Oracle NetSuite를 선택합니다.
   b. Oracle NetSuite 환경을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Oracle NetSuite 엔터티에서 읽기

사전 조건

읽으려는 Oracle NetSuite 객체. 객체 이름(예: `deposit` 또는 `timebill`)이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티:

개체	필터링 가능	정렬 기준 지원	제한 지원	SELECT * 지원	분할 지원
입금	예	아니요	예	예	예
설명 항목	예	아니요	예	예	예
인벤토리 항목	예	아니요	예	예	예
항목 이행	예	아니요	예	예	예

개체	필터링 가능	정렬 기준 지원	제한 지원	SELECT * 지원	분할 지원
항목 그룹	예	아니요	예	예	예
저널 항목	예	아니요	예	예	예
인벤토리 외 구매 항목	예	아니요	예	예	예
인벤토리 외 재판매 항목	예	아니요	예	예	예
인벤토리 외 판매 항목	예	아니요	예	예	예
구매 주문	예	아니요	예	예	예
자회사	예	아니요	예	예	예
공급 업체	예	아니요	예	예	예
공급업체 청구서	예	아니요	예	예	예
공급업체 반환 권한 부여	예	아니요	예	예	예
시간 청구서	예	아니요	예	예	예
고객 결제	예	아니요	예	예	예
이행 요청	예	아니요	예	예	예

예제:

```
netsuiteerp_read = glueContext.create_dynamic_frame.from_options(
    connection_type="netsuiteerp",
    connection_options={
        "connectionName": "connectionName",
```

```
        "ENTITY_NAME": "deposit",
        "API_VERSION": "v1"
    }
)
```

Oracle NetSuite 엔터티 및 필드 세부 정보:

Oracle NetSuite에서는 선택한 엔터티 아래에서 사용 가능한 필드를 동적으로 로드합니다. 필드의 데이터 유형에 따라 다음 필터 연산자를 지원합니다.

필드 데이터 유형	지원되는 필터 연산자
String	LIKE, =, !=
날짜	BETWEEN, =, <, <=, >, >=
DateTime	BETWEEN, <, <=, >, >=
숫자	=, !=, <, <=, >, >=
불	=, !=

쿼리 파티셔닝

필드 기반 분할

Oracle NetSuite 커넥터에는 동적 메타데이터가 있으므로 필드 기반 분할에 대해 지원되는 필드가 동적으로 선택됩니다. 필드 기반 분할은 정수, BigInteger, 날짜 또는 DateTime와 같은 데이터 유형을 사용하는 필드에서 지원됩니다.

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리를 파티셔닝하는 데 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  타임스탬프 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

```
"TIMESTAMP \"1707256978123\""
"TIMESTAMP \"1702600882\""
"TIMESTAMP '2024-02-06T22:00:00:00.000Z'"
"TIMESTAMP '2024-02-06T22:00:00:00Z'"
"TIMESTAMP '2024-02-06'"
```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예제:

```
oracle-netsuite_read = glueContext.create_dynamic_frame.from_options(
    connection_type="oracle-netsuite",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "company",
        "API_VERSION": "v3",
        "PARTITION_FIELD": "hs_object_id"
        "LOWER_BOUND": "50"
        "UPPER_BOUND": "16726619290"
        "NUM_PARTITIONS": "10"
    }
```

레코드 기반 분할

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(NUM_PARTITIONS)을 제공할 수 있습니다. 이 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

레코드 기반 분할에서는 존재하는 총 레코드 수를 Oracle NetSuite API에서 쿼리하고 제공된 NUM_PARTITIONS 수로 나눕니다. 그런 다음, 결과 레코드 수를 각 하위 쿼리에서 동시에 가져옵니다.

- NUM_PARTITIONS: 파티션 수.

예제:

```
netsuiteerp_read = glueContext.create_dynamic_frame.from_options(
    connection_type="netsuiteerp",
    connection_options={
```

```
        "connectionName": "connectionName",
        "ENTITY_NAME": "deposit",
        "API_VERSION": "v1",
        "NUM_PARTITIONS": "3"
    }
```

Oracle NetSuite 연결 옵션

다음은 Oracle 에 대한 연결 옵션입니다 NetSuite.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Oracle NetSuite 엔터티의 이름입니다. 예: 입금.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Oracle NetSuite Rest API 버전입니다. Oracle은 NetSuite 현재 버전 v1만 지원하므로 값은 v1입니다.
- SELECTED_FIELDS(목록<문자열>) - 기본값: 비어 있음(SELECT*). 읽기에 사용됩니다. 선택한 엔터티에 대해 선택할 열의 쉼표로 구분된 목록입니다.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리를 분할하는 데 사용할 필드(필드 기반 분할).
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 필드 기반 파티셔닝에 사용되는 선택한 파티션 필드의 포함 하한 값입니다.
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 필드 기반 파티셔닝에 사용되는 선택한 파티션 필드의 배타적 상한 값입니다.
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수. 필드 기반 파티셔닝과 레코드 기반 파티셔닝 모두에 사용됩니다.
- INSTANCEE_URL(문자열) - 형식이 URL인 유효한 NetSuite 인스턴스입니다 https://{account-id}.suitetalk.api.netsuite.com.

Oracle NetSuite 커넥터에 대한 제한 사항 및 참고 사항

다음은 Oracle NetSuite 커넥터에 대한 제한 사항 또는 참고 사항입니다.

- access_token 및 refresh_token 파라미터의 값은 JSON 웹 토큰(JWT) 형식입니다. 액세스 토큰은 60분 동안 유효하고 refresh_token은 7일 동안 유효합니다.
- 클라이언트 ID 및 클라이언트 보안 암호 생성 중에 “”PUBLIC와 CLIENT함께 “”AUTHORIZATION를 선택하면 CODE GRANT새로 고침 토큰은 3시간 동안만 유효하며 일회용입니다.

- 커넥터를 사용하여 최대 1,000,000개의 레코드를 가져올 수 있습니다.
- 파티션은 각 파티션이 1000의 배수로 레코드를 가져오도록 생성됩니다. 단, 나머지 레코드를 가져올 마지막 파티션은 예외입니다.

AWS Glue Studio에서 OpenSearch Service에 연결

AWS Glue에서는 Amazon OpenSearch Service를 기본으로 지원합니다. AWS Glue Studio에서는 Amazon OpenSearch Service에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스를 제공합니다. 이 기능은 OpenSearch Service 서버리스와 호환되지 않습니다.

AWS Glue Studio에서 Amazon OpenSearch Service용 통합 연결을 생성합니다. 자세한 내용은 고려 사항 단원을 참조하십시오.

주제

- OpenSearch Service 연결 생성
- OpenSearch Service 소스 노드 생성
- OpenSearch Service 대상 노드 생성
- 고급 옵션

OpenSearch Service 연결 생성

사전 조건:

- Amazon OpenSearch Service 설명서의 지침에 따라 읽으려는 도메인 엔드포인트, *aosEndpoint* 및 포트, *aosPort*를 식별하거나 리소스를 생성하십시오. 도메인 생성에 관해 자세한 내용을 알아보려면 Amazon OpenSearch Service 설명서의 Amazon OpenSearch Service 도메인 생성 및 관리를 참조하십시오.

  Amazon OpenSearch Service 도메인 엔드포인트는 기본 형식이 https://search-*domainName*-*unstructuredIdContent*.*region*.es.amazonaws.com입니다. 도메인 엔드포인트에 관해 자세한 내용을 알아보려면 Amazon OpenSearch Service 설명서의 Amazon OpenSearch Service 도메인 생성 및 관리를 참조하십시오.

  도메인의 HTTP 기본 보안 인증 정보, *aosUser*와 *aosPassword*를 확인하거나 생성하십시오.

OpenSearch Service에 대한 연결을 구성하는 방법:

1. AWS Secrets Manager에서 OpenSearch 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

   - 키/값 페어를 선택하면 값 *aosUser*이 포함된 키 USERNAME에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *aosPassword*이 포함된 키 PASSWORD에 대한 페어를 생성합니다.

2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때는 OpenSearch Service를 선택합니다.
   - 도메인 엔드포인트를 선택할 때는 *aosEndpoint*를 제공하십시오.
   - 포트를 선택할 때는 *aosPort*를 제공하십시오.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

OpenSearch Service 소스 노드 생성

필수 전제 조건

- 이전 섹션 the section called “OpenSearch Service 연결 생성”에서 설명한 대로 AWS Secrets Manager 암호로 구성된 AWS Glue OpenSearch Service 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 읽으려는 OpenSearch Service 인덱스, *aosIndex*.

OpenSearch Service 데이터 소스 추가

데이터 소스 - OpenSearch Service 노드를 추가하는 방법:

1. OpenSearch Service 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 OpenSearch Service 연결 생성을 선택합니다. 자세한 내용은 이전the section called “OpenSearch Service 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.

2. 읽고 싶은 인덱스인 인덱스을 입력합니다.

3. 더 구체적인 결과를 제공하기 위해 OpenSearch 쿼리인 Query를 제공할 수도 있습니다. OpenSearch 쿼리 작성에 대한 자세한 내용은 the section called “OpenSearch Service에서 읽기”를 참조하십시오.
4. 사용자 지정 OpenSearch Service 속성에서 필요한 경우 파라미터와 값을 입력합니다.

OpenSearch Service 대상 노드 생성

필수 전제 조건

- 이전 섹션 the section called “OpenSearch Service 연결 생성”에서 설명한 대로 AWS Secrets Manager 암호로 구성된 AWS Glue OpenSearch Service 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 쓰려는 OpenSearch Service 인덱스, `aosIndex`.

OpenSearch Service 데이터 대상 추가

데이터 대상을 추가하는 방법 - OpenSearch Service 노드:

1. OpenSearch Service 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 OpenSearch Service 연결 생성을 선택합니다. 자세한 내용은 이전the section called “OpenSearch Service 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. 읽고 싶은 인덱스인 인덱스을 입력합니다.
3. 사용자 지정 OpenSearch Service 속성에서 필요한 경우 파라미터와 값을 입력합니다.

고급 옵션

OpenSearch Service 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “OpenSearch Service 연결”를 참조하세요.

Okta에 연결

Okta API는 대규모 또는 복잡한 Okta 계정 및 캠페인을 관리하는 데 사용되는 Okta에 대한 프로그래밍 방식 인터페이스입니다. Okta 사용자인 경우 Okta 계정에 AWS Glue를 연결할 수 있습니다. 그런 다

음, Okta를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Okta 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 Okta 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Okta 구성
- Okta 연결 구성
- Okta 엔터티에서 읽기
- Okta 연결 옵션 참조
- Okta 새 계정 및 개발자 앱 생성 단계
- 제한 사항

AWS Glue의 Okta 지원

AWS Glue에서는 다음과 같이 Okta를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Okta에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Okta API 버전

v1.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Okta 구성

AWS Glue를 사용하여 Okta에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- Okta 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Okta 새 계정 및 개발자 앱 생성 단계 섹션을 참조하세요.
- Okta 계정이 API 액세스에 대해 활성화되어 있습니다.
- Okta 계정에 OAuth2 API 통합을 생성했습니다. 이 통합에서는 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하는 데 사용하는 클라이언트 자격 증명을 제공합니다. 자세한 내용은 클라이언트 앱 및 OAuth2.0 보안 인증 생성 단계: Okta 새 계정 및 개발자 앱 생성 단계를 참조하세요.
- OktaApiToken이 있는 Okta 계정이 있습니다. Okta 설명서를 참조하세요.

이러한 요구 사항을 충족하면 Okta 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Okta에서 다른 작업을 수행하지 않아도 됩니다.

Okta 연결 구성

Okta는 두 가지 유형의 인증 메커니즘을 지원합니다.

- OAuth 인증: Okta는 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.
  - 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Okta로 리디렉션합니다. 사용자가 로그인하고 Okta 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
  - 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Okta에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Okta로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
  - 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
  - 자세한 내용은 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성의 공개 Okta 설명서를 참조하세요.
- 사용자 지정 인증:
  - 사용자 지정 인증에 필요한 API 키 생성에 대한 공개 Okta 설명서는 Okta 설명서를 참조하세요.

다음 단계를 따라 Okta 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.

   a. OAuth 인증:

   - 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

   b. 사용자 지정 인증:

   - 고객 관리형 연결된 앱 - 보안 암호는 키 역할을 하는 `OktaApiToken`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.

a. 연결에서 연결 생성을 선택합니다.

b. 데이터 소스를 선택할 때 Okta를 선택합니다.

c. Okta 하위 도메인을 입력합니다.

d. Okta 계정의 Okta 도메인 URL을 선택합니다.

e. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

f. 인증 유형을 선택하여 데이터 소스에 연결합니다.

g. OAuth2 인증 유형의 경우 Okta 앱의 사용자 관리형 클라이언트 애플리케이션 ClientId를 입력합니다.

h. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

i. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

Okta 엔터티에서 읽기

사전 조건

- 읽으려는 Okta 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Applications	예	예	아니요	예	아니요
Devices	예	예	아니요	예	예
Groups	예	예	예	예	예
Users	예	예	예	예	예
User Types	아니요	아니요	아니요	예	아니요

예

```
okta_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Okta",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "applications",
        "API_VERSION": "v1"
    }
```

Okta 엔터티 및 필드 세부 정보

엔터티 목록:

- Application: https://developer.okta.com/docs/api/openapi/okta-management/management/tag/Application/
- Device: https://developer.okta.com/docs/api/openapi/okta-management/management/tag/Device/
- Group: https://developer.okta.com/docs/api/openapi/okta-management/management/tag/Group/
- User: https://developer.okta.com/docs/api/openapi/okta-management/management/tag/User/
- User Type: https://developer.okta.com/docs/api/openapi/okta-management/management/tag/UserType/

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-02-06".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예

```
okta_read = glueContext.create_dynamic_frame.from_options(
    connection_type="okta",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "lastUpdated",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "lastMembershipUpdated"
        "LOWER_BOUND": "2022-08-10T10:28:46.000Z"
        "UPPER_BOUND": "2024-08-10T10:28:46.000Z"
        "NUM_PARTITIONS": "10"
    }
```

Okta 연결 옵션 참조

다음은 Okta의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Okta에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용하려는 Okta Rest API 버전입니다. 예: v1.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Okta 새 계정 및 개발자 앱 생성 단계

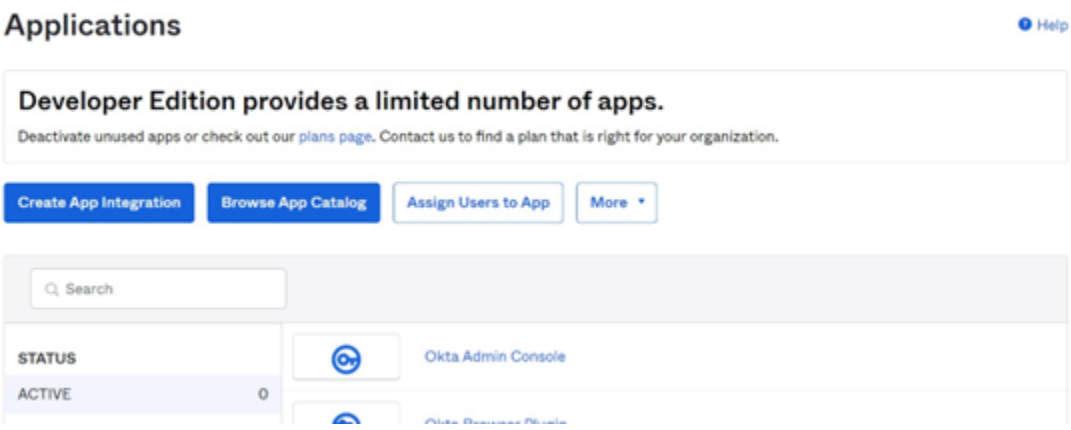
Okta API에 액세스할 수 있도록 Okta에서 개발자 계정을 생성합니다. 무료 Okta 개발자 계정을 사용하면 Okt API에 액세스하는 데 필요한 대부분의 주요 개발자 기능에 액세스할 수 있습니다.

Okta에서 개발자 계정 생성

1. https://developer.okta.com/signup/ 페이지로 이동합니다.
2. 계정 정보 이메일, 이름, 성 및 국가/지역을 입력합니다. I'm not a robot을 선택하고 Signup을 선택합니다.
3. 확인 메일이 등록된 메일 ID로 전송됩니다. Okta 개발자 계정을 활성화하기 위한 링크가 이메일로 전송됩니다. 활성화를 선택합니다.
4. 암호 재설정 페이지로 리디렉션됩니다. 새 암호를 두 번 입력하고 Reset password를 선택합니다.
5. Okta 개발자 계정 대시보드로 리디렉션됩니다.

클라이언트 앱 및 OAuth 2.0 자격 증명 생성

1. 개발자 대시보드에서 앱 통합 생성을 선택합니다.



2. Create a new app Integration 창이 표시되고 다양한 로그인 방법이 표시됩니다. OIDC –OpenID Connect를 선택합니다.

3. Application type 섹션으로 스크롤합니다. Web Application으로 선택하고 Next를 선택합니다.
4. 'New Web App Integration' 화면에서 다음 정보를 입력합니다.
   - App integration name - 앱의 이름을 입력합니다.
   - Grant type - 목록에서 Authorization Code 및 Refresh Token을 선택합니다.
   - Sign-in redirect URIs - Add URI를 선택하고 `https://{regioncode}.console.aws.amazon.com/appflow/oauth`를 추가합니다. 예를 들어 `us-west-2 (Oregon)`을 사용하는 경우 `https://us-east-1.console.aws.amazon.com/appflow/oauth`를 추가할 수 있습니다.
   - Controlled Access - 필요에 따라 사용자 그룹에 앱을 할당하고 Save를 선택합니다.
5. 클라이언트 ID와 클라이언트 보안 암호가 생성됩니다.

제한 사항

다음은 Okta 커넥터의 제한 사항입니다.

- 'Applications' 엔터티의 경우 1개의 필터만 적용할 수 있습니다. 2개 이상의 필터가 적용되는 경우 400 Bad Request가 오류 요약 - 'Invalid Search criteria'와 함께 반환됩니다.
- 순서 기준은 검색 쿼리에서만 지원될 수 있습니다. 예제: `http://dev-15940405.okta.com/api/v1/groups?search=type e.q. "OKTA_GROUP"&sortBy=lastUpdated&sortOrder=asc`

PayPal에 연결

PayPal은 고객과 온라인 공급업체 간의 이체와 같이 당사자 간의 온라인 자금 이체를 용이하게 하는 결제 시스템입니다. PayPal 사용자인 경우 계정에 지급인, 날짜 및 상태와 같은 거래에 관한 데이터가 포함되어 있습니다. AWS Glue를 사용하여 PayPal에서 특정 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송할 수 있습니다.

주제

- AWS Glue의 PayPal 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- PayPal 구성
- PayPal 연결 구성
- PayPal 엔터티에서 읽기

- PayPal 연결 옵션 수정
- PayPal 커넥터의 제한 사항 및 참고 사항

AWS Glue의 PayPal 지원

AWS Glue에서는 다음과 같이 PayPal를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 PayPal에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 PayPal API 버전

다음 PayPal API 버전이 지원됩니다.

- v1

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

PayPal 구성

AWS Glue를 사용하여 PayPal에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- 클라이언트 자격 증명을 사용하는 PayPal 계정이 있습니다.
- PayPal 계정에 유효한 라이선스가 있는 API 액세스 권한이 있습니다.

이러한 요구 사항을 충족하면 PayPal 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 PayPal에서 다른 작업을 수행하지 않아도 됩니다.

PayPal 연결 구성

PayPal은 OAuth2에 대한 CLIENT CREDENTIALS 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 클라이언트가 사용자의 컨텍스트 외부에서 액세스 토큰을 얻는 데 사용하므로 2각 OAuth 2.0으로 간주됩니다. AWS Glue는 클라이언트 ID와 클라이언트 암호를 사용하여 사용자가 정의한 사용자 지정 서비스에서 제공하는 PayPal API를 인증할 수 있습니다.
- 각 사용자 지정 서비스는 API 전용 사용자가 소유하며, API 전용 사용자는 서비스에 특정 작업을 수행하도록 권한을 부여하는 역할 및 권한 집합을 가집니다. 액세스 토큰은 단일 사용자 지정 서비스와 연결됩니다.

- 이 권한 부여 유형으로 인해 액세스 토큰이 수명이 짧아지고 `/v2/oauth2/token` 엔드포인트를 다시 직접적으로 호출하여 갱신할 수 있습니다.
- 클라이언트 자격 증명이 있는 OAuth 2.0에 대한 퍼블릭 PayPal 설명서는 Authentication을 참조하세요.

PayPal 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 PayPal을 선택합니다.
   b. 연결하려는 PayPal 인스턴스의 `INSTANCE_URL`을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

OAuth 2.0 자격 증명 가져오기

Rest API를 호출하려면 클라이언트 ID와 클라이언트 보안 암호를 액세스 토큰으로 교환해야 합니다. 자세한 내용은 PayPal REST API 시작하기를 참조하세요.

PayPal 엔터티에서 읽기

사전 조건

읽으려는 PayPal 객체입니다. 객체 이름, `transaction`이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
transaction	예	예	아니요	예	예

예시:

```
paypal_read = glueContext.create_dynamic_frame.from_options(
    connection_type="paypal",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "transaction",
        "API_VERSION": "v1",
        "INSTANCE_URL": "https://api-m.paypal.com"
    }
```

PayPal 엔터티 및 필드 세부 정보:

정적 메타데이터를 포함하는 엔터티:

엔터티	필드	데이터 유형	지원되는 연산자
transaction	transaction_initiation_date	DateTime	Between

엔터티	필드	데이터 유형	지원되는 연산자
	last_refreshed_datetime	String	N/A
	payment_instrument_type	String	=
	balance_affecting_records_only	String	=
	store_id	String	=
	terminal_id	String	=
	transaction_currency	String	=
	transaction_id	String	N/A
	transaction_status	String	N/A
	transaction_type	String	N/A
	transaction_info	Struct	N/A
	payer_info	Struct	N/A
	shipping_info	Struct	N/A
	cart_info	Struct	N/A
	store_info	Struct	N/A
	auction_info	Struct	N/A
	incentive_info	Struct	N/A

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2024-07-01T00:00:00.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티별 분할에는 다음 필드가 지원됩니다.

엔터티 이름	분할 필드	데이터 유형
transaction	transaction_initiation_date	DateTime

예시:

```
paypal_read = glueContext.create_dynamic_frame.from_options(
    connection_type="paypal",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "transaction",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "transaction_initiation_date"
        "LOWER_BOUND": "2024-07-01T00:00:00.000Z"
        "UPPER_BOUND": "2024-07-02T00:00:00.000Z"
        "NUM_PARTITIONS": "10"
    }
```

PayPal 연결 옵션 수정

다음은 PayPal의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. PayPal에서의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 PayPal Rest API 버전입니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

PayPal 커넥터의 제한 사항 및 참고 사항

다음은 PayPal 커넥터의 제한 사항입니다.

- PayPal 거래 설명서에 따르면 실행된 거래가 거래 내역 직접 호출에 표시되는 데 최대 3시간이 걸린다고 나와 있습니다. 그러나 last_refreshed_datetime에 따라 시간이 더 걸리는 것으로 관찰되었습니다. 여기서 last_refreshed_datetime은 API에서 데이터를 사용할 수 있을 때까지의 시간입니다.
- last_refreshed_datetime이 요청된 end_date보다 작으면 end_date는 해당 시점까지의 데이터만 있으므로 last_refreshed_datetime과 같습니다.
- transaction_initiation_date 필드는 transaction 엔터티에 제공할 필수 필터이며 이 필드에 지원되는 최대 날짜 범위는 31일입니다.
- transaction_initiation_date 필드 이외의 필터(쿼리 파라미터)를 사용하여 transaction 엔터티 API 요청을 직접적으로 호출하면 ending_balance 필드 값을 응답에서 가져오지 않을 것으로 예상됩니다.

Pendo에 연결

Pendo는 사용자 상호 작용 데이터를 저장할 수 있는 풍부한 데이터 스토어를 제공합니다. 고객은 이 데이터를 AWS로 전송하여 다른 제품 데이터와 결합하고, 추가 분석 및 대시보딩을 수행하고, 원하는 경우 알림을 설정할 수 있습니다.

주제

- AWS Glue의 Pendo 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Pendo 구성
- Pendo 연결 구성
- Pendo 엔터티에서 읽기
- Pendo 연결 옵션
- 제한 사항

AWS Glue의 Pendo 지원

AWS Glue에서는 다음과 같이 Pendo를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Pendo에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Pendo API 버전

v1

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
```

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management 콘솔을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Pendo 구성

AWS Glue를 사용하여 Pendo에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- `write access`가 활성화된 `apiKey`가 있는 Pendo 계정이 있습니다.
- Pendo 계정에 유효한 라이선스가 있는 API 액세스 권한이 있습니다.

이러한 요구 사항을 충족하면 Pendo 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Pendo에서 다른 작업을 수행하지 않아도 됩니다.

Pendo 연결 구성

Pendo는 사용자 지정 인증을 지원합니다.

사용자 지정 권한 부여에 필요한 API 키 생성에 대한 공개 Pendo 설명서는 Authentication – Pendo REST API Documentation을 참조하세요.

다음 단계를 따라 Pendo 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `apiKey`와 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

   > **Note**
   >
   > AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 데이터 소스를 선택할 때 Pendo를 선택합니다.
   b. 연결하려는 Pendo 인스턴스의 `instanceUrl` 항목을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
    "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "secretsmanager:DescribeSecret",
                    "secretsmanager:GetSecretValue",
                    "secretsmanager:PutSecretValue",
                    "ec2:CreateNetworkInterface",
                    "ec2:DescribeNetworkInterface",
                    "ec2:DeleteNetworkInterface",
                ],
                "Resource": "*"
            }
        ]
    }
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 입력합니다.

Pendo 엔터티에서 읽기

사전 조건

읽으려는 Pendo 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

- Feature
- Guide
- Page
- Report
- Report Data
- Visitor
- Account
- Event
- Feature Event
- Guide Event
- Page Event
- Poll Event
- Track Event

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Feature	아니요	아니요	아니요	예	아니요
Guide	아니요	아니요	아니요	예	아니요
Page	아니요	아니요	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Report	아니요	아니요	아니요	예	아니요
Report Data	아니요	아니요	아니요	예	아니요
Visitor (Aggregation API)	예	아니요	예	예	아니요
Account (Aggregation API)	예	아니요	예	예	아니요
Event (Aggregation API)	예	아니요	예	예	아니요
Feature Event (Aggregation API)	예	아니요	예	예	예
Guide Event (Aggregation API)	예	아니요	예	예	예
Account (Aggregation API)	예	아니요	예	예	예
Page Event (Aggregation API)	예	아니요	예	예	예
Poll Event (Aggregation API)	예	아니요	예	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Track Event (Aggregation API)	예	아니요	예	예	예

예

```
Pendo_read = glueContext.create_dynamic_frame.from_options(
    connection_type="glue.spark.Pendo",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "feature",
        "API_VERSION": "v1",
        "INSTANCE_URL": "instanceUrl"
    }
```

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  DateTime 필드의 경우 ISO 형식의 값이 허용됩니다.

  유효한 값의 예제:

  ```
  "2024-07-01T00:00:00.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

다음 표에서는 엔터티 분할 필드 지원 세부 정보를 설명합니다.

엔터티 이름
Event
Feature Event
Guide Event
Page Event
Poll Event
Track Event

예시:

```
pendo_read = glueContext.create_dynamic_frame.from_options(
    connection_type="glue.spark.pendo",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "event",
        "API_VERSION": "v1",
        "INSTANCE_URL": "instanceUrl"
        "NUM_PARTITIONS": "10",
        "PARTITION_FIELD": "appId"
        "LOWER_BOUND": "4656"
        "UPPER_BOUND": "7788"
    }
```

Pendo 연결 옵션

다음은 Pendo에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Pendo에서의 객체 이름.
- INSTANCE_URL(문자열) - (필수) 다음 값이 허용되는 유효한 Pendo 인스턴스 URL.
  - 기본값
  - 유럽
  - US1

- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Pendo Engage Rest API 버전. 예: 3.0.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

제한 사항

다음은 Pendo 커넥터에 대한 제한 사항입니다.

- 페이지 매김은 Pendo에서 지원되지 않습니다.
- 필터링은 Aggregate API 객체(Account, Event, Feature Event, Guide Events, Page Event, Poll Event, Track Event 및 Visitor)에서만 지원됩니다.
- DateTimeRange는 Aggregate API 객체(Event, Feature Event, Guide Events, Page Event, Poll Event, Track Event)에 대한 필수 필터 파라미터입니다.
- dayRange 기간은 시간대의 기간 시작 시점으로 버림됩니다. 예를 들어 제공된 필터가 2023-01-12T07:55:27.065Z인 경우 이 기간은 기간의 시작 시점, 즉 2023-01-12T00:00:00Z로 버림됩니다.

Pipedrive에 연결

Pipedrive는 중소기업이 리드를 관리하고, 영업 활동을 추적하고, 더 많은 거래를 성사할 수 있도록 설계된 영업 파이프라인 CRM입니다. Pipedrive를 통해 중소기업의 영업 팀은 프로세스를 간소화하고 하나의 통합 CRM 영업 도구로 영업 데이터를 통합할 수 있습니다. Pipedrive 사용자인 경우 Pipedrive 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, Pipedrive를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Pipedrive 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 Pipedrive 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Pipedrive 구성
- Pipedrive 연결 구성
- Pipedrive 엔터티에서 읽기
- Pipedrive 연결 옵션 참조
- 제한 사항

AWS Glue의 Pipedrive 지원

AWS Glue는 다음과 같이 Pipedrive를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Pipedrive에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Pipedrive API 버전

v1.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
```

```
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Pipedrive 구성

AWS Glue를 사용하여 Pipedrive에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- Pipedrive 계정이 있습니다.
- Pipedrive 계정이 API 액세스에 대해 활성화되어 있습니다.

이러한 요구 사항을 충족하면 Pipedrive 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Pipedrive에서 다른 작업을 수행하지 않아도 됩니다.

Pipedrive 연결 구성

Pipedrive에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Pipedrive instanceurl을 제외한 OAuth 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Pipedrive

로 리디렉션합니다. 사용자가 로그인하고 Pipedrive 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Pipedrive에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택해야 합니다. 이 시나리오에서는 여전히 Pipedrive로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 1시간 동안 활성 상태가 되고 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 자세한 내용은 AUTHORIZATION_CODE OAuth 흐름을 위한 연결된 앱 생성 관련 설명서를 참조하세요.

다음 단계를 따라 Pipedrive 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. 고객 관리형 연결된 앱 - 보안 암호는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 데이터 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Pipedrive를 선택합니다.
   c. Pipedrive instanceURL을 입력합니다.
   d. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

e. 연결하려는 Pipedrive의 사용자 관리형 클라이언트 애플리케이션 ClientId를 입력합니다.

f. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

g. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다. Next(다음)를 선택합니다.
4. connectionName을 입력하고 다음을 선택합니다.
5. 다음 페이지에서 연결 생성을 선택합니다. Pipedrive에 로그인하라는 메시지가 표시됩니다. 사용자 이름과 암호를 입력하고 로그인을 선택합니다.
6. 로그인하면 앱으로 계속을 선택합니다. 이제 연결을 사용할 준비가 되었습니다.
7. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

Pipedrive 엔터티에서 읽기

사전 조건

- 읽으려는 Pipedrive 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Activities	예	예	아니요	예	예
Activity Type	아니요	아니요	아니요	예	아니요
Call Logs	아니요	아니요	아니요	예	아니요
Currencies	예	예	아니요	예	아니요
Deals	예	예	예	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Leads	예	예	예	예	아니요
Lead Sources	아니요	예	아니요	예	아니요
Lead Labels	아니요	아니요	아니요	아니요	아니요
Notes	예	예	예	예	예
Organization	예	예	아니요	예	예
Permission Sets	예	아니요	아니요	예	아니요
Persons	예	예	예	예	예
Pipelines	아니요	예	아니요	예	아니요
Products	예	예	아니요	예	예
Roles	아니요	예	아니요	예	아니요
Stages	예	예	아니요	예	아니요
Users	아니요	아니요	아니요	예	아니요

예

```
pipedrive_read= glueContext.create_dynamic_frame.from_options(
    connection_type="PIPEDRIVE",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "activites",
        "API_VERSION": "v1"
    }
```

Pipedrive 엔터티 및 필드 세부 정보

엔터티 목록:

- Activities: https://developers.pipedrive.com/docs/api/v1/Activities
- Activity Type: https://developers.pipedrive.com/docs/api/v1/ActivityTypes
- Call Logs: https://developers.pipedrive.com/docs/api/v1/CallLogs
- Currencies: https://developers.pipedrive.com/docs/api/v1/Currencies
- Deals: https://developers.pipedrive.com/docs/api/v1/Deals
- Leads: https://developers.pipedrive.com/docs/api/v1/Leads
- Lead Sources: https://developers.pipedrive.com/docs/api/v1/LeadSources
- Lead Labels: https://developers.pipedrive.com/docs/api/v1/LeadLabels
- Notes: https://developers.pipedrive.com/docs/api/v1/Notes
- Organizations: https://developers.pipedrive.com/docs/api/v1/Organizations
- Permission Sets: https://developers.pipedrive.com/docs/api/v1/PermissionSets
- Persons: https://developers.pipedrive.com/docs/api/v1/Persons
- Pipelines: https://developers.pipedrive.com/docs/api/v1/Pipelines
- Products: https://developers.pipedrive.com/docs/api/v1/Products
- Roles: https://developers.pipedrive.com/docs/api/v1/Roles
- Stages: https://developers.pipedrive.com/docs/api/v1/Stages
- Users: https://developers.pipedrive.com/docs/api/v1/Users

개체	데이터 형식	지원되는 연산자
Activities, Deals, Notes, Organization, Persons 및 Products.	날짜	'='
	Integer	'='
	String	'='
	부울	'='

분할 쿼리

Pipedrive에서는 Activities 엔터티의 필드 하나(due_date)만 필드 기반 분할을 지원합니다. 이 필드는 Date 필드입니다.

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-02-06".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예

```
pipedrive_read = glueContext.create_dynamic_frame.from_options(
    connection_type="PIPEDRIVE",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "activites",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "due_date"
        "LOWER_BOUND": "2023-09-07T02:03:00.000Z"
        "UPPER_BOUND": "2024-05-07T02:03:00.000Z"
        "NUM_PARTITIONS": "10"
    }
```

Pipedrive 연결 옵션 참조

다음은 Pipedrive의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Pipedrive에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용하려는 Pipedrive Rest API 버전. 예: v1.
- INSTANCE_URL(문자열) - (필수) 사용자가 작업을 실행하려는 인스턴스의 URL. 예: v1.

- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

제한 사항

다음은 Pipedrive 커넥터의 제한 사항입니다.

- Pipedrive는 하나의 엔터티(Activities)에 대해서만 필드 기반 분할을 지원합니다.
- Pipedrive는 Activities, Deals, Notes, Persons, Organizations 및 Products 엔터티에 대한 레코드 기반 분할을 지원합니다.
- Deals 엔터티에서 잘못된 값 필터 값이 사용되는 경우 필터로서의 상태 필드는 모든 레코드를 반환합니다.
- Deals 엔터티에서 여러 필드의 순서 정렬은 지원되지 않습니다.
- 성능 데이터를 얻기 위해 로컬 AWS 계정을 활용합니다. 하지만 로컬에서 액세스 토큰 새로 고침 제한으로 인해 1GB의 데이터를 처리하는 AWS Glue 작업이 실패합니다. 그에 따라 179MB의 데이터로 성능 테스트를 최적화했으며, 위의 결과는 이 최적화를 기반으로 합니다. 그럼에도 불구하고 파티션 수의 증가에 따라 SaaS 엔드포인트는 단일 파티션에 비해 더 많은 시간이 걸리는 것을 관측했습니다. 이 동작과 관련하여 Pipedrive 지원 팀과 논의했으며, Pipedrive가 요청을 자동으로 스로틀링하고 응답을 지연시키고 있음을 알렸습니다. 따라서 대규모 데이터 세트로 AWS Glue 작업을 실행하거나 동일한 API 엔드포인트를 여러 번 호출할 때 Pipedrive API의 구현으로 인해 시간 초과 문제가 발생할 수 있습니다. 하지만 파티션 수의 증가에 따른 커넥터 및 심 응답 시간이 예상대로 감소하고 있습니다.

Productboard에 연결

Productboard는 제품 팀이 적합한 제품을 더욱 빠르게 출시할 수 있도록 지원하는 제품 관리 시스템입니다. Zendesk, UiPath, Microsoft와 같은 3,000곳 이상의 최신 제품 중심 기업이 Productboard를 사용

하여 사용자에게 무엇이 실제로 필요한지 이해하고, 앞으로 구축해야 할 사항의 우선순위를 정하고, 로드맵을 중심으로 모든 의견을 모읍니다.

주제

- AWS Glue의 Productboard 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Productboard 구성
- Productboard 연결 구성
- Productboard 엔터티에서 읽기
- Productboard 연결 옵션
- Productboard 계정 생성
- 제한 사항

AWS Glue의 Productboard 지원

AWS Glue는 다음과 같이 Productboard를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Productboard에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Productboard API 버전

v1

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management 콘솔을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Productboard 구성

AWS Glue를 사용하여 Productboard에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호를 사용하는 Productboard 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Productboard 계정 생성을 참조하세요.
- AWS Glue에 대한 서비스 액세스 권한으로 AWS 계정을 생성해야 합니다.
- Productboard 계정의 인증 세부 정보가 있습니다(사용자 지정 인증 사용 시 JWT 토큰, 또는 OAuth2.0 사용 시 클라이언트 ID 및 보안 암호).
- 사용자가 `OAuth2.0`을 사용하려는 경우 Productboard에 애플리케이션을 등록하고 How to integrate with Productboard via OAuth2 - developer documentation의 지침에 따라 애플리케이션을 설정합니다.

이러한 요구 사항을 충족하면 Productboard 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Productboard에서 다른 작업을 수행하지 않아도 됩니다.

Productboard 연결 구성

Productboard는 사용자 지정 인증 및 `OAuth2.0`을 지원합니다. `OAuth2.0` Productboard는 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' `OAuth`로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Productboard Client ID 및 Client Secret을 제외한 `OAuth` 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Productboard로 리디렉션합니다. 사용자가 로그인하고 Productboard 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Productboard에서 자체 연결된 앱을 생성하고 자체 Client ID 및 Client Secret을 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Productboard로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- `AUTHORIZATION_CODE OAuth` 흐름에 대해 연결된 앱을 생성하는 방법에 대한 공개 Productboard 설명서는 How to integrate with Productboard via OAuth2 - developer documentation을 참조하세요.

다음 단계를 따라 Productboard 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - `OAuth` 인증 - 고객 관리형 연결된 앱: 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   - `Custom auth` - 고객 관리형 연결된 앱: 보안 암호는 키 역할을 하는 `access_token`과 함께 연결된 앱 `JWT token`을 포함해야 합니다.

   > **Note**
   >
   > AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 데이터 소스를 선택할 때 Productboard를 선택합니다.
   b. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   c. 인증 유형을 선택하여 데이터 소스에 연결합니다.
      - OAuth 인증 - Productboard 앱의 `Token URL` 및 `User Managed Client Application ClientId`를 입력합니다.
   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 입력합니다.

Productboard 엔터티에서 읽기

사전 조건

읽으려는 Productboard 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

- Abuse-reports
- 자동화
- 캠페인
- Click-details
- Lists
- 회원
- Open-details
- Segments
- Stores
- Unsubscribed

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Features	예	예	아니요	예	예
Components	아니요	예	아니요	예	아니요
Products	아니요	예	아니요	예	아니요
Feature Statuses	아니요	예	아니요	예	예
Custom Field Definitions	아니요	예	아니요	예	아니요
Custom Field Values	예	예	아니요	예	아니요

예

```
Productboard_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Productboard",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "feature",
```

```
        "API_VERSION": "1"
    }
```

Productboard 엔터티 및 필드 세부 정보

- Features
- Components
- Feature statuses
- Products
- Custom fields definitions
- Custom fields values

Productboard 연결 옵션

다음은 Productboard에 대한 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기/쓰기에 사용됩니다. Productboard에 있는 객체의 이름.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Productboard Engage Rest API 버전. 예: 3.0.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Productboard 계정 생성

1. Productboard 가입 페이지로 이동하여 이메일 ID와 암호를 입력한 다음 Log me in을 선택합니다.
2. Account Name 필드에 Productboard 계정의 이름을 입력한 다음 I agree to the Privacy Policy check 확인란을 선택합니다.
3. Now create your workspace 페이지의 Workspace URL 필드에 새 워크스페이스의 URL을 입력합니다. 이후 Continue를 선택하여 다음 페이지로 이동하고 나머지 세부 정보를 입력합니다.

그러면 평가판 계정이 생성됩니다. 평가판 계정은 15일 동안 무료입니다. 평가판 기간이 만료되면 유료 플랜을 구매할 수 있습니다. 이메일 주소, 암호 및 워크스페이스 URL을 기록해 둡니다. 나중에 계정에 액세스하려면 이 정보가 필요합니다.

`OAuth2.0` 애플리케이션 등록

1. Productboard login 페이지로 이동하여 이메일 ID와 암호를 입력하고 Log in을 선택합니다.
2. 오른쪽 상단 모서리에서 User 아이콘을 선택한 다음 드롭다운 메뉴에서 Account and billing을 선택합니다.
3. Extras를 선택하고 드롭다운 메뉴에서 Registered apps를 선택합니다.
4. Register An App을 찾아 선택합니다.
5. 다음 세부 정보를 입력합니다.
   - App name – 앱의 이름.
   - Company / Organization – 회사 또는 조직의 이름.
   - App website – 앱의 웹사이트.
   - Redirect URI - 리디렉션 URI 패턴은 로그인 흐름이 완료될 때 Productboard가 리디렉션(요청된 경우)할 수 있는 URI 경로(또는 쉼표로 구분된 경로 목록)입니다. 예제: `https://ap-southeast-2\\.console\\.aws\\.amazon\\.com`
6. 생성(Create)을 선택합니다.
7. Client ID와 Client Secret이 표시됩니다. 복사하여 안전한 위치에 저장합니다. 그런 다음 완료를 선택합니다.

> **Note**
>
> Client ID 및 Client Secret 문자열은 AppFlow 또는 AWS Glue를 사용할 때 이 커넥터와의 연결을 설정하는 데 사용되는 자격 증명입니다.

CustomAuth 자격 증명 검색

1. Productboard 로그인 페이지로 이동하여 이메일 ID와 암호를 입력하고 Log me in을 선택합니다.

   홈 페이지로 리디렉션됩니다.

2. 홈 페이지에서 Workspace Settings > Integrations > Public APIs > Access Token으로 이동합니다.

> **Note**
>
> Public APIs 섹션이 표시되지 않으면 계정이 Essentials 플랜을 사용 중인 것일 수 있습니다. API 토큰에 액세스하려면 Pro 플랜 이상이 필요합니다. 플랜별 기능 및 이름은 변경될 수 있습니다. 패키지에 대한 자세한 내용은 Productboard pricing을 참조하세요.

3. +를 선택하여 새 토큰을 생성하고, 나중에 참조할 수 있도록 안전하게 저장해야 합니다.

`OAuth2.0` 자격 증명 생성

Productboard 커넥터를 사용한 `OAuth2.0` 인증을 활용하려면 Productboard 플랫폼에 애플리케이션을 등록하고 `Client ID` 및 `Client Secret`을 생성해야 합니다

1. Productboard 로그인 페이지로 이동하여 이메일 ID와 암호를 입력하고 Log me in을 선택합니다.
2. Productboard 계정에 새 OAuth2 애플리케이션을 등록하려면 Producboard 페이지로 이동합니다.
3. 필요한 필드를 입력하고 액세스하려는 각 엔터티에 필요한 범위를 선택합니다.

> **Note**
>
> 지원되는 6개의 엔터티에 필요한 다음 네 가지 범위를 선택했습니다.

4. 리디렉션 URL은 `https://ap-southeast-2\\.console\\.aws\\.amazon\\.com` 형식이어야 합니다.

> **Note**
>
> Appflow 리디렉션 URL은 변경될 수 있습니다. 사용 가능한 경우 AWS Glue 플랫폼의 리디렉션 URL을 업데이트하세요.

5. Client ID와 Client Secret이 표시됩니다. 복사하여 안전한 위치에 저장합니다.
6. How to Integrate with Productboard via OAuth2 developer 설명서의 단계에 따라 `OAuth2`를 설정하고 확인할 수 있습니다.

제한 사항

다음은 Productboard 커넥터의 제한 사항입니다.

- Productboard는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.

QuickBooks에 연결

QuickBooks는 중소기업을 위한 최고의 회계 애플리케이션입니다. QuickBooks 회계 애플리케이션은 1980년대 Intuit의 첫 번째 제품 중 하나로, 원래 데스크톱 소프트웨어였기 때문에 1980년대로 거슬러 올라갑니다. 현재 QuickBooks는 여러 회계 및 비즈니스 재무 애플리케이션을 설치형 소프트웨어와 클라우드 기반 SaaS 소프트웨어로 제공합니다. QuickBooks 사용자는 QuickBooks 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, QuickBooks를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 QuickBooks 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 QuickBooks 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- QuickBooks 구성
- QuickBooks 연결 구성
- QuickBooks 엔터티에서 읽기
- QuickBooks 연결 옵션
- QuickBooks 커넥터의 제한 사항 및 참고 사항

AWS Glue의 QuickBooks 지원

AWS Glue에서는 다음과 같이 QuickBooks를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 QuickBooks에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 QuickBooks API 버전

다음 QuickBooks API 버전이 지원됩니다.

- v3

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

QuickBooks 구성

AWS Glue를 사용하여 QuickBooks에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- QuickBooks 계정이 있습니다.
- QuickBooks 계정이 API 액세스에 대해 활성화되어 있습니다.

자세한 내용은 QuickBooks 설명서에서 다음 주제를 참조하세요.

- Intuit 계정 생성
- 앱 생성 및 개발 시작

이러한 요구 사항을 충족하면 QuickBooks 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 QuickBooks에서 다른 작업을 수행하지 않아도 됩니다.

QuickBooks 연결 구성

QuickBooks에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다. 권한 부여 유형은 AWS Glue에서 QuickBooks와 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.
- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 QuickBooks에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 QuickBooks로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 QuickBooks 설명서는 Set up OAuth 2.0을 참조하세요.

QuickBooks 연결을 구성하는 방법:

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 QuickBooks를 선택합니다.
   b. 연결하려는 QuickBooks 인스턴스의 인스턴스 URL과 회사 ID를 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

QuickBooks 엔터티에서 읽기

사전 조건

읽으려는 QuickBooks 객체입니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Account	예	예	예	예	예
Bill	예	예	예	예	예
Company Info	아니요	아니요	아니요	예	아니요
Customer	예	예	예	예	예
Employee	예	예	예	예	예
Estimate	예	예	예	예	예
Invoice	예	예	예	예	예
Item	예	예	예	예	예
Payment	예	예	예	예	예
Preferences	아니요	아니요	아니요	예	아니요
Profit and Loss	예	아니요	아니요	예	아니요
Tax Agency	예	예	예	예	예
Vendors	예	예	예	예	예

예시:

```
QuickBooks_read = glueContext.create_dynamic_frame.from_options(
    connection_type="quickbooks",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v3"
    }
```

QuickBooks 엔터티 및 필드 세부 정보:

엔터티 및 필드 세부 정보에 대한 자세한 내용은 다음을 참조하세요.

- Account
- Bill
- CompanyInfo
- Customer
- Employee
- Estimate
- Invoice
- Item
- Payment
- Preferences
- ProfitAndLoss
- TaxAgency
- Vendor

분할 쿼리

필드 기반 분할:

QuickBooks에서 Integer 및 DateTime 데이터 유형 필드는 필드 기반 분할을 지원합니다.

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

```
"2024-05-07T02:03:00.00Z"
```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예시:

```
QuickBooks_read = glueContext.create_dynamic_frame.from_options(
    connection_type="quickbooks",
    connection_options={
        "connectionName": "connectionName",
        "REALMID": "12345678690123456789",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v3",
        "PARTITION_FIELD": "MetaData_CreateTime"
        "LOWER_BOUND": "2023-09-07T02:03:00.000Z"
        "UPPER_BOUND": "2024-05-07T02:03:00.000Z"
        "NUM_PARTITIONS": "10"
    }
```

레코드 기반 분할:

원본 쿼리가 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 분할됩니다.

- NUM_PARTITIONS: 파티션 수.

예시:

```
QuickBooks_read = glueContext.create_dynamic_frame.from_options(
    connection_type="quickbooks",
    connection_options={
        "connectionName": "connectionName",
        "REALMID": "1234567890123456789",
        "ENTITY_NAME": "Bill",
        "API_VERSION": "v3",
        "NUM_PARTITIONS": "10"
    }
```

QuickBooks 연결 옵션

다음은 QuickBooks의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. QuickBooks에서의 객체 이름입니다.
- INSTANCE_URL(문자열)-(필수) 유효한 QuickBooks 인스턴스 URL입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 QuickBooks Rest API 버전입니다.
- REALM_ID(문자열)-요청을 보내는 개별 QuickBooks Online 회사를 식별하는 ID입니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

QuickBooks 커넥터의 제한 사항 및 참고 사항

다음은 QuickBooks 커넥터의 제한 사항입니다.

- taxAgency API에서 필터링을 통한 정렬이 예상대로 작동하지 않습니다.

Salesforce에 연결

Salesforce는 영업, 고객 서비스, 전자 상거래 등에 도움이 되는 고객 관계 관리(CRM) 소프트웨어를 제공합니다. Salesforce 사용자인 경우 Salesforce 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, ETL 작업에서 Salesforce를 데이터 소스 또는 대상으로 사용할 수 있습니다. 이러한 작업을 실행하여 Salesforce와 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송하세요.

주제

- Salesforce에 대한 AWS Glue 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Salesforce 구성
- 시스템 관리자 프로필 적용
- Salesforce 연결 구성
- Salesforce에서 읽기

- Salesforce에 쓰기
- Salesforce 연결 옵션
- Salesforce 커넥터의 제한 사항
- Salesforce에 대한 권한 부여 코드 흐름 설정
- Salesforce용 JWT 베어러 OAuth 흐름 설정

Salesforce에 대한 AWS Glue 지원

AWS Glue에서는 다음과 같이 Salesforce를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Salesforce에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Salesforce에 레코드를 쓸 수 있습니다.

지원되는 Salesforce API 버전

다음 Salesforce API 버전이 지원됩니다.

- v58.0
- v59.0
- v60.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 IAM 정책은 AWS Glue ETL 작업 내에서 Salesforce 연결을 생성하고 관리하고 사용하는 데 필요한 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
```

```
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

다음 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Salesforce 연결을 생성할 때 네트워크 옵션을 제공하는 경우 다음 작업도 IAM 역할에 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

제로 ETL Salesforce 연결은 제로 ETL 사전 조건을 참조하세요.

제로 ETL Salesforce 연결은 제로 ETL 사전 조건을 참조하세요.

Salesforce 구성

AWS Glue를 사용하여 Salesforce로 또는 Salesforce에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Salesforce 계정이 있습니다.
- Salesforce 계정이 API 액세스에 대해 활성화되어 있습니다. API 액세스는 기본적으로 Enterprise, Unlimited, Developer, Performance 에디션에 대해 활성화됩니다.

이러한 요구 사항을 충족하면 Salesforce 계정에 AWS Glue를 연결할 준비가 된 것입니다. AWS Glue에서는 AWS 관리형 연결된 앱을 사용하여 나머지 요구 사항을 처리합니다.

Salesforce용 AWS 관리형 연결된 앱

AWS 관리형 연결된 앱을 사용하면 더 적은 단계로 Salesforce 연결을 생성할 수 있습니다. Salesforce에서 연결된 앱은 외부 애플리케이션(예: AWS Glue)이 OAuth 2.0을 사용하여 Salesforce 데이터에 액세스할 수 있도록 권한을 부여하는 프레임워크입니다. AWS 관리형 연결된 앱을 사용하려면 AWS Glue 콘솔을 사용하여 Salesforce 연결을 생성합니다. 연결을 구성할 때 OAuth 권한 부여 유형을 권한 부여 코드로 설정하고 AWS 관리형 클라이언트 애플리케이션 사용 확인란을 선택한 상태로 둡니다.

연결을 저장하면 Salesforce 계정에 로그인하고 AWS Glue 액세스를 승인하도록 Salesforce로 리디렉션됩니다.

시스템 관리자 프로필 적용

Salesforce에서 다음 단계에 따라 시스템 관리자 프로필을 적용합니다.

1. Salesforce에서 설정 > 연결된 앱 > 연결된 앱 OAuth 사용으로 이동합니다.
2. 연결된 앱 목록에서 AWS Glue를 찾아 설치를 선택합니다. 필요한 경우 차단 해제를 선택합니다.
3. 설정 > 연결된 앱 관리로 이동한 다음 AWS Glue를 선택합니다. OAuth 정책에서 관리자가 승인한 사용자가 사전 인증됨을 선택하고 시스템 관리자 프로필을 선택합니다. 이 작업은 AWS Glue에 대한 액세스를 시스템 관리자 프로필이 있는 사용자로만 제한합니다.

시스템 관리자 프로필 적용

Salesforce에서 다음 단계에 따라 시스템 관리자 프로필을 적용합니다.

1. Salesforce에서 설정 > 연결된 앱 > 연결된 앱 OAuth 사용으로 이동합니다.
2. 연결된 앱 목록에서 AWS Glue를 찾아 설치를 선택합니다. 필요한 경우 차단 해제를 선택합니다.
3. 설정 > 연결된 앱 관리로 이동한 다음 AWS Glue를 선택합니다. OAuth 정책에서 관리자가 승인한 사용자가 사전 인증됨을 선택하고 시스템 관리자 프로필을 선택합니다. 이 작업은 AWS Glue에 대한 액세스를 시스템 관리자 프로필이 있는 사용자로만 제한합니다.

Salesforce 연결 구성

Salesforce 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. JWT_TOKEN 권한 부여 유형의 경우 시크릿에는 해당 값과 함께 JWT_TOKEN 키가 포함되어야 합니다.
   b. AuthorizationCode 권한 부여 유형의 경우:
      i. AWS 관리형 연결된 앱의 경우 비어 있는 보안 암호 또는 임시 값이 지정된 보안 암호를 제공해야 합니다.
      ii. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 `Consumer Secret`를 포함해야 합니다.
   c. 참고: AWS Glue에서 연결에 대한 보안 암호를 생성해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Salesforce를 선택합니다.
   b. 연결하려는 Salesforce 인스턴스의 INSTANCE_URL을 제공합니다.
   c. Salesforce 환경을 제공합니다.
   d. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

e. 연결에 사용할 OAuth2 권한 부여 유형을 선택합니다. 권한 부여 유형은 AWS Glue에서 Salesforce와 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다. 선택한 항목은 연결을 생성하기 전에 충족해야 하는 요구 사항에 영향을 미칩니다. 다음 유형 중 하나를 선택할 수 있습니다.

- JWT_BEARER 권한 부여 유형: 이 권한 부여 유형은 JSON 웹 토큰(JWT)을 Salesforce 인스턴스에서 특정 사용자의 권한으로 미리 생성할 수 있으므로 자동화 시나리오에 적합합니다. 생성자는 JWT의 유효 기간을 제어할 수 있습니다. AWS Glue에서는 JWT를 사용하여 Salesforce API를 직접 호출하는 데 사용되는 액세스 토큰을 가져올 수 있습니다.

  이 흐름을 사용하려면 사용자가 Salesforce 인스턴스에서 연결된 앱을 생성하여 사용자를 위한 JWT 기반 액세스 토큰을 발급할 수 있어야 합니다.

  JWT 베어러 OAuth 흐름에 대한 연결된 앱을 생성하는 방법에 대한 자세한 내용은 OAuth 2.0 JWT bearer flow for server-to-server integration을 참조하세요. Salesforce 연결된 앱을 사용하여 JWT 베어러 흐름을 설정하려면 Salesforce용 JWT 베어러 OAuth 흐름 설정 섹션을 참조하세요.

- AUTHORIZATION_CODE 권한 부여 유형: 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Salesforce 인스턴스 URL을 제외한 OAuth 관련 정보를 제공할 필요가 없는 AWS Glue 연결된 앱(AWS Glue 관리형 클라이언트 애플리케이션)에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Salesforce로 리디렉션합니다. 사용자가 Salesforce에 로그인하고 Salesforce 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

  사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때 Salesforce에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 시크릿을 제공하기로 선택할 수 있습니다. 이 시나리오에

서는 여전히 Salesforce로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 자세한 내용은 the section called "Salesforce에 대한 권한 부여 코드 흐름 설정" 섹션을 참조하세요.

f. OAuth 2.0 토큰을 저장하기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

g. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

4. 네트워크 옵션을 제공하는 경우 IAM 역할에 다음 권한도 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

AWS CLI를 사용하여 Salesforce 연결 구성

AWS CLI를 사용하여 Salesforce 연결을 생성할 수 있습니다.

```
aws glue create-connection --connection-input \
"{\"Name\": \"salesforce-conn1\",\"ConnectionType\": \"SALESFORCE\",
\"ConnectionProperties\": {\"ROLE_ARN\": \"arn:aws:iam::123456789012:role/glue-role\",
\"INSTANCE_URL\": \"https://example.my.salesforce.com\"},\"ValidateCredentials\": true,
\"AuthenticationConfiguration\": {\"AuthenticationType\": \"OAUTH2\",\"SecretArn\":
 \"arn:aws:secretsmanager:us-east-1:123456789012:secret:salesforce-conn1-secret-IAmcdk
\",\"OAuth2Properties\": {\"OAuth2GrantType\": \"JWT_BEARER\",\"TokenUrl\": \"https://
login.salesforce.com/services/oauth2/token\"}}}" \
```

```
--endpoint-url https://glue.us-east-1.amazonaws.com \
--region us-east-1
```

Salesforce에서 읽기

사전 조건

읽으려는 Salesforce sObject. `Account` `Case` 또는 `Opportunity`와 같은 객체 이름이 필요합니다.

예시:

```
salesforce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v60.0"
    }
```

쿼리 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(`PARTITION_FIELD`, `LOWER_BOUND`, `UPPER_BOUND`, `NUM_PARTITIONS`)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 `NUM_PARTITIONS`개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- `PARTITION_FIELD`: 쿼리 분할에 사용할 필드의 이름입니다.
- `LOWER_BOUND`: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜 또는 타임스탬프 필드의 경우 커넥터는 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  "TIMESTAMP \"1707256978123\""
  "TIMESTAMP '2018-01-01 00:00:00.000 UTC'"
  "TIMESTAMP \"2018-01-01 00:00:00 Pacific/Tahiti\""
  "TIMESTAMP \"2018-01-01 00:00:00\""
  "TIMESTAMP \"-123456789\" Pacific/Tahiti"
  "TIMESTAMP \"1702600882\""
  ```
- `UPPER_BOUND`: 선택한 파티션 필드의 상한 값(경계 제외).
- `NUM_PARTITIONS`: 파티션 수.

- TRANSFER_MODE: SYNC와 ASYNC의 두 가지 모드를 지원합니다. 기본값은 SYNC입니다. ASYNC로 설정하면 Bulk API 2.0 Query가 처리에 활용됩니다.

예시:

```
salesforce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v60.0",
        "PARTITION_FIELD": "SystemModstamp",
        "LOWER_BOUND": "TIMESTAMP '2021-01-01 00:00:00 Pacific/Tahiti'",
        "UPPER_BOUND": "TIMESTAMP '2023-01-10 00:00:00 Pacific/Tahiti'",
        "NUM_PARTITIONS": "10",
        "TRANSFER_MODE": "ASYNC"
    }
```

Salesforce에 쓰기

사전 조건

쓰려는 Salesforce sObject입니다. Account Case 또는 Opportunity와 같은 객체 이름이 필요합니다.

Salesforce 커넥터는 네 가지 쓰기 작업을 지원합니다.

- INSERT
- UPSERT
- UPDATE
- DELETE

UPSERT 쓰기 작업을 사용하는 경우 레코드의 외부 ID 필드를 지정하려면 ID_FIELD_NAMES 옵션을 제공해야 합니다.

연결 옵션을 추가할 수도 있습니다.

- TRANSFER_MODE: SYNC와 ASYNC의 두 가지 모드를 지원합니다. 기본값은 SYNC입니다. ASYNC로 설정하면 Bulk API 2.0 Ingest가 처리에 활용됩니다.

- FAIL_ON_FIRST_ERROR: 기본값은 FALSE로, 실패한 쓰기 레코드가 몇 개 있더라도 AWS Glue 작업이 모든 데이터 처리를 계속한다는 의미입니다. TRUE로 설정하면 실패한 쓰기 레코드가 있는 경우 AWS Glue 작업이 실패하고 처리를 계속하지 않습니다.

예제

```
salesforce_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="salesforce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v60.0",
        "WRITE_OPERATION": "INSERT",
        "TRANSFER_MODE": "ASYNC",
        "FAIL_ON_FIRST_ERROR":""true"
    }
```

Salesforce 연결 옵션

다음 연결 옵션은 Salesforce 커넥터에서 지원됩니다.

- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Salesforce에서의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용하려는 Salesforce Rest API 버전.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.

  필터 조건자를 제공할 때에는 AND 연산자만 지원됩니다. 현재 OR, IN과 같은 다른 연산자는 지원되지 않습니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

- `IMPORT_DELETED_RECORDS`(문자열) - 기본값: FALSE. 읽기에 사용됩니다. 쿼리하는 동안 삭제 레코드를 가져오려면 다음을 수행합니다.
- `WRITE_OPERATION`(문자열) - 기본값: INSERT. 쓰기에 사용됩니다. 값은 INSERT, UPDATE, UPSERT, DELETE여야 합니다.
- `ID_FIELD_NAMES`(문자열) - 기본값: null. UPDATE 및 UPSERT에 필요합니다.

Salesforce 커넥터의 제한 사항

다음은 Salesforce 커넥터의 제한 사항입니다.

- Spark SQL만 지원하며 Salesforce SOQL은 지원되지 않습니다.
- 작업 북마크는 지원되지 않습니다.
- Salesforce 필드 이름은 대/소문자를 구분합니다. Salesforce에 쓸 때 데이터는 Salesforce 내에 정의된 필드의 대/소문자와 일치해야 합니다.

Salesforce에 대한 권한 부여 코드 흐름 설정

OAuth 2.0 권한 부여 코드 흐름을 활성화하려면 Salesforce 공개 문서를 참조하세요.

연결된 앱을 구성하려면:

1. OAuth 설정 활성화 확인란을 활성화합니다.
2. 콜백 URL 텍스트 필드에 AWS Glue에 대한 리디렉션 URL을 하나 이상 입력합니다.

   리디렉션 URL 형식은 다음과 같습니다.

   https://*region*.console.aws.amazon.com/gluestudio/oauth

   이 URL에서 region은 AWS Glue를 사용하여 Salesforce의 데이터를 전송하는 AWS 리전의 코드입니다. 미국 동부(버지니아 북부) 리전의 경우 이 코드는 `us-east-1`입니다. 해당 리전의 URL은 다음과 같습니다.

   https://us-east-1.console.aws.amazon.com/gluestudio/oauth

   AWS Glue가 지원하는 AWS 리전과 해당 코드에 대해서는 AWS 일반 참조의 AWS Glue 엔드포인트 및 할당량을 참조하세요.
3. 웹 서버 흐름에 보안 암호 필요 확인란을 활성화합니다.
4. 사용 가능한 OAuth 범위 목록에서 다음 범위를 추가합니다.

- API로 사용자 데이터 관리(api)
- 사용자 지정 권한 액세스(custom_permissions)
- 자격 증명 URL 서비스 액세스(id, profile, email, address, phone)
- 고유한 사용자 식별자 액세스(openid)
- 언제든지 요청 수행(refresh_token, offline_access)

5. 연결된 앱의 새로 고침 토큰 정책을 해지될 때까지 새로 고침 토큰이 유효함으로 설정합니다. 그렇지 않으면 새로 고침 토큰이 만료되면 작업이 실패합니다. 새로 고침 토큰 정책을 확인하고 편집하는 방법에 대한 자세한 내용은 Salesforce 설명서의 연결된 앱에 대한 OAuth 액세스 정책 관리를 참조하세요.

Salesforce용 JWT 베어러 OAuth 흐름 설정

OAuth 2.0 JSON 웹 토큰과 서버 간 통합을 활성화하려면 Salesforce 공개 설명서를 참조하세요.

JWT를 생성하고 Salesforce에서 연결된 앱을 적절하게 구성한 후에는 Secrets Manager 보안 암호에 설정된 `JWT_TOKEN` 키를 사용하여 새 Salesforce 연결을 생성할 수 있습니다. 연결을 생성할 때 OAuth 권한 부여 유형을 JWT 보유자 토큰으로 설정합니다.

Salesforce Marketing Cloud에 연결

Salesforce Marketing Cloud는 이메일, 모바일, 소셜 및 온라인 마케팅을 위한 마케팅 자동화 및 분석 소프트웨어 제공업체입니다. 또한 컨설팅 및 구현 서비스도 제공합니다. Salesforce Marketing Cloud 사용자는 Salesforce Marketing Cloud 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, ETL 작업에서 Salesforce Marketing Cloud를 데이터 소스 또는 대상으로 사용할 수 있습니다. 이러한 작업을 실행하여 Salesforce Marketing Cloud 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue Salesforce Marketing Cloud 지원
- 연결 생성 및 사용 API 작업이 포함된 정책
- Salesforce Marketing Cloud 구성
- Salesforce Marketing Cloud 연결 구성
- Salesforce Marketing Cloud 엔터티에서 읽기
- Salesforce Marketing Cloud 엔터티에 쓰기
- Salesforce Marketing Cloud 연결 옵션

- Salesforce Marketing Cloud 커넥터에 대한 제한 사항 및 참고 사항

AWS Glue Salesforce Marketing Cloud 지원

AWS Glue 는 다음과 같이 Salesforce Marketing Cloud를 지원합니다.

소스로 지원되나요?

예. 작업을 사용하여 AWS Glue ETL Salesforce Marketing Cloud에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Salesforce Marketing Cloud API 버전

지원되는 Salesforce Marketing Cloud API 버전은 다음과 같습니다.

- v1

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려면 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS 관리 콘솔을 사용할 때 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

Salesforce Marketing Cloud 구성

AWS Glue를 사용하여 Salesforce Marketing Cloud에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Salesforce Marketing Cloud 계정이 있습니다. 자세한 내용은 Salesforce Marketing Cloud 계정 생성 단원을 참조하십시오.
- Salesforce Marketing Cloud 계정이 API 액세스에 대해 활성화되어 있습니다. API 액세스는 기본적으로 Enterprise, Unlimited, Developer, Performance 에디션에 대해 활성화됩니다.

이러한 요구 사항을 충족하면 Salesforce Marketing Cloud 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Salesforce Marketing Cloud에서 다른 작업을 수행하지 않아도 됩니다.

Salesforce Marketing Cloud 계정 생성

Salesforce Marketing Cloud의 경우 계정 생성을 위해 공급업체에 문의해야 합니다. 사용자 또는 회사가 Salesforce와 연결되어 있는 경우 Salesforce 계정 관리자에게 문의하여 Salesforce Marketing Cloud 라이선스를 요청합니다. 그렇지 않으면 다음과 같이 Salesforce 담당자에게 연락처를 요청할 수 있습니다.

1. https://www.salesforce.com/in/products/marketing-cloud/overview/로 이동하고 Sign up을 선택하세요.
2. 페이지 오른쪽 상단의 문의하기 링크를 선택하세요.

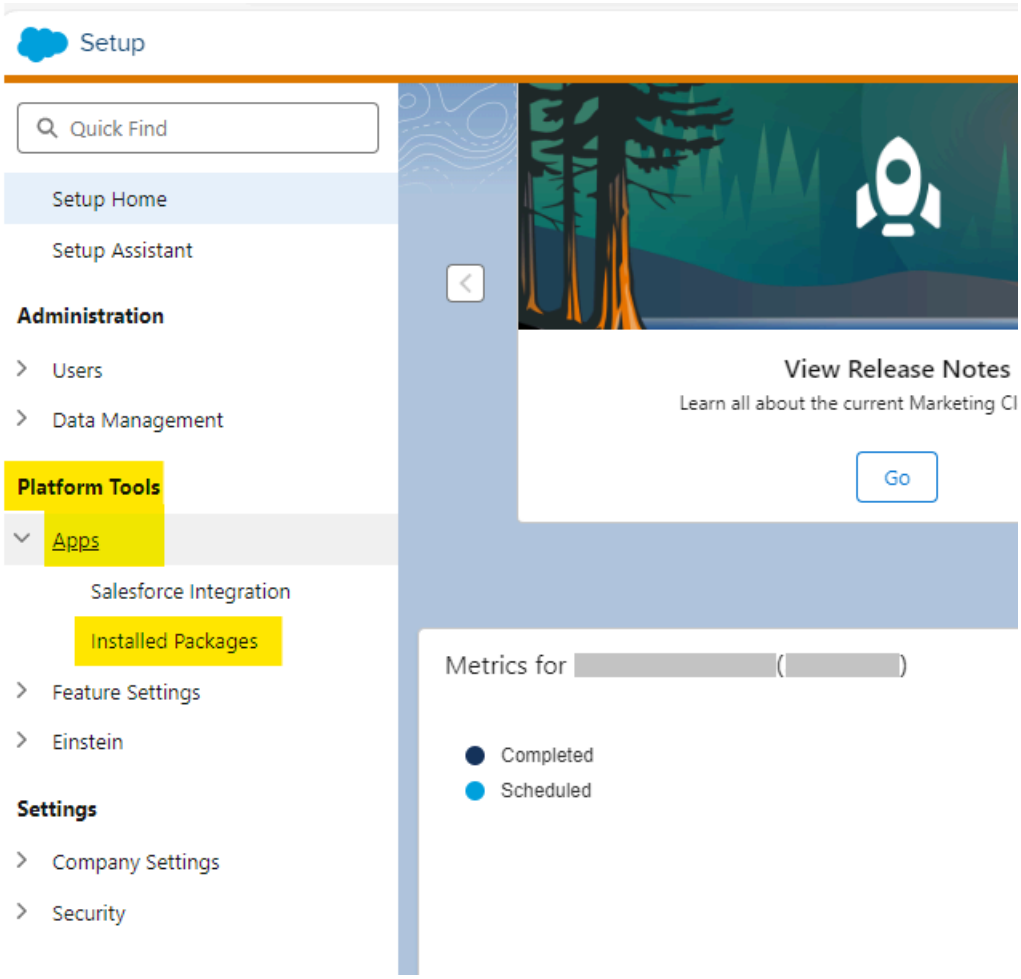
3. 양식에 필요한 정보를 입력하고 연락 요청을 선택하세요.

Salesforce 담당자가 요구 사항을 논의하기 위해 연락을 드릴 것입니다.

프로젝트 및 OAuth 2.0 자격 증명 생성

프로젝트 및 OAuth 2.0 자격 증명을 가져오는 방법:

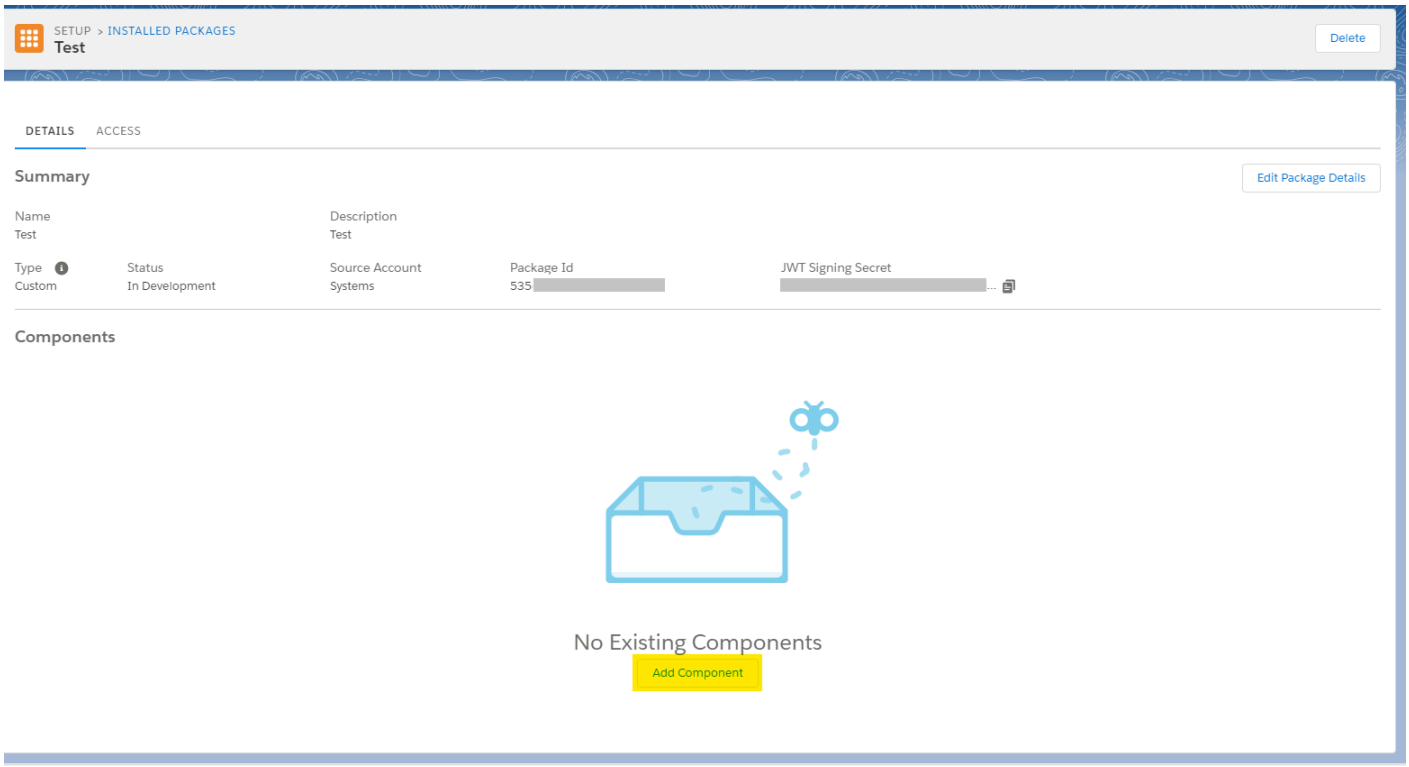
1. 사용자 이름과 암호를 사용하여 Salesforce Marketing Cloud 인스턴스에 로그인하고 등록된 휴대폰 번호를 사용하여 인증하세요.
2. 오른쪽 상단에서 프로필을 클릭한 다음, Setup으로 이동하세요.
3. Platform Tools에서 Apps를 선택하고 Installed Packages를 선택하세요.



4. Installed Packages 페이지에서 오른쪽 상단에 있는 New를 클릭하세요. 패키지의 이름 및 설명을 제공하세요.

   패키지를 저장하세요. 패키지를 저장한 후 패키지 세부 정보를 볼 수 있습니다.

5. 패키지의 Details 페이지에서 Component 섹션 아래 Add Component를 선택하세요.



6. Component Type을 'API Integration'으로 선택하고 Next를 클릭하세요.
7. Integration Type을 'Server-to-Server'(클라이언트 자격 증명 OAuth 권한 부여 유형)로 선택하고 Next를 클릭하세요.
8. 요구 사항에 따라 범위를 추가하고 Save를 클릭하세요.

Salesforce Marketing Cloud 연결 구성

Salesforce Marketing Cloud는 OAuth2에 대한 CLIENT CREDENTIALS 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 클라이언트가 사용자의 컨텍스트 외부에서 액세스 토큰을 얻는 데 사용하므로 2각 OAuth 2.0으로 간주됩니다. AWS Glue는 클라이언트 ID와 클라이언트 암호를 사용하여 사용자가 정의한 사용자 지정 서비스에서 제공하는 Salesforce Marketing Cloud API를 인증할 수 있습니다.
- 각 사용자 지정 서비스는 API 전용 사용자가 소유하며, API 전용 사용자는 서비스에 특정 작업을 수행하도록 권한을 부여하는 역할 및 권한 집합을 가집니다. 액세스 토큰은 단일 사용자 지정 서비스와 연결됩니다.
- 이 권한 부여 유형은 수명이 짧은 액세스 토큰을 생성하며 ID 엔드포인트를 호출하여 갱신할 수 있습니다.

- 클라이언트 자격 증명이 포함된 OAuth 2.0용 퍼블릭 Salesforce Marketing Cloud 설명서는 Set Up Your Development Environment for Enhanced Packages를 참조하세요.

Salesforce Marketing Cloud 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Salesforce Marketing Cloud를 선택합니다.
   b. 연결하려는 Salesforce Marketing Cloud의 `Subdomain Endpoint` 항목을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Salesforce Marketing Cloud 엔터티에서 읽기

사전 조건

읽으려는 Salesforce Marketing Cloud 객체. 객체 이름(예: `Activity` 또는 `Campaigns`)이 필요합니다. 다음 표에는 지원되는 엔터티가 나와 있습니다.

소스에 대해 지원되는 엔터티:

개체	인터페이스	필터링 가능	제한 지원	정렬 기준 지원	SELECT * 지원	분할 지원
이벤트 알림 콜백	REST	아니요	아니요	아니요	예	아니요
시드-목록	REST	아니요	예	아니요	예	아니요
설정	REST	예	예	아니요	예	아니요
도메인 확인	REST	예	예	예	예	아니요
객체 중첩된 태그	REST	예	아니요	아니요	예	아니요
Contact	REST	아니요	예	아니요	예	아니요
이벤트 알림 구독	REST	아니요	아니요	아니요	예	아니요
메시지 전송	REST	아니요	예	아니요	예	아니요
활동	SOAP	아니요	아니요	아니요	예	예
반송 이벤트	SOAP	아니요	아니요	아니요	예	예
클릭 이벤트	SOAP	아니요	아니요	아니요	예	예

개체	인터페이스	필터링 가능	제한 지원	정렬 기준 지원	SELECT * 지원	분할 지원
콘텐츠 영역	SOAP	아니요	아니요	아니요	예	예
데이터 확장	SOAP	아니요	예	아니요	예	예
이메일	SOAP	아니요	예	아니요	예	예
전달된 이메일 이벤트	SOAP	아니요	예	아니요	예	예
이메일 OptInEvent 전달	SOAP	아니요	예	아니요	예	예
링크	SOAP	아니요	예	아니요	예	예
링크 전송	SOAP	아니요	예	아니요	예	예
나열	SOAP	아니요	예	아니요	예	예
구독자 나열	SOAP	아니요	예	아니요	예	예
전송되지 않은 이벤트	SOAP	아니요	예	아니요	예	예
열린 이벤트	SOAP	아니요	예	아니요	예	예
Send	SOAP	아니요	예	아니요	예	예
전송된 이벤트	SOAP	아니요	예	아니요	예	예

개체	인터페이스	필터링 가능	제한 지원	정렬 기준 지원	SELECT * 지원	분할 지원
구독자	SOAP	아니요	예	아니요	예	예
설문조사 이벤트	SOAP	아니요	예	아니요	예	예
구독 해지 이벤트	SOAP	아니요	예	아니요	예	예
감사 이벤트	REST	아니요	예	예	예	아니요
Campaigns	REST	아니요	예	예	예	아니요
상호작용	REST	아니요	예	예	예	아니요
콘텐츠 자산	REST	아니요	예	예	예	아니요

REST에 대한 예제:

```
salesforcemarketingcloud _read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Campaigns",
        "API_VERSION": "v1",
        "INSTANCE_URL": "https://**********************.rest.marketingcloudapis.com"
    }
```

SOAP에 대한 예제:

```
salesforcemarketingcloud _read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Activity",
        "API_VERSION": "v1",
```

```
        "INSTANCE_URL": "https://**********************.soap.marketingcloudapis.com"
    }
```

Salesforce Marketing Cloud 엔터티 및 필드 세부 정보:

다음 표에서는 Salesforce Marketing Cloud 엔터티를 설명합니다. 정적 메타데이터를 포함하는 REST 엔터티와 동적 메타데이터를 포함하는 SOAP 엔터티가 있습니다.

정적 메타데이터를 포함하는 REST 엔터티:

개체	필드	데이터 유형	지원되는 연산자
이벤트 알림 콜백	callbackId	String	
	callbackName	String	
	url	String	
	maxBatchSize	Integer	
	status	String	
	statusReason	String	
시드-목록	id	String	
	name	String	
	설명	String	
	activeSeedCount	Integer	
설정	customerKey	String	
	name	String	
	설명	String	
	locationType	String	'='
	awsFileTransferLocation	Struct	

개체	필드	데이터 유형	지원되는 연산자
도메인 확인	enterpriseId	Integer	
	status	String	'='
	domainType	String	'='
	memberId	Integer	
	emailSendTime	DateTime	
	도메인	String	
	isSendable	불	
객체 중첩된 태그	id	Integer	
	modifiedDate	DateTime	
	tags	나열	
	name	String	
	설명	String	
	parentId	Integer	
Contact	values	나열	
이벤트 알림 구독	subscriptionName	String	
	callbackId	String	
	callbackName	String	
	eventCategoryTypes	나열	
	filters	나열	
	url	String	

개체	필드	데이터 유형	지원되는 연산자
	maxBatchSize	Integer	
	subscriptionId	String	
	status	String	
	statusReason	String	
메시지 전송	deliveryTime	DateTime	
	id	String	
	messageId	String	
	status	String	
	아래로 변경합니다.	Struct	
상호작용	status	String	'='
	id	String	
	키	String	
	name	String	
	lastPublishedDate	DateTime	
	설명	String	
	version	Integer	
	workflowApiVersion	Integer	
	createdDate	DateTime	
	modifiedDate	DateTime	
	goals	Struct	

개체	필드	데이터 유형	지원되는 연산자
	stats	Struct	
	entryMode	String	
	defaults	Struct	
	executionMode	Struct	
	definitionId	String	
콘텐츠 자산	id	Integer	
	customerKey	String	
	objectId	String	
	contentType	String	
	assetType	Struct	
	name	String	
	설명	String	
	owner	Struct	
	createdDate	DateTime	
	createdBy	Struct	
	modifiedDate	DateTime	
	modifiedBy	Struct	
	thumbnail	Struct	
	category	Struct	
	meta	Struct	

개체	필드	데이터 유형	지원되는 연산자
	뷰	Struct	
	availableViews	Struct	
	data	Struct	
	legacyData	Struct	
	modelVersion	Integer	
	버전	Integer	
	Locked	불	
	FileProperties	Struct	
	Tags	나열	
	내용	String	
	설계	String	
	SuperContent	String	
	CustomFields	Struct	
	블록	Struct	
	MinBlocks	Integer	
	MaxBlocks	Integer	
	채널	Struct	
	AllowedBlocks	나열	
	슬롯	Struct	
	BusinessUnitAvailability	Struct	

개체	필드	데이터 유형	지원되는 연산자
	sharingProperties	Struct	
	sharingProperties. sharedWith	Struct	
	sharingProperties. sharingType	String	
	템플릿	Struct	
	파일	String	
	GenerateFrom	String	
감사 이벤트	id	Integer	
	createdDate	DateTime	
	memberId	Integer	
	enterpriseId	Integer	
	employee	Struct	
	objectType	Struct	
	작업	Struct	
	객체	Struct	
	transactionId	String	
Campaigns	id	Integer	
	createdDate	DateTime	
	modifiedDate	DateTime	
	name	String	

개체	필드	데이터 유형	지원되는 연산자
	설명	String	
	campaignCode	String	
	color	String	
	favorite	불	

동적 메타데이터를 포함하는 SOAP 엔터티:

개체	데이터 유형	지원되는 연산자
활동	String	LIKE,!=,=
	Struct	
	Integer	!=,=,>=,<=,<,>
	배정밀도 실수	!=,=,>=,<=,<,>
	불	!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
반송 이벤트	Integer	!=,=,>=,<=,<,>
	DateTime	>=,<=,<,>,=,BETWEEN
	String	LIKE,!=,=
	Struct	
클릭 이벤트	Integer	!=,=,>=,<=,<,>
	DateTime	>=,<=,<,>,=,BETWEEN
	String	LIKE,!=,=

개체	데이터 유형	지원되는 연산자
	Struct	
콘텐츠 영역	Struct	
	String	LIKE,!=,=
	Integer	!=,=,>=,<=,<,>
	DateTime	>=,<=,<,>,=,BETWEEN
	불	!=,=
데이터 확장	DateTime	>=,<=,<,>,=,BETWEEN
	String	LIKE,!=,=
이메일	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	불	!=,=
	Struct	
전달된 이메일 이벤트	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	Struct	
전달된 이메일 OptInEvent	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN

개체	데이터 유형	지원되는 연산자
	Struct	
링크	Integer	!=,=,>=,<=,<,>
링크 전송	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	배정밀도 실수	!=,=,>=,<=,<,>
나열	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	Struct	
구독자 나열	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	Struct	
전송되지 않은 이벤트	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	Struct	
열린 이벤트	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN

개체	데이터 유형	지원되는 연산자
	Struct	
Send	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	불	!=,=
	Struct	
전송된 이벤트	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	Struct	
구독자	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	Struct	
설문조사 이벤트	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=
	DateTime	>=,<=,<,>,=,BETWEEN
	Struct	
구독 해지 이벤트	Integer	!=,=,>=,<=,<,>
	String	LIKE,!=,=

개체	데이터 유형	지원되는 연산자
	DateTime	>=,<=,<,>,=,BETWEEN
	불	!=,=
	Struct	

분할 쿼리

Salesforce Marketing Cloud에서 정수 및 DateTime 데이터 유형 필드는 필드 기반 분할을 지원합니다.

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  타임스탬프 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  “2024-05-07T02:03:00.00Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예시:

```
salesforcemarketingcloud_read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "ListSubscriber",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "CreatedDate"
        "LOWER_BOUND": "2023-09-07T02:03:00.000Z"
        "UPPER_BOUND": "2024-05-07T02:03:00.000Z"
```

```
        "NUM_PARTITIONS": "10"
    }
```

Salesforce Marketing Cloud 엔터티에 쓰기

사전 조건

- 쓰기 대상 Salesforce Marketing 객체입니다. `List`나 `Campaigns` 또는 아래 표에 설명된 다른 엔터티와 같은 객체의 이름을 지정해야 합니다.
- Salesforce Marketing Cloud 커넥터는 3가지 쓰기 작업을 지원합니다.
  - INSERT
  - UPSERT
  - UPDATE

  `UPDATE` 및 `UPSERT` 쓰기 작업을 사용하는 경우 레코드의 외부 ID 필드를 지정하려면 `ID_FIELD_NAMES` 옵션을 제공해야 합니다.

대상에 지원되는 엔터티:

개체	우선순위	인터페이스(REST, SOAP 등)	삽입 가능	업데이트 가능	업서트 가능
Campaigns	P0	REST	Y- 단일	Y- 단일	N
콘텐츠 자산	P0	REST	Y- 단일, 대량	Y- 단일	N
Contact	P1	REST	Y- 단일	Y- 단일	N
도메인 확인	P1	REST	Y- 단일	Y- 단일, 대량	N
이벤트 알림 콜백	P1	REST	Y- 단일	Y- 단일	N
이벤트 알림 구독	P1	REST	Y- 단일	Y- 단일	N
메시지 전송	P1	REST	Y- 단일	N	N

개체	우선순위	인터페이스(REST, SOAP 등)	삽입 가능	업데이트 가능	업서트 가능
객체 중첩된 태그	P2	REST	Y- 단일	Y- 단일	N
시드-목록	P1	REST	Y- 단일	Y- 단일	N
설정	P1	REST	Y- 단일	Y- 단일	N
데이터 확장	P0	SOAP	Y- 단일	Y- 단일	Y- 단일
이메일	P0	SOAP	Y- 단일	Y- 단일	N
나열	P0	SOAP	Y- 단일	Y- 단일	N
Send	P0	SOAP	Y- 단일	N	N
구독자	P0	SOAP	Y- 단일	Y- 단일	N

REST에 대한 INSERT 작업의 예시:

```
salesforcemarketingcloud _write = glueContext.write_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Campaigns",
        "API_VERSION": "v1",
        "writeOperation" : "INSERT",
        "INSTANCE_URL": "https://**********************.rest.marketingcloudapis.com"
    }
```

SOAP에 대한 INSERT 작업의 예시:

```
salesforcemarketingcloud _write = glueContext.write_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "List",
        "API_VERSION": "v1",
```

```
        "writeOperation" : "INSERT",
        "INSTANCE_URL": "https://*********************.rest.marketingcloudapis.com"
    }
```

REST에 대한 UPDATE 작업의 예시:

```
salesforcemarketingcloud _write = glueContext.write_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Campaigns",
        "API_VERSION": "v1",
        "writeOperation" : "UPDATE",
         "ID_FIELD_NAMES": "id",
        "INSTANCE_URL": "https://*********************.rest.marketingcloudapis.com"
    }
```

SOAP에 대한 UPDATE 작업의 예시:

```
salesforcemarketingcloud _write = glueContext.write_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "List",
        "API_VERSION": "v1",
        "writeOperation" : "UPDATE",
         "ID_FIELD_NAMES": "id",
        "INSTANCE_URL": "https://*********************.rest.marketingcloudapis.com"
    }
```

SOAP에 대한 UPSERT 작업의 예시:

```
salesforcemarketingcloud _write = glueContext.write_dynamic_frame.from_options(
    connection_type="salesforcemarketingcloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "DataExtension/Insert-***E/6*******3",
        "API_VERSION": "v1",
        "writeOperation" : "UPSERT",
        "INSTANCE_URL": "https://*********************.rest.marketingcloudapis.com"
    }
```

Salesforce Marketing Cloud 연결 옵션

다음은 Salesforce Marketing Cloud에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Salesforce Marketing Cloud에서의 객체 이름입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Salesforce Marketing Cloud Rest 및 SOAP API 버전입니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- WRITE_OPERATION(문자열) - 기본값: INSERT. 쓰기에 사용됩니다. 값은 INSERT, UPDATE, UPSERT여야 합니다.
- ID_FIELD_NAMES(문자열) - 기본값: null. UPDATE/UPSERT에 필요합니다.

Salesforce Marketing Cloud 커넥터에 대한 제한 사항 및 참고 사항

다음은 Salesforce Marketing Cloud 커넥터에 대한 제한 사항 및 참고 사항입니다.

- DateTime 데이터 유형 필드에 필터를 사용하는 경우 ‘yyyy-mm-ddThh:MM:ssZ’ 형식으로 값을 전달해야 합니다.
- 데이터 미리 보기에서 부울 데이터 유형 값은 공백으로 제공됩니다.
- SOAP 엔터티의 경우 최대 2개의 필터를 정의할 수 있으며, REST 엔터티의 경우 하나의 필터만 정의할 수 있어서 필터를 사용한 파티셔닝 테스트가 제한됩니다.
- SaaS 측에서 몇 가지 예상치 못한 동작이 관찰되었습니다. `linksend` 엔터티의 Link.Alias 필드는 CONTAINS 연산자(예: Link.Alias CONTAINS "ViewPrivacyPolicy")를 지원하지 않으며, 데이터 확장 엔터티의 필터 연산자(예: EQUALS 및 GREATER THAN)는 예상 결과를 반환하지 않습니다.

- SFMC ClickEvent SOAP API는 새로 생성된 레코드 반영이 지연되므로 최근에 생성된 레코드는 API 응답에서 즉시 사용할 수 없을 수 있습니다.

  예: 2025-01-10T14:30:00에 5개의 새 ClickEvent 레코드를 생성하고 SOAP API를 사용하여 즉시 가져오는 경우 응답에 5개의 레코드가 모두 포함되지 않을 수 있습니다. 새로 생성된 레코드가 API 응답에 표시되는 데 최대 5분이 걸릴 수 있습니다. 이 지연은 데이터 검색과 예약된 실행에도 영향을 미칠 수 있습니다.
- 2025-03-11T04:46:00(밀리초 제외) 및 2025-03-11T04:46:00.000Z의 두 가지 DateTime 형식은 AWS Glue(밀리초 포함)에서 쓰기 작업을 수행할 때 지원됩니다.
- 이벤트 알림 구독 엔터티의 경우 확인된 콜백 URL에 대해서만 구독을 생성할 수 있으며 콜백당 최대 200개의 구독을 보유할 수 있습니다.
- 이벤트 알림 콜백 엔터티의 경우 계정당 최대 50개의 레코드를 생성할 수 있습니다.

Salesforce Commerce Cloud에 연결

B2C Commerce API는 B2C Commerce 인스턴스와 상호 작용하기 위한 RESTful API 모음입니다. Salesforce Commerce API와 그 약어인 SCAPI, 또는 Commerce API라고도 합니다.

API를 통해 개발자는 전체 스토어프론트부터 사용자 지정 판매자 도구, Business Manager 강화에 이르기까지 다양한 애플리케이션을 구축할 수 있습니다. 모든 B2C Commerce 고객의 경우 API를 추가 비용 없이 사용할 수 있습니다.

API는 Shopper API와 Admin API로 나뉩니다. 그리고 각 그룹 내에서 API 패밀리와 관련 기능에 초점을 맞춘 소규모 그룹으로 나뉩니다.

주제

- AWS Glue의 Salesforce Commerce Cloud에 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Salesforce Commerce Cloud 구성
- Salesforce Commerce Cloud 연결 구성
- Salesforce Commerce Cloud 엔터티에서 읽기
- Salesforce Commerce Cloud 연결 옵션 참조
- 제한 사항

AWS Glue의 Salesforce Commerce Cloud에 지원

AWS Glue에서는 다음과 같이 Salesforce Commerce Cloud를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Salesforce Commerce Cloud에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Salesforce Commerce Cloud API 버전

v1.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및

Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Salesforce Commerce Cloud 구성

AWS Glue를 사용하여 Salesforce Commerce Cloud에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- clientId 및 clientSecret이 있는 Salesforce Commerce Cloud 클라이언트 애플리케이션이 있습니다.
- Salesforce Commerce Cloud 계정이 API 액세스에 대해 활성화되어 있습니다.

이러한 요구 사항을 충족하면 Salesforce Commerce Cloud 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Salesforce Commerce Cloud에서 다른 작업을 수행하지 않아도 됩니다.

Salesforce Commerce Cloud 연결 구성

Salesforce Commerce Cloud는 OAuth2에 대한 CLIENT CREDENTIALS 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 클라이언트가 사용자의 컨텍스트 외부에서 액세스 토큰을 얻는 데 사용하므로 2각 OAuth 2.0으로 간주됩니다. AWS Glue는 클라이언트 ID와 클라이언트 보안 암호를 사용하여 사용자가 정의한 사용자 지정 서비스에서 제공하는 Salesforce Commerce Cloud API를 인증할 수 있습니다.
- 각 사용자 지정 서비스는 API 전용 사용자가 소유하며, API 전용 사용자는 서비스에 특정 작업을 수행하도록 권한을 부여하는 역할 및 권한 집합을 가집니다. 액세스 토큰은 단일 사용자 지정 서비스와 연결됩니다.
- 이 권한 부여 유형은 수명이 짧은 액세스 토큰을 생성하며 ID 엔드포인트를 호출하여 갱신할 수 있습니다.
- 클라이언트 자격 증명 생성에 대한 Salesforce Commerce Cloud 설명서에서 정보를 찾으려면 Salesforce 설명서를 참조하세요.

다음 단계를 따라 Salesforce Commerce Cloud 연결을 구성합니다.

1. AWS Secrets Manager에서 다음 세부 정보를 사용하여 보안 암호를 생성합니다. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. 고객 관리형 연결된 앱 - 보안 암호는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 데이터 연결에서 연결 생성을 선택합니다.
   b. 데이터 소스를 선택할 때 Salesforce Commerce Cloud를 선택합니다.
   c. Salesforce Commerce Cloud 단축 코드, 조직 ID 및 사이트 ID를 입력합니다.
   d. Salesforce Commerce Cloud 계정의 Salesforce Commerce Cloud 도메인 URL을 선택합니다.
   e. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   f. 연결하려는 Salesforce Commerce Cloud의 OAuth 범위(선택 사항으로 사용자 관리형 클라이언트 애플리케이션 ClientId)를 입력합니다.
   g. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   h. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
4. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

Salesforce Commerce Cloud 엔터티에서 읽기

사전 조건

- 읽으려는 Salesforce Commerce Cloud 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Assignments	예	예	예	예	예
Campaigns	예	예	예	예	예
Catalogs	예	예	예	예	예
Categories	예	예	예	예	예
Coupons	예	예	예	예	예
Gift Certificates	예	예	예	예	예
Products	예	예	예	예	예
Promotions	예	예	예	예	예
Source Code Groups	예	예	예	예	예

예

```
salesforce_commerce_cloud_read = glueContext.create_dynamic_frame.from_options(
     connection_type="SalesforceCommerceCloud",
     connection_options={
         "connectionName": "connectionName",
         "ENTITY_NAME": "campaign",
         "API_VERSION": "v1"
```

```
    }
```

Salesforce Commerce Cloud 엔터티 및 필드 세부 정보

엔터티 목록:

- Assignments: https://developer.salesforce.com/docs/commerce/commerce-api/references/assignments
- Campaigns: https://developer.salesforce.com/docs/commerce/commerce-api/references/campaigns
- Catalogs: https://developer.salesforce.com/docs/commerce/commerce-api/references/catalogs
- Categories: https://developer.salesforce.com/docs/commerce/commerce-api/references/catalogs?meta=searchCategories
- Gift Certificates: https://developer.salesforce.com/docs/commerce/commerce-api/references/gift-certificates
- Products: https://developer.salesforce.com/docs/commerce/commerce-api/references/products
- Promotions: https://developer.salesforce.com/docs/commerce/commerce-api/references/promotions
- Source Code Groups: https://developer.salesforce.com/docs/commerce/commerce-api/references/source-code-groups

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-02-06".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준 분할 필드 지원 세부 정보는 아래 표에 나와 있습니다.

개체	분할 필드	데이터 유형
Campaigns	lastModified	DateTime
Campaigns	startDate	DateTime
Campaigns	endDate	DateTime
Catalogs	creationDate	DateTime
Categories	creatiionDate	DateTime
Gift Certificates	merchantId	String
Gift Certificates	creatiionDate	DateTime
Products	creatiionDate	DateTime
Products	lastModified	DateTime
Source Code Groups	creationDate	DateTime
Source Code Groups	startTime	DateTime
Source Code Groups	endTime	DateTime

예

```
salesforceCommerceCloud_read = glueContext.create_dynamic_frame.from_options(
    connection_type="SalesforceCommerceCloud",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "coupons",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "creationDate"
        "LOWER_BOUND": "2020-05-01T20:55:02.000Z"
        "UPPER_BOUND": "2024-07-11T20:55:02.000Z"
        "NUM_PARTITIONS": "10"
    }
```

Salesforce Commerce Cloud 연결 옵션 참조

다음은 Salesforce Commerce Cloud의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Salesforce Commerce Cloud에서의 객체 이름.
- `API_VERSION`(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용하려는 Salesforce Commerce Cloud Rest API 버전. 예: v1.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- `PARTITION_FIELD`(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- `LOWER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- `UPPER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- `NUM_PARTITIONS`(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

제한 사항

다음은 Salesforce Commerce Cloud 커넥터의 제한 사항입니다.

- 분할 시 Contains 필터가 예상대로 작동하지 않습니다.
- CDN Zones의 엔터티는 샌드박스 인스턴스를 지원하지 않으며, 개발 및 프로덕션 인스턴스 유형만 지원합니다. 자세한 내용은 https://help.salesforce.com/s/articleView?id=cc.b2c_embedded_cdn_overview.htm 페이지를 참조하세요.
- Salesforce Commerce Cloud에서 동적 메타데이터를 가져올 API 엔드포인트가 없습니다. 따라서 제품 및 카테고리 엔터티의 사용자 지정 필드를 지원하는 프로비저닝이 없습니다.
- 사이트 ID는 필수 쿼리 파라미터입니다. 사용자 지정 커넥터 설정을 통해 사이트 ID 값을 전달해야 합니다. 자세한 내용은 Base URL and Request Formation을 참조하세요.
- 아래 표에 언급된 대로 서로 다른 연산자를 조합하여 단일 API 요청에서 최대 두 개의 필드(있는 경우 Levels 제외)에 필터를 적용할 수 있습니다.

필터 기준	지원 여부
단일 API 요청에서 CONTAINS 연산자가 있는 1개 필드.	예
단일 API 요청에서 Equals 연산자가 있는 1개 필드.	예
단일 API 요청에서 BETWEEN 연산자가 있는 1개 필드.	예
단일 API 요청에서 CONTAINS 연산자가 있는 2개 이상의 필드.	아니요
단일 API 요청에서 Equals 연산자가 있는 2개 이상의 필드.	아니요
단일 API 요청에서 BETWEEN 연산자가 있는 2개 이상의 필드.	아니요
단일 API 요청에서 Equals 연산자가 있는 1개 필드와 CONTAINS 연산자가 있는 1개 필드.	예
단일 API 요청에서 BETWEEN 연산자가 있는 1개 필드와 CONTAINS 연산자가 있는 1개 필드.	예
단일 API 요청에서 BETWEEN 연산자가 있는 1개 필드와 Equals 연산자가 있는 1개 필드.	예
단일 API 요청에서 Equals 연산자가 있는 1개 필드, CONTAINS 연산자가 있는 1개 필드와 BETWEEN 연산자가 있는 1개 필드.	아니요
INCREMENTAL PULL이 단일 API 요청에 적용된 경우 Equals 연산자가 있는 1개 필드.	예

필터 기준	지원 여부
INCREMENTAL PULL이 단일 API 요청에 적용된 경우 CONTAINS 연산자가 있는 1개 필드.	예
INCREMENTAL PULL이 단일 API 요청에 적용된 경우 BETWEEN 연산자가 있는 1개 필드.	아니요
INCREMENTAL PULL이 단일 API 요청에 적용된 경우 Equals 연산자 1개와 CONTAINS 연산자 1개.	아니요

- 일부 엔터티에서 검색할 때 필드의 데이터 유형은 검색 가능한 필드로 사용되는 경우와 다릅니다. 그에 따라 이러한 필드에는 필터 특성의 프로비저닝이 없습니다. 다음 표에는 이러한 필드에 대한 세부 정보가 나와 있습니다.

검색 번호	Entity Name	검색 가능한 필드 이름	검색(Search) 가능한 데이터 유형	검색(Retrieve) 가능한 데이터 유형
1	카탈로그	name	String	Struct
2	카탈로그	설명	String	Struct
3	범주	name	String	Struct
4	범주	설명	String	Struct
5	Product	name	String	Struct
6	Product	searchable	불	Struct
7	Product	validFrom	DateTime	Struct
8	Product	validTo	DateTime	Struct
9	Product	type	String	Struct
10	Product	onlineFlag	불	Struct

검색 번호	Entity Name	검색 가능한 필드 이름	검색(Search) 가능한 데이터 유형	검색(Retrieve) 가능한 데이터 유형
11	Promotion	name	String	Struct

Salesforce Marketing Cloud Account Engagement에 연결

Salesforce Marketing Cloud Account Engagement는 회사가 의미 있는 연결을 생성하고, 추가 파이프라인을 생성하며, 더 많은 거래를 성사할 수 있도록 영업을 지원하는 마케팅 자동화 솔루션입니다. Salesforce Marketing Cloud Account Engagement 사용자인 경우 Salesforce Marketing Cloud Account Engagement 계정에 AWS Glue를 연결할 수 있습니다. Salesforce Marketing Cloud Account Engagement를 ETL 작업에서 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Salesforce Marketing Cloud Account Engagement에서 AWS 서비스 또는 기타 지원되는 애플리케이션으로 데이터를 전송합니다.

주제

- Salesforce Marketing Cloud Account Engagement에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Salesforce Marketing Cloud Account Engagement 구성
- Salesforce Marketing Cloud Account Engagement 연결 구성
- Salesforce Marketing Cloud Account Engagement 엔터티에서 읽기
- Salesforce Marketing Cloud Account Engagement 연결 옵션
- Salesforce Marketing Cloud Account Engagement 커넥터에 대한 제한 사항 및 참고 사항

Salesforce Marketing Cloud Account Engagement에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Salesforce Marketing Cloud Account Engagement를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Salesforce Marketing Cloud Account Engagement에서 비동기 또는 동기화 모드로 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Salesforce Marketing Cloud Account Engagement API 버전

다음 Salesforce Marketing Cloud Account Engagement API 버전이 지원됩니다.

- v5

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Salesforce Marketing Cloud Account Engagement 구성

AWS Glue를 사용하여 Salesforce Marketing Cloud Account Engagement에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Salesforce 마케팅 계정이 있습니다.
- Salesforce 계정에 대해 라이선스가 부여된 Account Engagement 계획이 있습니다.
- Salesforce 사용자를 Account Engagement 사용자와 동기화했습니다.
- OAuth 자격 증명을 얻기 위해 App Manager 아래에서 새 연결된 앱을 생성했습니다.

이러한 요구 사항을 충족하면 Salesforce Marketing Cloud Account Engagement 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Salesforce Marketing Cloud Account Engagement 연결 구성

권한 부여 유형은 AWS Glue에서 Salesforce Marketing Cloud Account Engagement와 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다. 선택한 항목은 연결을 생성하기 전에 충족해야 하는 요구 사항에 영향을 미칩니다. Salesforce Marketing Cloud Account Engagement에서는 OAuth 2.0에 대한 AUTHORIZATION_CODE 권한 부여 유형만 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.
- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Salesforce Marketing Cloud Account Engagement에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Salesforce Marketing Cloud Account Engagement로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Salesforce Marketing Cloud Account Engagement 설명서는 Authentication을 참조하세요.

Salesforce Marketing Cloud Account Engagement 연결을 구성하는 방법:

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Salesforce Marketing Cloud Account Engagement를 선택하세요.
   b. 연결하려는 Salesforce Marketing Cloud Account Engagement 인스턴스의 `INSTANCE_URL` 항목을 제공하세요.
   c. 연결하려는 Salesforce Marketing Cloud Account Engagement 인스턴스의 `PARDOT_BUSINESS_UNIT_ID` 항목을 제공하세요.
   d. 드롭다운에서 적절한 권한 부여 코드 URL을 선택하세요.
   e. 드롭다운에서 적절한 토큰 URL을 선택하세요.
   f. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   g. 사용자 관리형 클라이언트 애플리케이션 클라이언트 ID(연결된 앱의 클라이언트 ID)를 제공합니다.
   h. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다. 선택한 보안 암호에는 값이 연결된 앱의 클라이언트 보안 암호인 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET` 키가 있어야 합니다.
   i. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
3. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

Salesforce Marketing Cloud Account Engagement 엔터티에서 읽기

사전 조건

읽으려는 Salesforce Marketing Cloud Account Engagement 객체. 객체 이름이 필요합니다.

동기화 소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
캠페인	예	예	예	예	예
동적 콘텐츠	예	예	예	예	예
이메일	예	예	예	예	예
이메일 템플릿	예	예	예	예	예
참여 스튜디오 프로그램	예	예	예	예	예
폴더 콘텐츠	예	예	예	예	예
랜딩 페이지	예	예	예	예	예
수명 주기 기록	예	예	예	예	예
수명 주기 단계	예	예	예	예	예
나열	예	예	예	예	예
이메일 나열	예	예	예	예	예
멤버쉽 나열	예	예	예	예	예
기회	예	예	예	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
잠재 고객	예	예	예	예	예
잠재 고객 계정	예	예	예	예	예
User	예	예	예	예	예

예시:

```
salesforcepardot_read = glueContext.create_dynamic_frame.from_options(
    connection_type="SalesforcePardot",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v5"
    }
   )
```

비동기 소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
캠페인	예	아니요	아니요	예	아니요
동적 콘텐츠	예	아니요	아니요	예	아니요
이메일 템플릿	예	아니요	아니요	예	아니요
랜딩 페이지	예	아니요	아니요	예	아니요
수명 주기 기록	예	아니요	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
수명 주기 단계	예	아니요	아니요	예	아니요
나열	예	아니요	아니요	예	아니요
이메일 나열	예	아니요	아니요	예	아니요
멤버쉽 나열	예	아니요	아니요	예	아니요
기회	예	아니요	아니요	예	아니요
잠재 고객	예	아니요	아니요	예	아니요
잠재 고객 계정	예	아니요	아니요	예	아니요
User	예	아니요	아니요	예	아니요

예시:

```
salesforcepardot_read = glueContext.create_dynamic_frame.from_options(
    connection_type="SalesforcePardot",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v5",
        "TRANSFER_MODE": "ASYNC"
    }
   )
```

Salesforce Marketing Cloud Account Engagement 엔터티 및 필드 세부 정보:

다음 엔터티의 필드 세부 정보를 보려면 Salesforce Marketing Cloud Account Engagement API로 이동하여 가이드를 선택하고 오픈 소스 API 래퍼로 스크롤한 다음, 메뉴에서 버전 5 문서를 확장하고 엔터티를 선택합니다.

엔터티 목록:

- 캠페인
- 동적 콘텐츠
- 이메일
- 이메일 템플릿
- 참여 스튜디오 프로그램
- 폴더 콘텐츠
- 랜딩 페이지
- 수명 주기 기록
- 수명 주기 단계
- 나열
- 이메일 나열
- 멤버쉽 나열
- Opportunity
- 잠재 고객
- 잠재 고객 계정
- User

위에서 언급한 필드 외에도 비동기 모드는 아래 표와 같이 각 엔터티에 대해 특정 필터링 가능 필드를 지원합니다.

개체	비동기에서 지원되는 추가 필터링 가능 필드
캠페인	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
동적 콘텐츠	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
이메일 템플릿	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`

개체	비동기에서 지원되는 추가 필터링 가능 필드
참여 스튜디오 프로그램	-
랜딩 페이지	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
수명 주기 기록	`createdAfter` , `createdBefore`
수명 주기 단계	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
나열	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
이메일 나열	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
멤버쉽 나열	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
Opportunity	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
잠재 고객	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`
잠재 고객 계정	`createdAfter` , `createdBefore` , `deleted`

개체	비동기에서 지원되는 추가 필터링 가능 필드
User	`createdAfter` , `createdBefore` , `deleted`, `updatedAfter` , `updatedBefore`

추가 필드에 대한 자세한 내용은 Salesforce Export API를 참조하세요.

커넥터에 대한 다음 고려 사항에 유의합니다.

- 엔터티에서 `delete` 필드 값은 `false`(기본값), `true` 또는 `all`일 수 있습니다.

분할 쿼리

필터 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(`PARTITION_FIELD`, `LOWER_BOUND`, `UPPER_BOUND`, `NUM_PARTITIONS`)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 `NUM_PARTITIONS`개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- `PARTITION_FIELD`: 쿼리 분할에 사용할 필드의 이름입니다.
- `LOWER_BOUND`: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  "2022-01-01T01:01:01.000Z"
  ```

- `UPPER_BOUND`: 선택한 파티션 필드의 상한 값(경계 제외).
- `NUM_PARTITIONS`: 파티션 수.
- `PARTITION_BY`: 수행할 분할 유형. 필드 기반 분할의 경우 "FIELD"를 전달해야 합니다.

예시:

```
salesforcepardot_read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforcepardot",
    connection_options={
```

```
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v5",
        "PARTITION_FIELD": "createdAt"
        "LOWER_BOUND": "2022-01-01T01:01:01.000Z"
        "UPPER_BOUND": "2024-01-01T01:01:01.000Z"
        "NUM_PARTITIONS": "10",
        "PARTITION_BY": "FIELD"
    }
   )
```

Salesforce Marketing Cloud Account Engagement 연결 옵션

다음은 Salesforce Marketing Cloud Account Engagement에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Salesforce Marketing Cloud Account Engagement에서의 객체 이름.
- PARDOT_BUSINESS_UNIT_ID - (필수) 연결을 생성하는 데 사용됩니다. 연결하려는 Salesforce Marketing Cloud Account Engagement 인스턴스의 사업부 ID.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Salesforce Marketing Cloud Account Engagement Rest API 버전.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) -
  - 동기화 모드에서 - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
  - 비동기 모드에서 - 기본값: 현재 DateTime 값(사용자 시간대 기준) - 1년. 읽기에 사용됩니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - (필수) 읽기에 사용됩니다. 유효한 Salesforce Marketing Cloud Account Engagement 인스턴스 URL.
- PARTITION_BY(문자열) - (필수) 읽기에 사용됩니다. 수행할 분할의 유형. 필드 기반 분할의 경우 "FIELD"를 전달해야 합니다.

- TRANSFER_MODE(문자열) - (선택 사항), 비동기 모드에서 작업을 실행하는 데 사용할 값으로, 이 옵션이 제공되지 않은 경우 동기화 모드에서 작업이 실행됩니다.

Salesforce Marketing Cloud Account Engagement 커넥터에 대한 제한 사항 및 참고 사항

다음과 같은 참조 사항 및 제한 사항이 적용됩니다.

- 제한과 분할이 모두 적용되면 제한이 분할보다 우선합니다.
- API 문서에 따라 SalesforceMarketingCloudEngagement에서는 일일 및 동시 요청에 RateLimit를 적용했습니다. 자세한 내용은 Rate Limits를 참조하세요.
- 내보내기 API에는 일일 Account Engagement API 직접 호출 한도와, 계정에 대한 동시 Account Engagement API 직접 호출 한도가 적용됩니다.
- 대기열과 마찬가지로 내보내기/비동기 API 직접 호출은 각 계정에 대해 순차적으로 실행됩니다. 이전 내보내기는 최신 내보내기 전에 처리됩니다.
- 파티션은 비동기 모드에서 지원되지 않습니다.
- 내보내기/비동기 API 직접 호출에 지정된 선택한 필드 수는 150개를 초과할 수 없습니다.
- Prospect 엔터티는 150개 이상의 필드를 지원하지만 한 번에 150개 필드만 선택할 수 있습니다. Select All을 선택하면 일부 필드가 제외됩니다. 제외된 필드에 대한 데이터를 검색하려면 Selected Fields 옵션에 해당 필드를 포함해야 합니다.

  다음은 SELECT_ALL에서 제외된 필드 목록입니다. - updatedBy.firstName, updatedBy.lastName, updatedBy.jobTitle, updatedBy.roleName, updatedBy.salesforceId, updatedBy.createdAt, updatedBy.updatedAt, updatedBy.isDeleted, updatedBy.createdById, updatedBy.updatedById, updatedBy.tagReplacementLanguage
- 컬렉션 필드는 비동기화에 대해 내보낼 수 없습니다. 예를 들어 이메일 목록에서 senderOptions 및 replyToOptions 필드는 지원되지 않습니다.
- 모든 엔터티에 대해 필터는 필수입니다. 필터가 제공되지 않은 경우 기본 필터 조건자가 현재 날짜-시간(표준 시간대에 맞게 조정됨)에서 1년을 뺀 값으로 Created After 필드에 설정됩니다.
- Salesforce Marketing Cloud Account Engagement 제한 사항에 따라 비동기에서 데이터를 가져오는 최대 범위는 1년입니다. 1년 이상의 기간에 대해 쿼리하면 작업에서 오류가 발생합니다.
- 현재 Salesforce Pardot에 버그가 있습니다. 데이터가 없는 단일 필드만 작업에 포함된 경우 필드 값은 올바른 결과를 반환하지 않으며, 대신 필드 이름이 여러 번 반환됩니다. Salesforce Pardot 팀은 문제를 알고 있으며 해결을 위해 적극적으로 노력하고 있습니다.

AWS Glue Studio에서 SAP HANA에 연결

AWS Glue에서는 SAP HANA을 기본으로 지원합니다. AWS Glue Studio에서는 SAP HANA에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스를 제공합니다.

AWS Glue Studio에서 SAP HANA용 통합 연결을 생성합니다. 자세한 내용은 고려 사항 단원을 참조하십시오.

주제

- SAP HANA 커넥션 생성
- SAP HANA 소스 노드 생성
- SAP HANA 대상 노드 생성
- 고급 옵션

SAP HANA 커넥션 생성

AWS Glue에서 SAP HANA에 연결하려면 AWS Secrets Manager 보안 암호에서 SAP HANA 보안 인증 정보를 생성하고 저장한 다음 해당 보안 암호를 SAP HANA AWS Glue 연결에 연결해야 합니다. SAP HANA 서비스와 AWS Glue 간에 네트워크 연결을 구성해야 합니다.

사전 조건:

- SAP HANA 서비스가 Amazon VPC에 있는 경우, 퍼블릭 인터넷을 통과하는 트래픽 없이 AWS Glue 작업이 SAP HANA 서비스와 통신할 수 있도록 Amazon VPC를 구성하십시오.

  Amazon VPC에서 작업을 실행하는 동안 AWS Glue가 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 Amazon VPC가 SAP HANA 엔드포인트와 이 위치 간의 네트워크 트래픽을 허용하도록 구성되어 있는지 확인해야 합니다. 작업을 수행하려면 SAP HANA JDBC 포트와의 TCP 연결을 설정해야 합니다. SAP HANA 포트에 대한 자세한 내용은 SAP HANA 설명서를 참조하십시오. 네트워크 레이아웃에 따라 보안 그룹 규칙, 네트워크 ACL, NAT 게이트웨이 및 피어링 연결을 변경해야 할 수도 있습니다.

SAP HANA에 대한 연결 구성하는 방법:

1. AWS Secrets Manager에서 SAP HANA 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets

Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

- 키/값 페어를 선택하면 값 *saphanaUsername*이 포함된 키 user에 대한 페어를 생성합니다.
- 키/값 페어를 선택하면 값 *saphanaPassword*가 포함된 키 password에 대한 페어를 생성합니다.

2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때 SAP HANA를 선택합니다.
   - SAP HANA URL을 제공할 때는 인스턴스의 URL을 제공하십시오.

     SAP HANA JDBC URL은 jdbc:sap://*saphanaHostname*:*saphanaPort*/?*databaseName*=*saphanaDBname*,*Paramete* 형식입니다

     AWS Glue는 JDBC URL 매개 변수가 필요합니다.

     - databaseName - 연결할 SAP HANA의 기본 데이터베이스입니다.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

AWS Glue SAP HANA 연결을 생성한 후에는 AWS Glue 작업을 실행하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업 권한과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.

SAP HANA 소스 노드 생성

필수 전제 조건

- 이전 섹션 the section called “SAP HANA 커넥션 생성”에서 설명한 것처럼 AWS Secrets Manager 암호로 구성된 AWS Glue SAP HANA 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 읽으려는 SAP HANA 테이블, *tableName* 또는 쿼리 *targetQuery*.

SAP HANA 테이블 이름 및 스키마 이름을 *schemaName*.*tableName* 형식으로 테이블을 지정할 수 있습니다. 테이블이 기본 스키마 "public"에 있는 경우 스키마 이름과 "." 구분 기호는 필요하지 않습

니다. 이것을 *tableIdentifier*라고 합니다. 데이터베이스는 connectionName에서 JDBC URL 매개변수로 제공된다는 점에 유의하십시오.

SAP HANA 데이터 소스 추가

데이터 소스 - SAP HANA 노드를 추가하는 방법:

1. SAP HANA 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 SAP HANA 연결 생성을 선택합니다. 자세한 내용은 이전the section called “SAP HANA 커넥션 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. SAP HANA 소스 옵션을 선택하십시오.
   - 단일 테이블 선택 - 단일 테이블에서 모든 데이터에 액세스할 수 있습니다.
   - 사용자 지정 쿼리 입력 - 사용자 지정 쿼리를 기반으로 여러 테이블의 데이터 세트에 액세스할 수 있습니다.
3. 단일 테이블을 선택한 경우 *tableName*을 입력합니다.

   사용자 지정 쿼리 입력을 선택한 경우 SQL SELECT 쿼리를 입력합니다.
4. 사용자 지정 SAP HANA 속성에서 필요한 경우 파라미터와 값을 입력합니다.

SAP HANA 대상 노드 생성

필수 전제 조건

- 이전 섹션 the section called “SAP HANA 커넥션 생성”에서 설명한 것처럼 AWS Secrets Manager 암호로 구성된 AWS Glue SAP HANA 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 쓰고 싶은 SAP HANA 테이블, *tableName*

  SAP HANA 테이블 이름 및 스키마 이름을 *schemaName*.*tableName* 형식으로 테이블을 지정할 수 있습니다. 테이블이 기본 스키마 "public"에 있는 경우 스키마 이름과 "." 구분 기호는 필요하지 않습니다. 이것을 *tableIdentifier*라고 합니다. 데이터베이스는 connectionName에서 JDBC URL 매개변수로 제공된다는 점에 유의하십시오.

SAP HANA 데이터 대상 추가

데이터 대상 - SAP HANA 노드 추가 방법:

1. SAP HANA 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 SAP HANA 연결 생성을 선택합니다. 자세한 내용은 이전the section called “SAP HANA 커넥션 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. *tableName*을 제공하여 테이블 이름을 구성합니다.
3. 사용자 지정 Teradata 속성에서 필요한 경우 파라미터와 값을 입력합니다.

고급 옵션

SAP HANA 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “SAP HANA 커넥션”를 참조하세요.

SAP OData에 연결

SAP OData는 Advanced Business Application Programming(ABAP)을 사용하여 SAP에 있는 데이터를 쿼리 및 업데이트하는 데 사용되는 표준 웹 프로토콜로, HTTP와 같은 웹 기술을 적용 및 빌드하여 다양한 외부 애플리케이션, 플랫폼 및 디바이스의 정보에 대한 액세스를 제공합니다. 이 제품을 사용하면 SAP 시스템, 애플리케이션 또는 데이터와 원활하게 통합하는 데 필요한 모든 요소에 액세스할 수 있습니다.

주제

- SAP OData에 대한 AWS Glue의 지원
- 연결 만들기
- SAP OData 작업 생성
- SAP OData 상태 관리 스크립트 사용
- 비 ODP 엔터티를 위한 파티셔닝
- SAP OData 연결 옵션
- SAP OData 엔터티 및 필드 세부 정보

SAP OData에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 SAP OData를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 SAP OData에서 데이터를 쿼리할 수 있습니다.

지원되는 SAP OData API 버전

다음 SAP OData API 버전이 지원됩니다.

- 2.0

지원되는 소스

지원되는 소스는 다음과 같습니다.

- ODP(운영 데이터 프로비저닝) 소스:
  - BW 추출기(DataSource)
  - CDS 뷰
  - SLT
- 비 ODP 소스, 예:
  - CDS 뷰 서비스
  - RFC 기반 서비스
  - 사용자 지정 ABAP 서비스

지원되는 SAP 구성 요소

다음은 최소 요구 사항입니다.

- 서비스 검색을 위해 카탈로그 서비스를 활성화해야 합니다.
  - SAP 시스템의 SAP Gateway에서 추출할 운영 데이터 프로비저닝(ODP) 데이터 소스를 구성합니다.
  - OData V2.0: `/IWFND/MAINT_SERVICE` 트랜잭션을 통해 SAP Gateway에서 OData V2.0 카탈로그 서비스를 활성화합니다.
  - `/IWFND/MAINT_SERVICE` 트랜잭션을 통해 SAP Gateway에서 OData V2.0 서비스를 활성화합니다.

  - SAP OData 서비스는 `$top` 및 `$skip`과 같은 클라이언트 측 페이지 매김/쿼리 옵션을 지원해야 합니다. 또한 시스템 쿼리 옵션 `$count`도 지원해야 합니다.
  - SAP 사용자가 서비스를 검색하고 SAP OData 서비스를 사용하여 데이터를 추출하려면 필요한 권한을 제공해야 합니다. SAP에서 제공하는 보안 설명서를 참조하세요.
- OAuth 2.0을 권한 부여 메커니즘으로 사용하려는 경우 OData 서비스에 대해 OAuth 2.0을 활성화하고 SAP 설명서에 따라 OAuth 클라이언트를 등록해야 합니다.
- ODP 데이터 소스를 기반으로 OData 서비스를 생성하려면 SAP Gateway Foundation을 ERP/BW 스택 또는 허브 구성에 로컬로 설치해야 합니다.
  - ERP/BW 애플리케이션의 경우 SAP NetWeaver AS ABAP 스택이 7.50 SP02 이상이어야 합니다.
  - 허브 시스템(SAP Gateway)의 경우 원격 허브 설정을 위해 허브 시스템의 SAP NetWeaver AS ABAP가 7.50 SP01 이상이어야 합니다.
- 비 ODP 소스의 경우 SAP NetWeaver 스택 버전은 7.40 SP02 이상이어야 합니다.

지원되는 인증 방법

지원되는 인증 방법은 다음과 같습니다.

- 기본 인증
- OAuth 2.0

사전 조건

SAP OData 연결을 사용하여 SAP OData에서 데이터를 추출하기 위한 AWS Glue 작업을 시작하기 전에 다음 사전 조건을 완료합니다.

- 관련 SAP OData 서비스는 SAP 시스템에서 활성화하여 데이터 소스를 사용할 수 있도록 해야 합니다. OData 서비스가 활성화되지 않은 경우 Glue 작업은 SAP에서 데이터에 액세스하거나 추출할 수 없습니다.
- SAP에서 기본(사용자 지정) 인증 또는 OAuth 2.0과 같은 적절한 인증 메커니즘을 구성하여 AWS Glue 작업이 SAP OData 서비스와 성공적으로 연결할 수 있도록 해야 합니다.
- AWS Glue 작업에 SAP, Secrets Manager 및 프로세스와 관련된 기타 AWS 리소스에 액세스할 수 있는 적절한 권한을 부여하도록 IAM 정책을 구성합니다.
- SAP 시스템이 프라이빗 네트워크 내에서 호스트되는 경우 AWS Glue 작업이 퍼블릭 인터넷을 통해 민감한 데이터를 노출하지 않고 SAP와 안전하게 통신할 수 있도록 VPC 연결을 구성해야 합니다.

AWS Secrets Manager를 사용하여 AWS Glue 작업이 런타임에 동적으로 검색할 수 있는 SAP 자격 증명과 같은 민감한 정보를 안전하게 저장할 수 있습니다. 이 접근 방식을 사용하면 자격 증명을 하드 코딩할 필요가 없으므로 보안 및 유연성이 향상됩니다.

다음 사전 조건은 AWS Glue와 SAP OData 간의 원활한 통합을 위해 각 구성 요소를 설정하는 방법에 대한 단계별 지침을 제공합니다.

주제

- SAP OData 활성화
- IAM 정책
- 연결/VPC 연결
- SAP 인증
- 인증 비밀을 저장할 AWS Secrets Manager

SAP OData 활성화

SAP OData 연결을 위한 다음 단계를 완료합니다.

ODBC 소스

ODP 공급자에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

- SAP NetWeaver AS ABAP 인스턴스가 있어야 합니다.
- SAP NetWeaver 인스턴스에 데이터를 전송할 ODP 공급자가 포함되어야 합니다. ODP 공급자가 다음을 포함해야 합니다.
  - SAP DataSources(트랜잭션 코드 RSO2)
  - SAP Core Data Services ABAP CDS 뷰
  - SAP BW 또는 SAP BW/4HANA 시스템(InfoObject, DataStore 객체)
  - SAP Landscape Replication Server(SAP SLT)를 통해 SAP 소스 시스템에서 테이블 및 DB 뷰를 실시간 복제
  - SAP ABAP 기반 소스의 SAP HANA 정보 뷰
- SAP NetWeaver 인스턴스에는 SAP Gateway Foundation 구성 요소가 있어야 합니다.
- ODP 공급자로부터 데이터를 추출하는 OData 서비스를 생성해야 합니다. OData 서비스를 생성하려면 SAP Gateway Service Builder를 사용합니다. ODP 데이터에 액세스하기 위해 Amazon AppFlow는 OData API를 사용하여 이 서비스를 직접적으로 호출합니다. 자세한 내용은 SAP BW/4HANA 설명서의 OData를 통해 OData 데이터를 추출하기 위한 서비스 생성을 참조하세요.

- ODP 데이터 소스를 기반으로 OData 서비스를 생성하려면 SAP Gateway Foundation을 ERP/BW 스택 또는 허브 구성에 로컬로 설치해야 합니다.
  - ERP/BW 애플리케이션의 경우 SAP NetWeaver AS ABAP 스택이 7.50 SP02 이상이어야 합니다.
  - 허브 시스템(SAP Gateway)의 경우 원격 허브 설정을 위해 허브 시스템의 SAP NetWeaver AS ABAP가 7.50 SP01 이상이어야 합니다.

비ODP 소스

- SAP NetWeaver 스택 버전은 7.40 SP02 이상이어야 합니다.
- 서비스 검색을 위해 카탈로그 서비스를 활성화해야 합니다.
  - OData V2.0: `/IWFND/MAINT_SERVICE` 트랜잭션을 통해 SAP Gateway에서 하나 이상의 OData V2.0 카탈로그 서비스를 활성화할 수 있습니다.
- SAP OData 서비스는 `$top` 및 `$skip`과 같은 클라이언트 측 페이지 매김/쿼리 옵션을 지원해야 합니다. 또한 시스템 쿼리 옵션 `$count`도 지원해야 합니다.
- OAuth 2.0의 경우 OData 서비스에 대해 OAuth 2.0을 활성화하고 SAP 설명서에 따라 OAuth 클라이언트를 등록하고 다음과 같이 권한이 부여된 리디렉션 URL을 설정해야 합니다.
  - `https://<region>.console.aws.amazon.com/gluestudio/oauth`. `<region>`을 AWS Glue가 실행 중인 리전으로 바꿉니다(예: us-east-1).
  - HTTPS를 통해 연결하려면 보안 설정을 활성화해야 합니다.
- SAP 사용자가 서비스를 검색하고 SAP OData 서비스를 사용하여 데이터를 추출하려면 필요한 권한을 제공해야 합니다. SAP에서 제공하는 보안 설명서를 참조하세요.

IAM 정책

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
```

```
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue"
            ],
            "Resource": "*"
         }
    ]
}
```

역할은 작업에서 사용하는 모든 리소스(예: Amazon S3)에 대한 액세스 권한을 부여해야 합니다. 위 메서드를 사용하지 않으려는 경우 다음 관리형 IAM 정책을 대신 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.
- SecretsManagerReadWrite – AWS Management Console을 통해 AWS Secrets Manager에 대한 읽기/쓰기 액세스를 제공합니다. 참고: 여기에는 IAM 작업이 제외되므로 교체 구성이 필요한 경우 `IAMFullAccess`와 함께 사용합니다.

VPC를 구성하는 데 필요한 IAM 정책/권한

VPC 연결을 사용하여 AWS Glue 연결을 생성하는 동안 다음 IAM 권한이 필요합니다. 자세한 내용은 AWS Glue에 대한 IAM 정책 생성을 참조하세요.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

연결/VPC 연결

VPC 연결 단계:

1. 기존 VPC 연결을 사용하거나 Amazon VPC 설명서에 따라 새 연결을 생성합니다.
2. 트래픽을 인터넷으로 라우팅하는 NAT 게이트웨이가 있어야 합니다.
3. VPC 엔드포인트를 Amazon S3 게이트웨이로 선택하여 연결을 생성합니다.
4. AWS에서 제공하는 DNS 서비스를 사용하려면 DNS 확인 및 DNS 호스트 이름을 활성화합니다.
5. 생성된 VPC로 이동하여 STS, AWS Glue, Secrets Manager와 같은 다양한 서비스에 필요한 엔드포인트를 추가합니다.
   a. 엔드포인트 생성을 선택합니다.
   b. 서비스 범주에서 AWS 서비스를 선택합니다.
   c. 서비스 이름에서 연결할 서비스를 선택합니다.
   d. VPC 및 DNS 이름 활성화를 선택합니다.
   e. VPC 연결에 필요한 VCP 엔드포인트:
      i. STS
      ii. AWS Glue
      iii. 보안 관리자

보안 그룹 구성

보안 그룹은 AWS Glue가 연결할 수 있도록 AWS Glue VPC에서 수신 대기 포트로의 트래픽을 허용해야 합니다. 소스 IP 주소의 범위를 최대한 제한하는 것이 좋습니다.

AWS Glue에는 자체로부터의 모든 인바운드 트래픽을 허용하는 특수 보안 그룹이 필요합니다. 보안 그룹에서 발생하는 모든 트래픽을 허용하는 자체 참조 규칙을 생성할 수 있습니다. 기존 보안 그룹을 수정하고 보안 그룹을 소스로 지정할 수 있습니다.

URL 엔드포인트(NLB 또는 SAP 인스턴스)의 HTTPS 포트에서 통신을 개방합니다.

연결 옵션

- 내부 및 외부 NLB와의 HTTPS 연결, 자체 서명된 SSL 인증서가 아닌 인증 기관(CA)의 SSL 인증서
- SAP 인스턴스와의 HTTPS 연결, 자체 서명된 SSL 인증서가 아닌 인증 기관(CA)의 SSL 인증서

SAP 인증

SAP 커넥터는 사용자 지정(SAP 기본 인증임) 및 OAUTH 인증 방법을 모두 지원합니다.

사용자 지정 인증

AWS Glue는 SAP 시스템에 대한 연결을 설정하기 위한 방법으로 사용자 지정(기본 인증)을 지원하므로 보안 액세스를 위해 사용자 이름 및 암호를 사용할 수 있습니다. 이 권한 부여 유형은 SAP OData 인스턴스에서 특정 사용자의 권한을 통해 사용자 이름 및 암호를 미리 사용할 수 있으므로 자동화 시나리오에 적합합니다. AWS Glue에서는 사용자 이름 및 암호를 사용하여 SAP OData API를 인증할 수 있습니다. AWS Glue에서 기본 권한 부여는 사용자 지정 권한 부여로 구현됩니다.

기본 권한 부여 흐름에 대한 퍼블릭 SAP OData 설명서는 HTTP Basic Authentication을 참조하세요.

OAuth 2.0 인증

AWS Glue는 SAP 시스템에 대한 연결을 설정하기 위한 보안 인증 메커니즘으로 OAuth 2.0도 지원합니다. 이를 통해 원활한 통합이 가능하면서도 최신 인증 표준을 준수하고 데이터 액세스의 보안을 강화할 수 있습니다.

AUTHORIZATION_CODE 권한 부여 유형

권한 부여 유형은 AWS Glue에서 SAP OData와 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다. SAP OData에서는 `AUTHORIZATION_CODE` 권한 부여 유형만 지원합니다. 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다.

사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 SAP OData에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 SAP OData로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 SAP OData 설명서는 Authentication Using OAuth 2.0을 참조하세요.

인증 비밀을 저장할 AWS Secrets Manager

SAP OData 연결 비밀을 AWS Secrets Manager에 저장하고, IAM 정책 섹션에 지정된 대로 검색에 필요한 권한을 구성하고, 연결을 생성하는 동안 사용해야 합니다.

AWS Secrets Manager용 AWS Management Console을 사용하여 SAP 소스에 대한 비밀을 생성합니다. 자세한 내용은 AWS Secrets Manager 비밀 생성을 참조하세요. AWS Secrets Manager의 세부 정보에는 다음 코드의 요소가 포함되어야 합니다.

사용자 지정 인증 비밀

<SAP 사용자 이름> 대신 SAP 시스템 사용자 이름, <SAP 사용자 이름 암호> 대신 해당 암호, 그리고 True 또는 False를 입력해야 합니다. 이 컨텍스트에서 `basicAuthDisableSSO`를 `true`로 설정하면 기본 인증 요청에 대해 AWS Single Sign-On(SSO)가 비활성화되어 각 요청에 대해 명시적 사용자 자격 증명이 필요합니다. 반대로 `false`로 설정하면 사용 가능한 경우 기존 SSO 세션을 사용할 수 있습니다.

```
{
   "basicAuthUsername": "<your SAP username>",
   "basicAuthPassword": "<your SAP username password>",
   "basicAuthDisableSSO": "<True/False>",
   "customAuthenticationType": "CustomBasicAuth"
}
```

OAuth 2.0 비밀

OAuth 2.0을 인증 메커니즘으로 사용하는 경우 AWS Secrets Manager의 비밀에는 다음 형식의 사용자 관리형 클라이언트 애플리케이션 ClientId가 있어야 합니다. <클라이언트 비밀> 대신 SAP 클라이언트 비밀을 입력해야 합니다.

```
{"USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET": "<your client secret>"
```

```
}
```

연결 만들기

SAP OData 연결을 구성하는 방법:

1. AWS Management Console에 로그인하여 AWS Glue 콘솔을 엽니다. AWS Glue Studio에서 아래 단계에 따라 연결을 생성합니다.
   a. 왼쪽 패널에서 데이터 연결을 클릭합니다.
   b. 연결 생성을 클릭합니다.
   c. 데이터 소스 선택에서 SAP OData를 선택합니다.
   d. 연결하려는 SAP OData 인스턴스의 애플리케이션 호스트 URL을 제공하세요. 이 애플리케이션 호스트 URL은 비 VPC 연결을 위해 퍼블릭 인터넷을 통해 액세스할 수 있어야 합니다.
   e. 연결하려는 SAP OData 인스턴스의 애플리케이션 서비스 경로를 제공하세요. 이 경로는 카탈로그 서비스 경로와 동일합니다. 예: `/sap/opu/odata/iwfnd/catalogservice;v=2`. AWS Glue는 특정 객체 경로를 허용하지 않습니다.
   f. 연결하려는 SAP OData 인스턴스의 클라이언트 번호를 제공하세요. 사용 가능한 값은 [001-999] 입니다. 예: 010
   g. 연결하려는 SAP OData 인스턴스의 포트 번호를 제공하세요. 예: 443
   h. 연결하려는 SAP OData 인스턴스의 로그온 언어를 제공하세요. 예: EN
   i. AWS Glue가 수임할 수 있고 IAM 정책 섹션에 설명된 권한이 있는 AWS IAM 역할을 선택합니다.
   j. AWS Glue에서 이 연결에 사용할 인증 유형을 드롭다운 목록에서 선택합니다. OAUTH2 또는 사용자 지정
      i. 사용자 지정 - 인증 비밀을 저장할 AWS Secrets Manager 섹션에 지정된 대로 생성한 비밀을 선택합니다.
      ii. OAuth 2.0 - OAuth 2.0의 경우에만 다음 항목을 입력합니다.
         A. 사용자 관리형 클라이언트 애플리케이션 ClientId에 클라이언트 ID를 입력합니다.
         B. AWS Secrets Manager에 인증 비밀을 저장할 AWS Secrets Manager 섹션에서 생성한 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`(클라이언트 비밀)를 입력합니다.
         C. 권한 부여 코드 URL에 권한 부여 코드 URL을 입력합니다.
         D. 권한 부여 토큰 URL에 권한 부여 토큰 URL을 입력합니다.
         E. OAuth 범위에 OAuth 범위를 공백으로 구분하여 입력합니다. 예시: `/IWFND/SG_MED_CATALOG_0002 ZAPI_SALES_ORDER_SRV_0001`

k. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다. 자세한 내용은 연결/VPC 연결을 참조하세요.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다. 자세한 내용은 IAM 정책 단원을 참조하십시오.
3. 연결 테스트를 선택하고 연결을 테스트합니다. 연결 테스트가 통과하면 다음을 클릭하여 연결 이름을 입력하고 연결을 저장합니다. 네트워크 옵션(VPC)을 선택한 경우 연결 테스트 기능을 사용할 수 없습니다.

SAP OData 작업 생성

AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 참조

운영 데이터 프로비저닝(ODP) 소스

운영 데이터 프로비저닝(ODP)은 다양한 대상 애플리케이션의 데이터 추출 및 복제를 지원하는 데 사용할 수 있는 기술 인프라를 제공하며 이러한 시나리오에서 델타 메커니즘을 지원합니다. 델타 절차의 경우 소스(ODP 공급자)의 데이터는 업데이트 프로세스를 사용하여 델타 대기열(운영 델타 대기열 - ODQ)에 자동으로 기록되거나 추출기 인터페이스를 사용하여 델타 대기열에 전달됩니다. ODP 공급자는 DataSource(추출기), ABAP Core Data Services 뷰(ABAP CDS 뷰), SAP BW 또는 SAP BW/4HANA, SAP Landscape Transformation Replication Server(SLT) 및 SAP HANA 정보 뷰(계산 뷰)일 수 있습니다. 대상 애플리케이션(ODQ '구독자' 또는 보다 일반적으로 'ODP 소비자'라고 함)은 델타 대기열에서 데이터를 검색하고 데이터를 계속 처리합니다.

전체 로드

SAP OData 및 ODP 엔터티의 컨텍스트에서 전체 로드는 단일 작업으로 ODP 엔터티에서 사용 가능한 모든 데이터를 추출하는 프로세스를 나타냅니다. 이 작업은 소스 시스템에서 전체 데이터세트를 검색하여 대상 시스템에 엔터티 데이터의 포괄적인 최신 사본이 로드되도록 보장합니다. 전체 로드는 일반적으로 소스가 증분 로드를 지원하지 않는 경우 또는 대상 시스템의 새로 고침이 필요한 경우에 사용됩니다.

예

DynamicFrame을 생성할 때 `ENABLE_CDC` 플래그를 false로 명시적으로 설정할 수 있습니다. 참고: `ENABLE_CDC`는 기본적으로 false입니다. 델타 대기열을 초기화하지 않으려는 경우 이 플래그를 보내거나 true로 설정할 필요가 없습니다. 이 플래그를 true로 설정하지 않으면 전체 로드 추출이 수행됩니다.

```
sapodata_df = glueContext.create_dynamic_frame.from_options(
    connection_type="SAPOData",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "ENABLE_CDC": "false"
    }, transformation_ctx=key)
```

증분 로드

ODP(운영 데이터 프로비저닝) 엔터티의 컨텍스트에서 증분 로드는 마지막 데이터 추출 이후 소스 시스템에서 새 데이터 또는 변경된 데이터(델타)만 추출하여 이미 처리된 레코드를 사전 처리하지 않도록 합니다. 이 접근 방식은 효율성을 크게 개선하고, 데이터 전송 볼륨을 줄이고, 성능을 향상하고, 시스템 간의 효율적인 동기화를 보장하고, 처리 시간을 최소화합니다. 특히 자주 변경되는 대규모 데이터세트의 경우 효과적입니다.

델타 토큰 기반 증분 전송

이를 지원하는 ODP 지원 엔터티에 대해 변경 데이터 캡처(CDC)를 사용하여 증분 전송을 활성화하려면 다음 단계를 따르세요.

1. 스크립트 모드에서 증분 전송 작업을 생성합니다.
2. DataFrame 또는 Glue DynamicFrame을 생성할 때 `"ENABLE_CDC": "True"` 옵션을 전달해야 합니다. 이 옵션을 사용하면 SAP로부터 변경된 데이터의 후속 검색에 사용할 수 있는 델타 토큰을 수신할 수 있습니다.

델타 토큰은 DELTA_TOKEN 열의 마지막 데이터프레임 행에 있습니다. 이 토큰을 후속 직접 호출에서 커넥터 옵션으로 사용하여 다음 데이터세트를 증분 검색할 수 있습니다.

예

- DynamicFrame을 생성할 때 `ENABLE_CDC` 플래그를 `true`로 설정합니다. 참고: `ENABLE_CDC`는 기본적으로 `false`입니다. 델타 대기열을 초기화하지 않으려는 경우 이 플래그를 보내거나 true로 설정할 필요가 없습니다. 이 플래그를 true로 설정하지 않으면 전체 로드 추출이 수행됩니다.

```
sapodata_df = glueContext.create_dynamic_frame.from_options(
    connection_type="SAPOData",
    connection_options={
        "connectionName": "connectionName",
```

```
        "ENTITY_NAME": "entityName",
        "ENABLE_CDC": "true"
    }, transformation_ctx=key)

# Extract the delta token from the last row of the DELTA_TOKEN column
delta_token_1 = your_logic_to_extract_delta_token(sapodata_df) # e.g.,
 D20241029164449_000370000
```

- 새 이벤트를 검색하는 옵션으로 추출된 델타 토큰을 전달할 수 있습니다.

```
sapodata_df_2 = glueContext.create_dynamic_frame.from_options(
    connection_type="SAPOData",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        // passing the delta token retrieved in the last run
        "DELTA_TOKEN": delta_token_1
    } , transformation_ctx=key)

# Extract the new delta token for the next run
delta_token_2 = your_logic_to_extract_delta_token(sapodata_df_2)
```

DELTA_TOKEN이 있는 마지막 레코드는 소스의 트랜잭션 레코드가 아니며 델타 토큰 값을 전달할 목적으로만 존재합니다.

DELTA_TOKEN 외에도 데이터프레임의 각 행에서 다음 필드가 반환됩니다.

- GLUE_FETCH_SQ: EPOC 타임스탬프에서 생성된 시퀀스 필드(레코드가 수신된 순서에 따름)이며 각 레코드에 대해 고유합니다. 소스 시스템에서 변경 순서를 알거나 설정해야 하는 경우 사용할 수 있습니다. 이 필드는 ODP 지원 엔터티에만 표시됩니다.
- DML_STATUS: 소스에서 새로 삽입된 레코드 및 업데이트된 레코드에 대해서는 UPDATED, 소스에서 삭제된 레코드에 대해서는 DELETED를 표시합니다.

상태를 관리하고 델타 토큰을 재사용하여 예제를 통해 변경된 레코드를 검색하는 방법에 대한 자세한 내용은 SAP OData 상태 관리 스크립트 사용 섹션을 참조하세요.

델타 토큰 무효화

델타 토큰은 서비스 컬렉션 및 사용자와 연결됩니다. 동일한 서비스 컬렉션 및 사용자에 대해 “ENABLE_CDC” : “true”를 사용한 새 초기 풀이 시작되는 경우 이전 초기화의 결과로 발급된 이전

의 모든 델타 토큰은 SAP OData 서비스에서 무효화됩니다. 만료된 델타 토큰으로 커넥터를 간접적으로 호출하면 예외가 발생합니다.

`Could not open data access via extraction API RODPS_REPL_ODP_OPEN`

OData 서비스(비ODP 소스)

전체 로드

비 ODP(운영 데이터 프로비저닝) 시스템의 경우 전체 로드는 소스 시스템에서 전체 데이터세트를 추출하여 대상 시스템에 로드합니다. 비 ODP 시스템은 본질적으로 델타와 같은 고급 데이터 추출 메커니즘을 지원하지 않으므로 프로세스는 간단하지만 데이터 크기에 따라 리소스 집약적일 수 있습니다.

증분 로드

ODP(운영 데이터 프로비저닝)를 지원하지 않는 시스템 또는 엔터티의 경우 변경 사항을 추적하고 추출하는 타임스탬프 기반 메커니즘을 구현하여 증분 데이터 전송을 수동으로 관리할 수 있습니다.

타임스탬프 기반 증분 전송

비 ODP 지원 엔터티(또는 ENABLE_CDC 플래그를 사용하지 않는 ODP 지원 엔터티)의 경우 커넥터에서 `filteringExpression` 옵션을 사용하여 데이터를 검색하려는 `datetime` 간격을 표시할 수 있습니다. 이 방법은 각 레코드가 마지막으로 생성/수정된 시점을 나타내는 데이터의 타임스탬프 필드에 의존합니다.

예

2024-01-01T00:00:00.000 이후 변경된 레코드 검색

```
sapodata_df = glueContext.create_dynamic_frame.from_options(
    connection_type="SAPOData",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "filteringExpression": "LastChangeDateTime >= 2024-01-01T00:00:00.000"
    }, transformation_ctx=key)
```

참고: 이 예제에서 `LastChangeDateTime`은 각 레코드가 마지막으로 수정된 시간을 나타내는 필드입니다. 실제 필드 이름은 특정 SAP OData 엔터티에 따라 다를 수 있습니다.

후속 실행에서 새 데이터의 새 하위 세트를 가져오려면 filteringExpression을 새 타임스탬프로 업데이트해야 합니다. 일반적으로 이전에 검색된 데이터의 최대 타임스탬프 값입니다.

예

```
max_timestamp = get_max_timestamp(sapodata_df)  # Function to get the max timestamp
 from the previous run
next_filtering_expression = f"LastChangeDateTime > {max_timestamp}"

# Use this next_filtering_expression in your next run
```

다음 섹션에서는 이러한 타임스탬프 기반 증분 전송을 관리하기 위한 자동화된 접근 방식을 제공합니다. 이 경우 실행 간에 필터링 표현식을 수동으로 업데이트하지 않아도 됩니다.

SAP OData 상태 관리 스크립트 사용

AWS Glue 작업에서 SAP OData 상태 관리 스크립트를 사용하려면 다음 단계를 수행하세요.

- 퍼블릭 Amazon S3 버킷에서 상태 관리 스크립트를 다운로드합니다.
- 스크립트를 AWS Glue 작업에서 액세스할 권한이 있는 Amazon S3 버킷에 업로드합니다.
- AWS Glue 작업에서 스크립트 참조: AWS Glue 작업을 생성하거나 업데이트할 때 Amazon S3 버킷의 스크립트 경로를 참조하는 '--extra-py-files' 옵션을 전달합니다. 예: --extra-py-files s3://your-bucket/path/to/sap_odata_state_management.py
- AWS Glue 작업 스크립트에서 상태 관리 라이브러리를 가져오고 사용하세요.

델타 토큰 기반 증분 전송 예제

다음은 델타 토큰 기반 증분 전송에 상태 관리 스크립트를 사용하는 방법에 대한 예제입니다.

```
from sap_odata_state_management import StateManagerFactory, StateManagerType, StateType

# Initialize the state manager
state_manager = StateManagerFactory.create_manager(
    manager_type=StateManagerType.JOB_TAG,
    state_type=StateType.DELTA_TOKEN,
    options={
        "job_name": args['JOB_NAME'],
        "logger": logger
    }
```

```
)

# Get connector options (including delta token if available)
key = "SAPODataNode"
connector_options = state_manager.get_connector_options(key)

# Use the connector options in your Glue job
df = glueContext.create_dynamic_frame.from_options(
    connection_type="SAPOData",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "ENABLE_CDC": "true",
        **connector_options
    }
)

# Process your data here...

# Update the state after processing
state_manager.update_state(key, sapodata_df.toDF())
```

타임스탬프 기반 증분 전송 예제

다음은 델타 토큰 기반 증분 전송에 상태 관리 스크립트를 사용하는 방법에 대한 예제입니다.

```
from sap_odata_state_management import StateManagerFactory, StateManagerType, StateType

# Initialize the state manager
state_manager = StateManagerFactory.create_manager(
    manager_type=StateManagerType.JOB_TAG,
    state_type=StateType.DELTA_TOKEN,
    options={
        "job_name": args['JOB_NAME'],
        "logger": logger
    }
)

# Get connector options (including delta token if available)
key = "SAPODataNode"
connector_options = state_manager.get_connector_options(key)

# Use the connector options in your Glue job
df = glueContext.create_dynamic_frame.from_options(
```

```
    connection_type="SAPOData",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "ENABLE_CDC": "true",
        **connector_options
    }
)

# Process your data here...

# Update the state after processing
state_manager.update_state(key, sapodata_df.toDF())
```

두 예제 모두에서 상태 관리 스크립트는 작업 실행 간에 상태(델타 토큰 또는 타임스탬프)를 저장하는 복잡성을 처리합니다. 커넥터 옵션을 가져올 때 마지막 알려진 상태를 자동으로 검색하고 처리 후 상태를 업데이트하므로 각 작업 실행이 새 데이터 또는 변경된 데이터만 처리할 수 있습니다.

비 ODP 엔터티를 위한 파티셔닝

Apache Spark에서 파니셔닝은 병렬 처리를 위해 클러스터의 워커 노드 간에 데이터를 분할하고 배포하는 방법을 나타냅니다. 각 파티션은 태스크에서 독립적으로 처리할 수 있는 데이터의 논리적 청크입니다. 파티셔닝은 성능, 확장성 및 리소스 사용률에 직접적인 영향을 미치는 Spark의 기본 개념입니다. AWS Glue 작업은 Spark의 파티셔닝 메커니즘을 사용하여 데이터세트를 클러스터의 워커 노드에서 병렬로 처리할 수 있는 더 작은 청크(파티션)로 나눕니다. 참고로 ODP 엔터티에는 파티셔닝이 적용되지 않습니다.

자세한 내용은 AWS Glue Spark 및 PySpark 작업을 참조하세요.

사전 조건

읽으려는 SAP OData의 객체. 예를 들어 `/sap/opu/odata/sap/API_SALES_ORDER_SRV/A_SalesOrder` 와 같은 객체/EntitySet 이름이 필요합니다.

예

```
sapodata_read = glueContext.create_dynamic_frame.from_options(
    connection_type="SAPOData",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "/sap/opu/odata/sap/API_SALES_ORDER_SRV/A_SalesOrder"
```

```
    }, transformation_ctx=key)
```

파티셔닝 쿼리

필드 기반 파티셔닝

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다. 정수, 날짜 및 DateTime 필드는 SAP OData 커넥터에서 필드 기반 분할을 지원합니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  데이터 유형이 DateTime인 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예: "2000-01-01T00:00:00.000Z"
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.
- PARTITION_BY: 수행할 파티셔닝 유형. 필드 기반 파티셔닝의 경우 FIELD가 전달됩니다.

예

```
sapodata= glueContext.create_dynamic_frame.from_options(
    connection_type="sapodata",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "/sap/opu/odata/sap/SEPM_HCM_SCENARIO_SRV/EmployeeSet",
        "PARTITION_FIELD": "validStartDate"
        "LOWER_BOUND": "2000-01-01T00:00:00.000Z"
        "UPPER_BOUND": "2020-01-01T00:00:00.000Z"
        "NUM_PARTITIONS": "10",
        "PARTITION_BY": "FIELD"
    }, transformation_ctx=key)
```

레코드 기반 파티셔닝

Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

ODP 엔터티에서 페이지 매김은 다음 토큰/건너뛰기 토큰을 통해 지원되므로 레코드 기반 분할은 비 ODP 엔터티에 대해서만 지원됩니다.

- PARTITION_BY: 수행할 파티셔닝 유형. 레코드 기반 파티셔닝의 경우 COUNT가 전달됩니다.

예제

```
sapodata= glueContext.create_dynamic_frame.from_options(
    connection_type="sapodata",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "/sap/opu/odata/sap/SEPM_HCM_SCENARIO_SRV/EmployeeSet",
        "NUM_PARTITIONS": "10",
        "PARTITION_BY": "COUNT"
    }, transformation_ctx=key)
```

제한 사항/콜아웃

- 페이지 매김은 건너뛰기 토큰/델타 토큰을 사용하여 처리되므로 ODP 엔터티는 레코드 기반 분할과 호환되지 않습니다. 따라서 레코드 기반 분할의 경우 maxConcurrency의 기본값은 사용자 입력과 관계없이 'null'로 설정됩니다.
- 제한과 파티션이 모두 적용되면 제한이 파티셔닝보다 우선합니다.

SAP OData 연결 옵션

다음은 SAP OData에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. SAP OData에서의 객체 이름.

  예: /sap/opu/odata/sap/API_SALES_ORDER_SRV/A_SalesOrder
- API_VERSION(문자열) - (선택 사항) 읽기에 사용됩니다. 사용할 SAP OData Rest API 버전. 예: 2.0.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.

  예: SalesOrder
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.

  예: `SalesOrder = "10"`

- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

  예: SELECT * FROM /sap/opu/odata/sap/API_SALES_ORDER_SRV/A_SalesOrder
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.

  예: ValidStartDate
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).

  예: "2000-01-01T00:00:00.000Z"
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).

  예: "2024-01-01T00:00:00.000Z"
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - SAP 인스턴스 애플리케이션 호스트 URL.

  예: https://example-externaldata.sierra.aws.dev
- SERVICE_PATH(문자열) - SAP 인스턴스 애플리케이션 서비스 경로.

  예: /sap/opu/odata/iwfnd/catalogservice;v=2
- CLIENT_NUMBER(문자열) - SAP 인스턴스 애플리케이션 클라이언트 번호.

  예: 100
- PORT_NUMBER(문자열) - 기본값: SAP 인스턴스 애플리케이션 포트 번호.

  예: 443
- LOGON_LANGUAGE(문자열) - SAP 인스턴스 애플리케이션 로그온 언어.

  예: EN
- ENABLE_CDC(문자열) - CDC가 활성화된 상태(즉, 트랙 변경 사항 포함)에서 작업 실행 여부를 정의합니다.

  예: True/False
- DELTA_TOKEN(문자열) - 제공된 유효한 델타 토큰을 기반으로 증분 데이터 풀을 실행합니다.

  예: D20241107043437_000463000
- PAGE_SIZE(정수) - 레코드 쿼리를 위한 페이지 크기를 정의합니다. 기본 페이지 크기는 50,000입니다. 페이지 크기를 지정하면 SAP는 전체 데이터세트가 아닌 API 직접 호출당 정의된 수의 레코드만

반환합니다. 커넥터는 여전히 총 레코드 수를 제공하고 지정된 페이지 크기를 사용하여 페이지 매김을 처리합니다. 더 큰 페이지 크기가 필요한 경우 허용되는 최댓값인 500,000까지 선택할 수 있습니다. 지정된 페이지 크기가 500,000을 초과하면 무시됩니다. 대신 시스템은 허용되는 최대 페이지 크기를 사용합니다. 연결 옵션 `PAGE_SIZE`를 원하는 값으로 추가하여 AWS Glue Studio UI에서 페이지 크기를 지정할 수 있습니다.

예: `20000`

SAP OData 엔터티 및 필드 세부 정보

개체	데이터 유형	지원되는 연산자
테이블(동적 엔터티)	String	=, !=, >, >=, <, <=, BETWEEN, LIKE
	Integer	=, !=, >, >=, <, <=, BETWEEN, LIKE
	Long	=, !=, >, >=, <, <=, BETWEEN, LIKE
	배정밀도 실수	=, !=, >, >=, <, <=, BETWEEN, LIKE
	날짜	=, !=, >, >=, <, <=, BETWEEN, LIKE
	DateTime	=, !=, >, >=, <, <=, BETWEEN, LIKE
	불	=, !=
	Struct	=, !=, <, <=, >, >=, BETWEEN, LIKE

SendGrid에 연결

SendGrid는 트랜잭션 및 마케팅 이메일을 위한 고객 커뮤니케이션 플랫폼입니다.

- SendGrid 커넥터는 연락처 목록을 생성 및 관리하고 이메일 마케팅 캠페인을 생성하는 데 도움이 됩니다.
- SendGrid를 사용하면 온라인 비즈니스, 비영리 단체 및 기타 온라인 단체가 마케팅 이메일을 작성하여 대규모 대상에게 전송하고 해당 이메일의 참여도를 모니터링할 수 있습니다.

주제

- AWS Glue의 SendGrid 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- SendGrid 구성
- SendGrid 연결 구성
- SendGrid 엔터티에서 읽기
- SendGrid 연결 옵션
- SendGrid 제한 사항

AWS Glue의 SendGrid 지원

AWS Glue에서는 다음과 같이 SendGrid를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 SendGrid에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 SendGrid API 버전

다음 SendGrid API 버전이 지원됩니다.

- v3

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

SendGrid 구성

AWS Glue를 사용하여 SendGrid에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- API 키가 있는 SendGrid 계정이 있습니다.
- SendGrid 계정에 유효한 라이선스가 있는 API 액세스 권한이 있습니다.

이러한 요구 사항을 충족하면 SendGrid 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 SendGrid에서 다른 작업을 수행하지 않아도 됩니다.

SendGrid 연결 구성

SendGrid는 사용자 지정 인증을 지원합니다.

사용자 지정 인증에 필요한 API 키 생성에 대한 퍼블릭 SendGrid 설명서는 Authentication을 참조하세요.

SendGrid 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 *api_key*과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 SendGrid를 선택합니다.
   b. 연결하려는 SendGrid 인스턴스의 `INSTANCE_URL` 항목을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

SendGrid 엔터티에서 읽기

사전 조건

읽으려는 SendGrid 객체입니다. 객체 이름(예: `lists`, `singlesends` 또는 `segments`)이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Lists	아니요	예	아니요	예	아니요
Single Sends	예	예	아니요	예	아니요
Marketing Campaign Stats-Automations	예	예	아니요	예	아니요
Marketing Campaign Stats-Single Sends	예	예	아니요	예	아니요
Segments	예	아니요	아니요	예	아니요
Contacts	예	아니요	아니요	예	아니요
Category	아니요	아니요	아니요	예	아니요
Stats	예	아니요	아니요	예	아니요
Unsubscribe Groups	예	아니요	아니요	예	아니요

예시:

```
sendgrid_read = glueContext.create_dynamic_frame.from_options(
    connection_type="sendgrid",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "lists",
        "API_VERSION": "v3",
        "INSTANCE_URL": "instanceUrl"
    }
```

SendGrid 엔터티 및 필드 세부 정보:

정적 메타데이터를 포함하는 엔터티:

엔터티	필드	데이터 유형	지원되는 연산자
Lists	id	String	N/A
	name	String	N/A
	contact_count	Integer	N/A
	_metadata	Struct	N/A
Single Sends	id	String	N/A
	name	String	EQUAL_TO
	abtest	Struct	N/A
	status	String	EQUAL_TO
	categories	List	EQUAL_TO
	send_at	String	N/A
	is_abtest	Boolean	N/A
	updated_at	String	N/A
	created_at	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자	
	channels	List	N/A	
Marketing Campaign Stats-Automations	id	String	N/A	
	aggregation	String	N/A	
	step_id	String	N/A	
	stats	Struct	N/A	
	automation_ids	List	EQUAL_TO	
Marketing Campaign Stats-Singlesends	id	String	N/A	
	ab_variation	String	N/A	
	ab_phase	String	N/A	
	aggregation	String	N/A	
	stats	Struct	N/A	
	singlesend_ids	List	EQUAL_TO	
Segments	id	String	N/A	
	name	String	N/A	
	query_version	String	N/A	
	contacts_count	Integer	N/A	
	sample_updated_at	String	N/A	
	next_sample_update	String	N/A	
	created_at	String	N/A	

엔터티	필드	데이터 유형	지원되는 연산자	
	updated_at	String	N/A	
	parent_list_id	String	N/A	
	status	Struct	N/A	
	parent_list_ids	String	EQUAL_TO	
	no_parent_list_id	Boolean	EQUAL_TO	
Contacts	id	String	N/A	
	first_name	String	N/A	
	last_name	String	N/A	
	unique_name	String	N/A	
	email	String	N/A	
	alternate_emails	List	N/A	
	address_line_1	String	N/A	
	address_line_2	String	N/A	
	city	String	N/A	
	state_province_region	String	N/A	
	country	String	N/A	
	postal_code	String	N/A	
	phone_number	String	N/A	
	whatsapp	String	N/A	
	line	String	N/A	

엔터티	필드	데이터 유형	지원되는 연산자	
	facebook	String	N/A	
	list_ids	List	N/A	
	custom_fields	Struct	N/A	
	created_at	String	N/A	
	updated_at	String	N/A	
	_metadata	Struct	N/A	
	event_timestamp	DateTime	BETWEEN	
Category	categories	List	N/A	
Stats	date	String	N/A	
	stats	List	N/A	
	start_date	DateTime	EQUAL_TO, BETWEEN	
	aggregated_by	String	EQUAL_TO	
Unsubscribe Groups	id	Integer	EQUAL_TO	
	name	String	N/A	
	description	String	N/A	
	last_emai l_sent_at	Integer	N/A	
	is_default	Boolean	N/A	
	unsubscribes	Integer	N/A	

> **Note**
>
> 커넥터의 응답에서 Struct 및 List 데이터 유형은 String 데이터 유형으로 변환되며, DateTime 데이터 유형은 타임스탬프로 변환됩니다.

분할 쿼리

SendGrid는 필터 기반 분할 또는 레코드 기반 분할을 지원하지 않습니다.

SendGrid 연결 옵션

다음은 SendGrid의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. SendGrid에서의 객체 이름입니다.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용할 SendGrid Rest API 버전입니다.
- `INSTANCE_URL`(문자열) - (필수) 읽기에 사용됩니다. 유효한 SendGrid 인스턴스 URL입니다.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

SendGrid 제한 사항

다음은 SendGrid의 제한 사항 또는 참고 사항입니다.

- 증분 풀은 `start_date` 필드의 Stats 엔터티와 `event_timestamp` 필드의 Contact 엔터티에서만 지원됩니다.
- 페이지 매김은 Marketing Campaign Stats(Automations), Marketing Campaign Stats(Single Sends), Single Sends 및 Lists 엔터티에서만 지원됩니다.
- Stats 엔터티의 경우 `start_date`는 필수 필터 파라미터입니다.
- 액세스가 제한된 API 키는 이메일 API 및 Stats 엔터티에 대한 읽기 액세스를 지원할 수 없습니다. 전체 액세스 권한이 있는 API 키를 사용합니다. 자세한 내용은 API 개요 섹션을 참조하세요.

에 연결 ServiceNow

ServiceNow 는 IT 관리 워크플로를 자동화하기 위한 클라우드 기반 SaaS 플랫폼입니다. 이 ServiceNow 플랫폼은 다른 도구와 쉽게 통합되어 사용자가 다양한 앱 및 플러그인을 사용하여 프로젝트, 팀 및 고객 상호 작용을 관리할 수 있습니다. ServiceNow 사용자는 ServiceNow 계정에 AWS Glue 연결할 수 있습니다. 그런 다음 를 ETL 작업의 데이터 소스 ServiceNow 로 사용할 수 있습니다. 이러한 작업을 실행하여 ServiceNow 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue 에 대한 지원 ServiceNow
- 연결 생성 및 사용 API 작업이 포함된 정책
- 구성 ServiceNow
- ServiceNow 연결 구성
- ServiceNow 엔터티에서 읽기
- ServiceNow 연결 옵션
- ServiceNow 커넥터의 제한 사항 및 참고 사항

AWS Glue 에 대한 지원 ServiceNow

AWS Glue 는 다음과 ServiceNow 같이 지원합니다.

소스로 지원되나요?

예. 작업을 사용하여 AWS Glue ETL 에서 데이터를 쿼리할 수 있습니다 ServiceNow.

대상으로서 지원되나요?

아니요.

지원되는 ServiceNow API 버전

지원되는 ServiceNow API 버전은 다음과 같습니다.

- v2

버전별 엔터티 지원은 지원되는 소스 엔터티를 참조하세요.

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려면 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS 관리 콘솔을 사용할 때 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

구성 ServiceNow

AWS Glue 를 사용하여 에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 ServiceNow합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- 이메일과 암호가 있는 ServiceNow 계정이 있습니다. 자세한 내용은 ServiceNow 계정 생성 단원을 참조하십시오.
- ServiceNow 계정에 API 액세스할 수 있습니다. 의 ServiceNow API 모든 사용은 추가 비용 없이 사용할 수 있습니다.

이러한 요구 사항을 충족하면 ServiceNow 계정에 AWS Glue 연결할 준비가 된 것입니다.

ServiceNow 계정 생성

ServiceNow 계정을 생성하려면:

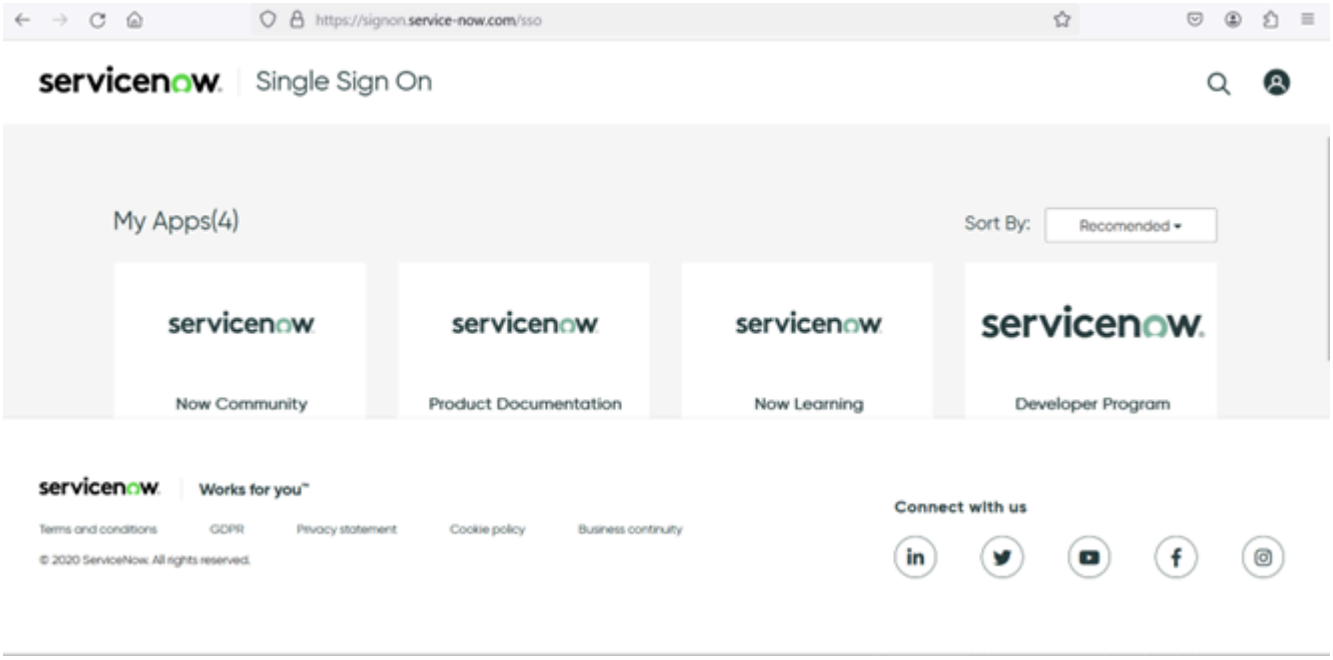
1. servicenow.com 가입 페이지로 이동하여 세부 정보를 입력하고 계속을 클릭합니다.
2. 등기 우편으로 확인 코드를 받으면 해당 코드를 입력하고 확인을 선택합니다.
3. 다단계 인증을 설정하거나 건너뜁니다.

계정이 생성되고 프로필이 ServiceNow 표시됩니다.

ServiceNow 개발자 인스턴스 생성

에 로그인한 후 개발자 인스턴스를 요청합니다 ServiceNow.

1. ServiceNow 로그인 페이지에서 계정 자격 증명을 입력합니다.
2. ServiceNow 개발자 프로그램 을 선택합니다.



3. 오른쪽 상단에서 인스턴스 요청을 선택합니다.
4. 직무 책임을 입력합니다. 사용 약관에 대한 동의를 표시하고 설정 완료를 선택합니다.
5. 인스턴스가 생성되면 인스턴스URL와 보안 인증 정보를 기록해 둡니다.

BasicAuth 보안 인증 정보 검색

무료 계정의 Basic Auth 보안 인증 정보를 검색하려면:

1. ServiceNow 로그인 페이지에 계정 보안 인증 정보를 입력합니다.
2. 홈 페이지에서 프로필 편집 섹션(오른쪽 상단 모서리)을 선택하고 인스턴스 암호 관리 를 선택합니다.
3. 사용자 이름, 암호 및 인스턴스와 같은 로그인 자격 증명을 검색합니다URL.

> **Note**
>
> 계정에 MFA가 활성화된 경우 기본 인증: <username>:<password><MFA Token>의 사용자 암호 MFA 끝에 토큰을 추가합니다.

자세한 내용은 ServiceNow 설명서의 애플리케이션 구축을 참조하세요.

OAuth 2.0 보안 인증 생성

ServiceNow 커넥터에서 OAuth2.0을 사용하려면 인바운드 클라이언트를 생성해야 함)을 사용하여 클라이언트 ID 및 클라이언트 보안 암호를 생성해야 합니다.

1. ServiceNow 로그인 페이지에서 계정 보안 인증 정보를 입력합니다.
2. 홈 페이지에서 구축 시작을 선택합니다.
3. App Engine Studio 페이지에서 Application Registry 를 검색합니다.
4. 오른쪽 상단에서 새로 만들기를 선택합니다.
5. 외부 클라이언트용 OAuth API 엔드포인트 생성 옵션을 선택합니다.
6. OAuth 구성을 필요한 대로 변경하고 업데이트를 선택합니다.

   리디렉션 URL: https://us-east-1.console.aws.amazon.com/gluestudio/oauth 예제
7. 새로 생성된 OAuth 클라이언트 앱을 선택하여 클라이언트 ID 및 클라이언트 보안 암호를 검색합니다.

8. 추가 처리를 위해 클라이언트 ID와 클라이언트 보안 암호를 저장합니다.

비프로덕션 개발자 계정OAuth에서 를 구성하려면:

1. ServiceNow 설명서의 인증 프로필 생성 주제를 사용하여 인증 프로필을 생성합니다.
2. 에 대한 인증 프로필에서 OAuth 유형 을 OAuth 선택하고 위에서 생성한 인바운드 클라이언트를 선택하여 OAuth 엔터티 를 설정합니다.
3. 클라이언트가 여러 개 있는 경우 여러 인증 프로파일을 생성하여 인증 프로파일에 필요한 OAuth엔터티를 설정해야 합니다.
4. 구성되지 않은 경우 REST API 액세스 정책을 생성하여 TABLE 에 대한 액세스 권한을 부여합니다 API. REST API 액세스 정책 생성을 참조하세요.

ServiceNow 연결 구성

권한 부여 유형은 AWS Glue에서 ServiceNow과 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다. 선택한 항목은 연결을 생성하기 전에 충족해야 하는 요구 사항에 영향을 미칩니다. ServiceNow에서는 OAuth 2.0에 대한 AUTHORIZATION_CODE 권한 부여 유형만 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 ServiceNow로 리디렉션합니다. 사용자가 로그인하고 ServiceNow 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 ServiceNow에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 ServiceNow로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 ServiceNow 설명서는 OAuth 설정을 참조하세요.

ServiceNow 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.

a. 기본 인증을 위해 보안 암호에는 USERNAME 및 PASSWORD를 키로 사용하여 연결된 앱 소비자 보안 암호가 포함되어야 합니다.

b. 인증 코드 권한 부여 유형의 경우 보안 암호에는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호가 포함되어야 합니다.

c. 참고: AWS Glue에서 연결당 시크릿을 생성해야 합니다.

2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.

a. 연결 유형을 선택할 때는 ServiceNow를 선택합니다.

b. 연결하려는 ServiceNow 인스턴스의 INSTANCE_URL을 제공합니다.

c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. AWS Glue에서 연결에 사용할 인증 유형을 선택하세요.

i. 기본 인증: 이 인증 유형은 ServiceNow 인스턴스에서 특정 사용자의 권한을 통해 사용자 이름 및 암호를 미리 사용할 수 있으므로 자동화 시나리오에 적합합니다. AWS Glue에서는 사용자 이름 및 암호를 사용하여 ServiceNow API를 인증할 수 있습니다. 기본 인증: Username 및 Password의 경우에만 다음 입력을 제공합니다.

ii. OAuth2: OAuth2: ClientId, ClientSecret, Authorization URL, Authorization Token URL의 경우에만 다음 입력을 제공합니다.

e. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 secretName을 선택합니다.

f. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

ServiceNow 엔터티에서 읽기

사전 조건

읽으려는 ServiceNow Tables 객체. 객체 이름(예: pa_bucket 또는 incident)이 필요합니다.

예시:

```
servicenow_read = glueContext.create_dynamic_frame.from_options(
    connection_type="servicenow",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "pa_buckets",
        "API_VERSION": "v2"
        "instanceUrl": "https://<instance-name>.service-now.com"
    }
)
```

ServiceNow 엔터티 및 필드 세부 정보:

다음 엔터티에 대해 ServiceNow에서는 메타데이터를 동적으로 가져오도록 엔드포인트를 제공하므로 운영자 지원은 각 엔터티의 데이터 유형 수준에서 캡처됩니다.

개체	데이터 유형	지원되는 연산자
테이블(동적 엔터티)	Integer	=, !=, <, <=, >, >=, BETWEEN
	BigDecimal	=, !=, <, <=, >, >=, BETWEEN
	Float	=, !=, <, <=, >, >=, BETWEEN
	Long	=, !=, <, <=, >, >=, BETWEEN
	날짜	=, !=, <, <=, >, >=, BETWEEN
	DateTime	=, !=, <, <=, >, >=, BETWEEN

개체	데이터 유형	지원되는 연산자
	불	=, !=
	String	=, !=, <, <=, >, >=, BETWEEN, LIKE
	구조체	N/A

> **Note**
>
> 구조체 데이터 유형은 커넥터의 응답에서 문자열 데이터 유형으로 변환됩니다.

> **Note**
>
> DML_STATUS는 CREATED/UPDATED 레코드를 추적하는 데 사용되는 추가 사용자 정의 속성입니다.

분할 쿼리

필드 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

개체 이름입니다.	분할 필드	데이터 유형
동적 엔터티	sys_mod_count	Integer
	sys_created_on, sys_updated_on	DateTime

- PARTITION_FIELD: 쿼리를 파티셔닝하는 데 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

Datetime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

유효한 값의 예제:

```
"2024-01-30T06:47:51.000Z"
```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

다음 표에서는 엔터티 분할 필드 지원 세부 정보를 설명합니다.

예시:

```
servicenow_read = glueContext.create_dynamic_frame.from_options(
    connection_type="servicenow",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "pa_buckets",
        "API_VERSION": "v2",
        "instanceUrl": "https://<instance-name>.service-now.com"
        "PARTITION_FIELD": "sys_created_on"
        "LOWER_BOUND": "2024-01-30T06:47:51.000Z"
        "UPPER_BOUND": "2024-06-30T06:47:51.000Z"
        "NUM_PARTITIONS": "10"
    }
```

레코드 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(NUM_PARTITIONS)을 제공할 수 있습니다. 이 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

레코드 기반 분할에서는 존재하는 총 레코드 수를 ServiceNow API에서 쿼리하고 제공된 NUM_PARTITIONS 수로 나눕니다. 그런 다음, 결과 레코드 수를 각 하위 쿼리에서 동시에 가져옵니다.

- NUM_PARTITIONS: 파티션 수.

예시:

```
servicenow_read = glueContext.create_dynamic_frame.from_options(
```

```
    connection_type="servicenow",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "pa_buckets",
        "API_VERSION": "v2",
        "instanceUrl": "https://<instance-name>.service-now.com"
        "NUM_PARTITIONS": "2"
    }
```

ServiceNow 연결 옵션

다음은 에 대한 연결 옵션입니다 ServiceNow.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. 의 객체 이름입니다 ServiceNow.
- `API_VERSION`(문자열) - (필수) 사용하려는 Read. ServiceNow Rest API 버전에 사용됩니다. 예: v1,v2,v3,v4.
- `SELECTED_FIELDS`(목록<문자열>) - 기본값: 비어 있음(SELECT*). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- `PARTITION_FIELD`(문자열) - 읽기에 사용됩니다. 쿼리를 파티셔닝하는 데 사용할 필드.
- `LOWER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함). 예: 2024-01-30T06:47:51.000Z.
- `UPPER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외). 예: 2024-06-30T06:47:51.000Z.
- `NUM_PARTITIONS`(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수. 예: 10.
- `INSTANCE_URL`(문자열) - (필수) https://<instance-name>.service-now.com 형식URL의 유효한 ServiceNow 인스턴스입니다.

ServiceNow 커넥터의 제한 사항 및 참고 사항

다음은 ServiceNow 커넥터의 제한 사항 또는 참고 사항입니다.

- SaaS 설명서에 따라 `sys_created_on`, `sys_updated_on`, `sys_mod_count`는 시스템에서 생성된 필드입니다. 커넥터는 SaaS API를 사용하여 응답 본문에 이 필드를 제공합니다.

  - SaaS가 개체에 대해 이 필드를 생성하지 않으면 필터 기반 파티셔닝을 지원할 수 없습니다.
- SaaS API가 응답에서 `sys_created_on` 및 `sys_updated_on` 필드를 반환하지 않으면 `DML_STATUS`를 계산할 수 없습니다.

AWS Glue Studio에서 Slack에 연결

Slack은 사용자가 다양한 공개 및 비공개 채널을 통해 메시지와 첨부 파일을 보낼 수 있는 엔터프라이즈 커뮤니케이션 앱입니다. Slack 사용자인 경우 Slack 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, Slack을 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Slack 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- Slack에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Slack 구성
- Slack 연결 구성
- Slack 엔터티에서 읽기
- Slack 연결 옵션
- 제한 사항
- 새 Slack 계정 생성 및 클라이언트 앱 구성

Slack에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Slack을 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Slack에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Slack API 버전

Slack API v2.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, Amazon CloudWatch Logs, Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Slack 구성

AWS Glue를 사용하여 Slack에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- Slack 계정이 있어야 합니다. 자세한 내용은 새 Slack 계정 생성 및 클라이언트 앱 구성 단원을 참조하십시오.

이러한 요구 사항을 충족하면 Slack 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Slack 연결 구성

Slack에서는 OAuth 2에 대한 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Slack으로 리디렉션합니다. 사용자가 로그인하고 Slack 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Slack에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Slack으로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 생성 후 1시간 후에 만료됩니다. 새로 고침 토큰을 사용하여 새 액세스 토큰을 가져올 수 있습니다.

권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 자세한 내용은 Slack API를 참조하세요.

Slack 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 연결에 대한 보안 암호를 생성해야 합니다.
   a. 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Slack을 선택하세요.
   b. Slack 환경을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
```

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Slack 엔터티에서 읽기

사전 조건

- 읽으려는 Slack 객체.

지원되는 엔터티

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
대화	예	예	아니요	예	예

예제

```
slack_read = glueContext.create_dynamic_frame.from_options(
    connection_type="slack",
    connection_options={
```

```
        "connectionName": "connectionName",
        "ENTITY_NAME": "conversations/C058W38R5J8"
    }
)
```

Slack 엔터티 및 필드 세부 정보

개체	필드	데이터 형식	지원되는 연산자
대화	attachments	나열	NA
대화	bot_id	String	NA
대화	블록	나열	NA
대화	client_msg_id	String	NA
대화	is_starred	불	NA
대화	last_read	String	NA
대화	latest_reply	String	NA
대화	reactions	나열	NA
대화	replies	나열	NA
대화	reply_count	Integer	NA
대화	reply_users	나열	NA
대화	reply_users_count	Integer	NA
대화	subscribed	불	NA
대화	subtype	String	NA
대화	텍스트	String	NA
대화	team	String	NA
대화	thread_ts	String	NA

개체	필드	데이터 형식	지원되는 연산자
대화	ts	String	EQUAL_TO, BETWEEN, LESS_THAN , LESS_THAN _OR_EQUAL_TO, GREATER_T HAN, GREATER_T HAN_OR_EQUAL_TO
대화	type	String	NA
대화	사용자	String	NA
대화	inviter	String	NA
대화	root	구조체	NA
대화	is_locked	불	NA
대화	files	나열	NA
대화	room	구조체	NA
대화	업로드	불	NA
대화	display_as_bot	불	NA
대화	채널	String	NA
대화	no_notifications	불	NA
대화	permalink	String	NA
대화	pinned_to	나열	NA
대화	pinned_info	구조체	NA
대화	edited	구조체	NA

개체	필드	데이터 형식	지원되는 연산자
대화	app_id	String	NA
대화	bot_profile	구조체	NA
대화	metadata	구조체	NA

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-07-01T00:00:00.000Z".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준 분할 필드 지원 세부 정보는 아래 표에 캡처되어 있습니다.

Entity Name	분할 필드	데이터 형식
대화	ts	String

예제

```
slack_read = glueContext.create_dynamic_frame.from_options(
    connection_type="slack",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "conversations/C058W38R5J8",
        "PARTITION_FIELD": "ts"
        "LOWER_BOUND": "2022-12-01T00:00:00.000Z"
```

```
        "UPPER_BOUND": "2024-09-23T15:00:00.000Z"
        "NUM_PARTITIONS": "2"
    }
)
```

Slack 연결 옵션

다음은 Slack에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. 지원되는 엔터티 이름. 예: conversations/C058W38R5J8.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 필드.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

제한 사항

다음은 Slack 커넥터에 대한 제한 사항입니다.

- 커넥터는 주어진 대화에서 사용할 수 있는 총 레코드(메시지) 수를 가져올 수 있는 수단을 제공하지 않으므로 레코드 기반 분할은 지원되지 않습니다.

새 Slack 계정 생성 및 클라이언트 앱 구성

Slack 계정 생성

1. Slack 홈 페이지를 열어 계정에 가입하세요.
2. SIGN UP WITH EMAIL ADDRESS를 선택하세요. 이메일 ID를 입력하고 계속을 선택하세요.
3. 이메일 주소로 전송된 6자리 코드를 입력하세요. 그러면 워크스페이스를 생성하거나 기존 워크스페이스에 가입하도록 리디렉션됩니다.
4. Create a workspace를 선택하여 새 워크스페이스를 생성하세요. 설정 프로세스의 일환으로 몇 가지 질문에 답하도록 리디렉션됩니다.
    - 회사 이름

- 이름.
- 이메일로 동료를 추가하는 방법
- 함께 작업하는 팀? (채널 이름이 됩니다.)

5. 이러한 질문에 대한 입력 필드를 채우고 계속하세요. 이제 계정을 사용할 준비가 되었습니다.

Slack 개발자 앱 생성

1. Slack 계정에 로그인하고 Slack 워크스페이스에 로그인하세요.
2. 워크스페이스 메뉴에서 Tools and settings를 선택하고 Manage apps를 선택하세요.
3. Slack App Directory 메뉴에서 Build를 선택하세요.
4. Your Apps 페이지에서 Create an App을 선택하세요.
5. Create an app 페이지에서 From scratch를 선택하세요.
6. 열리는 Name app & choose workspace 대화 상자에서 앱 이름을 추가하고 Pick a workspace to deploy your app in을 선택하세요. 그런 다음, Create App을 선택하세요.
7. 앱 자격 증명에 표시된 클라이언트 ID 및 보안 암호를 기록하세요.
8. OAuth & Permissions 사이드바에서 Scopes로 이동하고 Add an OAuth Scope를 선택하세요. 구성을 위해 리디렉션 URL을 앱에 추가하여 'Add to Slack' 버튼을 자동으로 생성하거나 앱을 배포할 수 있습니다. 리디렉션 URL 섹션까지 스크롤하여 Add New Redirect URL을 선택하고 저장하세요.
9. 그런 다음, OAuth Tokens for Your Workspace 섹션으로 스크롤하고 Install to Workspace를 선택하세요.
10. 생성한 앱에서 연결하려는 Slack 워크스페이스에 액세스할 수 있는 권한을 요청하고 있음을 알리는 대화 상자가 열리면 Allow을 선택하세요.
11. 성공적으로 완료되면 콘솔에 OAuth Tokens for Your Workspace 화면이 표시됩니다.
12. OAuth Tokens for Your Workspace 화면에서 AWS Glue에 연결하는 데 사용할 OAuth 토큰을 복사하고 저장하세요.
13. 다음으로, Slack 팀 ID를 검색하세요. Slack 워크스페이스 메뉴에서 Tools and settings를 선택하고 Manage apps를 선택하세요. 열리는 페이지의 URL에서 팀 ID를 찾을 수 있습니다.
14. 앱의 퍼블릭 배포를 위해 사이드바의 Manage Distribution 버튼으로 이동하여 활성화할 수 있습니다. 아래로 스크롤하여 Share Your App with Other Workspaces 섹션으로 이동하고 Remove Hard Coded Information을 선택하세요. 동의를 제공하고 Active Public Distribution을 선택하세요.

15. 이제 앱의 퍼블릭 배포가 수행됩니다. 엔터티 API에 액세스하려면 사용자가 액세스하려는 모든 워크스페이스 채널에 앱을 추가해야 합니다.
16. Slack 계정에 로그인하고 채널에 액세스해야 하는 워크스페이스를 여세요.
17. 워크스페이스에서 앱이 액세스하려는 채널을 열고 채널 제목을 선택하세요. 팝업에서 Integrations 탭을 선택하고 앱을 추가하세요. 그러면 앱이 채널과 통합되어 API에 액세스할 수 있습니다.

    OAuth 2.0 클라이언트 ID에는 하나 이상의 승인된 리디렉션 URL이 포함되어야 합니다. 리디렉션 URL 형식은 다음과 같습니다.

> **Note**
>
> Appflow 리디렉션 URL은 변경 가능하며, AWS Glue 플랫폼의 사후 리디렉션 URL을 사용할 수 있습니다. 클라이언트 ID 및 클라이언트 보안 암호는 OAuth 2.0 클라이언트 ID의 설정에서 가져옵니다.

리디렉션 URL은 다음 중 하나일 수 있습니다.	
감마 환경에 대한 URL 리디렉션	
https://us-west-2.console.aws.amazon.com/appflow/oauth	https://us-east-1.awsc-integ.aws.amazon.com/appflow/oauth
https://us-east-2.console.aws.amazon.com/appflow/oauth	
https://us-west-1.console.aws.amazon.com/appflow/oauth	
https://ap-south-1.console.aws.amazon.com/appflow/oauth	
https://ap-southeast-1.console.aws.amazon.com/appflow/oauth	
https://ap-southeast-2.console.aws.amazon.com/appflow/oauth	

리디렉션 URL은 다음 중 하나일 수 있습니다.	
감마 환경에 대한 URL 리디렉션	
https://ap-northeast-1.console.aws.amazon.com/appflow/oauth	
https://ap-northeast-2.console.aws.amazon.com/appflow/oauth	
https://ca-central-1.console.aws.amazon.com/appflow/oauth	
https://eu-central-1.console.aws.amazon.com/appflow/oauth	
https://eu.west-1.console.aws.amazon.com/appflow/oauth	
https://eu-west-2.console.aws.amazon.com/appflow/oauth	
https://eu-west-3.console.aws.amazon.com/appflow/oauth	
https://sa-east-1.console.aws.amazon.com/appflow/oauth	
https://us-west-2.awsc-integ.aws.amazon.com/appflow/oauth	
https://af-south-1.console.aws.amazon.com/appflow/oauth	

Smartsheet에 연결

Smartsheet는 작업 관리 및 공동 작업 SaaS 제품입니다. 기본적으로 Smartsheet를 사용하면 사용자가 스프레드시트와 같은 객체를 사용하여 비즈니스 데이터를 생성, 저장 및 활용할 수 있습니다.

주제

- AWS Glue의 Smartsheet 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Smartsheet 구성
- Smartsheet 연결 구성
- Smartsheet 엔터티에서 읽기
- Smartsheet 연결 옵션
- Smartsheet 계정 생성
- 제한 사항

AWS Glue의 Smartsheet 지원

AWS Glue에서는 다음과 같이 Smartsheet를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Smartsheet에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Smartsheet API 버전

v2.0

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
```

```
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

이전 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess-정책이 연결된 자격 증명이 AWS Management 콘솔을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Smartsheet 구성

AWS Glue를 사용하여 Smartsheet에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일과 암호를 사용하는 Smartsheet 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Smartsheet 계정 생성을 참조하세요.
- Smartsheet 계정에는 유효한 라이선스가 있는 API 액세스 권한이 있습니다.
- Smartsheet 계정에는 `Sheets` 엔터티에 대한 Pro 요금제와 `Events` 엔터티에 대한 이벤트 보고 추가 기능이 포함된 엔터프라이즈 요금제가 있습니다.

이러한 요구 사항을 충족하면 Smartsheet 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Smartsheet에서 다른 작업을 수행하지 않아도 됩니다.

Smartsheet 연결 구성

Smartsheet는 OAuth2에 대한 `AUTHORIZATION_CODE` 권한 부여 유형을 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' `OAuth`로 간주됩니다. 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Smartsheet에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Smartsheet으로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.

AUTHORIZATION_CODE OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Smartsheet 설명서는 Smartsheet API를 참조하세요.

스마트시트 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.

   고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.

   > **Note**
   >
   > AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

2. AWS Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성합니다.
   a. 연결 유형을 선택할 때 Smartsheet를 선택합니다.
   b. 연결하려는 Smartsheet의 `instanceUrl` 항목을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
```

```
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Smartsheet 엔터티에서 읽기

사전 조건

읽으려는 `Smartsheet` 객체입니다. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
List Sheet	예	예	아니요	예	아니요
Row Metadata	예	예	아니요	예	아니요
Sheet Metadata	아니요	아니요	아니요	예	아니요
Sheet Data	예	예	예	예	아니요
Event	예	예	아니요	예	아니요

예

```
Smartsheet_read = glueContext.create_dynamic_frame.from_options(
    connection_type="smartsheet",
    connection_options={
```

```
        "connectionName": "connectionName",
        "ENTITY_NAME": "list-sheets",
        "API_VERSION": "2.0",
        "INSTANCE_URL": "https://api.smartsheet.com"
    })
```

Smartsheet 엔터티 및 필드 세부 정보

엔터티	필드	데이터 형식	지원되는 연산자
List Sheets	id	Long	NA
	accessLevel	String	NA
	createdAt	DateTime	NA
	modifiedAt	DateTime	NA
	name	String	NA
	permalink	String	NA
	modifiedSince	DateTime	>=
	version	Integer	NA
	source	Struct	NA
Row Metadata	id	Long	NA
	sheetId	Long	NA
	accessLevel	String	NA
	attachments	List	NA
	columns	List	NA
	conditionalFormat	String	NA
	createdAt	DateTime	NA

엔터티	필드	데이터 형식	지원되는 연산자
	createdBy	Struct	NA
	discussions	List	NA
	proofs	Struct	NA
	expanded	Boolean	NA
	filteredOut	Boolean	NA
	format	String	NA
	inCriticalPath	Boolean	NA
	locked	Boolean	NA
	lockedForUser	Boolean	NA
	modifiedAt	DateTime	NA
	modifiedBy	Struct	NA
	permalink	String	NA
	rowNumber	Integer	NA
	version	Integer	NA
	totalRowCount	Integer	NA
	rowsModifiedSince	DateTime	>
	filterId	Long	“="
	siblingId	Long	NA
	parentId	Long	NA
Sheet metadata	id	Long	NA

엔터티	필드	데이터 형식	지원되는 연산자
	fromId	Long	NA
	ownerId	Long	NA
	accessLevel	String	NA
	attachments	List	NA
	columns	List	NA
	createdAt	DateTime	NA
	crossSheetReferences	List	NA
	dependenciesEnabled	Boolean	NA
	discussions	List	NA
	effectiveAttachmentOptions	List	NA
	favorite	Boolean	NA
	ganttEnabled	Boolean	NA
	hasSummaryFields	Boolean	NA
	modifiedAt	DateTime	NA
	name	String	NA
	owner	String	NA
	permalink	String	NA
	projectSettings	Struct	NA
	readOnly	Boolean	NA

엔터티	필드	데이터 형식	지원되는 연산자
	resourceManagementEnabled	Boolean	NA
	showParentRowsForFilters	Boolean	NA
	source	Struct	NA
	summary	Struct	NA
	totalRowCount	Integer	NA
	userPermissions	Struct	NA
	userSettings	Struct	NA
	version	Integer	NA
	WorkSpace	Struct	NA
	filters	List	NA
	ganttConfig	Struct	NA
	resourceManagementType	String	NA
	cellImageUploadEnabled	Boolean	NA
	isMultiPicklistEnabled	Boolean	NA
Events	eventId	String	NA
	objectType	String	NA
	action	String	NA
	objectId	Long	NA

엔터티	필드	데이터 형식	지원되는 연산자
	eventTimestamp	DateTime	NA
	userId	Long	NA
	requestUserId	Long	NA
	accessTokenName	String	NA
	source	String	NA
	additionalDetails	Struct	NA
	since	DateTime	>=

동적 메타데이터를 포함하는 엔터티:

다음 엔터티의 경우, Smartsheet는 메타데이터를 동적으로 가져올 수 있는 엔드포인트를 제공하여 데이터 유형 수준에서 운영자 지원을 캡처할 수 있습니다.

엔터티	데이터 형식	지원되는 연산자
Sheet Data	String	NA
Sheet Data	Long	"="
Sheet Data	Integer	NA
Sheet Data	DateTime	>

예제

```
Smartsheet_read = glueContext.create_dynamic_frame.from_options(
    connection_type="smartsheet",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "list-sheets",
        "API_VERSION": "2.0",
        "INSTANCE_URL": "https://api.smartsheet.com"
```

```
    }
```

Smartsheet 연결 옵션

다음은 Smartsheet의 연결 옵션입니다.

- `ENTITY_NAME`(문자열)-(필수) 읽기/쓰기에 사용됩니다. Smartsheet에서의 객체 이름입니다.
- `API_VERSION`(문자열)-(필수) 읽기/쓰기에 사용됩니다. 사용할 Smartsheet Rest API 버전입니다. 예: v2.0.
- `INSTANCE_URL`(문자열)-(필수) 읽기에 사용됩니다. Smartsheet 인스턴스 URL입니다.
- `SELECTED_FIELDS`(List<String>)-기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열)-기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Smartsheet 계정 생성

1. Smartsheet 가입 페이지에 액세스하여 Smartsheet 계정에 가입합니다.
2. 새로 만들기를 선택하여 새 계정을 만들거나 등록된 Google, Microsoft 또는 Apple 계정을 사용하여 로그인합니다.
3.
4.
5. Smartsheet에서 확인 이메일을 열고 확인 링크를 선택하여 계정을 확인합니다.

   기본적으로 평가판을 구독하게 됩니다.
6. 왼쪽 하단 모서리에서 계정 아이콘을 선택하고 라이선스 추가/업그레이드를 선택하여 요금제를 업그레이드합니다.

   > **Note**
   >
   > 이는 엔터프라이즈 요금제의 부가 기능인 이벤트 보고에 액세스하는 데 필요합니다.
7. 엔터프라이즈 요금제에서 문의를 선택하여 지원팀에 계정 업그레이드를 요청합니다.
8. 지원 요청 양식에서 요금제를 업그레이드하는 데 필요한 세부 정보와 요구 사항을 제공합니다.

이렇게 하면 엔터프라이즈 요금제로 업그레이드가 완료됩니다.

OAuth2.0 자격 증명 생성

1. 개발자 도구에 액세스하기 위해 계정의 요금제를 업그레이드한 후 Smartsheet 개발자에 액세스합니다.

   활성화 이메일을 받게 됩니다.
2. Smartsheet에서 활성화 이메일을 열고 활성화 링크를 선택하여 계정에서 개발자 도구를 활성화합니다.

   개발자 도구를 사용하면 앱을 생성할 수 있습니다.
3. Smartsheet 계정의 홈 페이지를 열고 계정을 선택하여 액세스를 확인합니다.
4. 서비스 목록에서 개발자 도구를 선택하고 개발자 프로필 세부 정보를 입력합니다.
5. 새 앱 생성을 선택합니다.
6. 앱 등록 양식에 다음 세부 정보를 입력합니다.

   - 이름-앱의 이름입니다.
   - 설명-앱 설명입니다.
   - URL-랜딩 페이지의 앱 또는 URL을 시작할 수 있는 URL입니다.
   - 연락처/지원-지원팀의 연락처 정보입니다.
   - 리디렉션 URL - OAuth 2.0 자격 증명을 수신할 애플리케이션 내의 URL(콜백 URL이라고도 함)입니다.
7. 저장을 선택합니다.

   Smartsheet는 앱에 클라이언트 ID와 클라이언트 보안 암호를 할당합니다. 다음 단계에 이러한 값을 기록합니다. 개발자 도구 섹션의 뒷부분에서도 다시 확인할 수 있습니다.

제한 사항

Smartsheet는 필드 기반 또는 레코드 기반 분할을 지원하지 않습니다.

AWS Glue Studio에서 Snapchat Ads에 연결

Snapchat은 원래 Snapchat Inc.인 Snap Inc.에서 개발한 멀티미디어 인스턴트 메시징 앱 및 서비스입니다. Snapchat의 주요 기능 중 하나는 수신자가 사진과 메시지를 짧은 시간 동안만 사용할 수 있고 이

후에는 사용할 수 없게 되는 기능입니다. Snapchat Marketing은 Snapchat 사용자에게 제공하기 위해 비즈니스에서 비용을 지불할 수 있는 게시물입니다.

주제

- Snapchat Ads에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Snapchat Ads 구성
- Snapchat Ads 연결 구성
- Snapchat Ads 엔터티에서 읽기
- Snapchat Ads 연결 옵션
- Snapchat Ad 계정 생성 및 클라이언트 앱 구성
- Snapchat Ads 계정에서 앱 생성

Snapchat Ads에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Snapchat Ads를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Snapchat Ads에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Snapchat Ads API 버전

v1.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책에서는 연결을 생성하고 사용하는 데 필요한 AWS 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, Amazon CloudWatch Logs, Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Snapchat Ads 구성

AWS Glue를 사용하여 Snapchat Ads에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- Snapchat Ads 계정이 있습니다. 계정 생성에 대한 자세한 내용은 Snapchat Ad 계정 생성 및 클라이언트 앱 구성 섹션을 참조하세요.
- Snapchat Ads 계정에서 OAuth2 앱을 생성했습니다. 이 통합에서는 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하는 데 사용하는 자격 증명을 제공합니다. 자세한 내용은 Snapchat Ads 계정에서 앱 생성 단원을 참조하십시오.

이러한 요구 사항을 충족하면 Snapchat Ads 계정에 AWS Glue를 연결할 준비가 된 것입니다. Snapchat Ads에서 연결된 앱은 외부 애플리케이션(예: AWS Glue)이 Snapchat Ads 데이터에 액세스할 수 있도록 권한을 부여하는 프레임워크입니다.

Snapchat Ads 연결 구성

Snapchat Ads에서는 AUTHORIZATION_CODE 권한 부여 유형만 지원합니다.

이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Snapchat Ads 인스턴스 URL을 제외한 OAuth 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱(AWS Glue 관리형 클라이언트 애플리케이션)에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Snapchat Ads로 리디렉션합니다. 사용자가 로그인하고 Snapchat Ads 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.

사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Snapchat Ads에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Snapchat Ads로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.

이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 생성 후 1시간 후에 만료됩니다. 새로 고침 토큰을 사용하여 새 액세스 토큰을 가져올 수 있습니다.

권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 자세한 내용은 Ads API를 참조하세요.

Snapchat Ads 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Snapchat Ads를 선택하세요.
   b. Snapchat Ads 환경을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Snapchat Ads 엔터티에서 읽기

사전 조건

- 읽으려는 Snapchat Ads 객체. 사용 가능한 엔터티를 확인하려면 아래 지원되는 엔터티 테이블을 참조하세요.

지원되는 엔터티

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
조직	아니요	아니요	아니요	예	아니요
광고 계정	아니요	아니요	아니요	예	아니요
크리에이티브	아니요	아니요	아니요	예	아니요
미디어	아니요	아니요	아니요	예	아니요
Campaign	예	아니요	아니요	예	아니요
광고 계정 아래 광고	예	아니요	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
캠페인 아래 광고	아니요	아니요	아니요	예	아니요
광고 스쿼드	예	아니요	아니요	예	아니요
세그먼트	아니요	아니요	아니요	예	아니요

예제

```
snapchatads_read = glueContext.create_dynamic_frame.from_options(
    connection_type="snapchatAds",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "organization",
        "API_VERSION": "v1"
    }
)
```

Snapchat Ads 엔터티 및 필드 세부 정보

Snapchat Ads에서는 선택한 엔터티 아래에서 사용 가능한 필드를 동적으로 로드합니다. 필드의 데이터 유형에 따라 다음 필터 연산자를 지원합니다.

필드 데이터 유형	지원되는 필터 연산자
불	=

분할 쿼리

- 필드 기반 분할: 지원되지 않습니다.
- 레코드 기반 분할: 지원되지 않습니다.

Snapchat Ads 연결 옵션

다음은 Snapchat Ads에 대한 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Snapchat Ads 엔터티의 이름. 예: `campaign`.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용할 Snapchat Ads Rest API 버전. Snapchat Ads에서는 현재 버전 v1만 지원하므로 값은 v1입니다.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 선택한 엔터티에 대해 선택할 열의 쉼표로 구분된 목록.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Snapchat Ad 계정 생성 및 클라이언트 앱 구성

주제

- Snapchat Ads 가입
- Snapchat Ad 계정을 생성하는 단계

Snapchat Ads 가입

Snapchat Ads에 가입하는 방법:

1. Snapchat Ads Manager로 이동하세요. Snapchat이 처음인가요? 옆의 가입을 선택하세요.
2. 계정 만들기 화면에서 프롬프트에 따라 회사 이름, 이메일, 비밀번호 등을 입력합니다. Next(다음)를 선택합니다.
3. 프로필 만들기 화면에서 사용자 이름, 웹사이트(선택 사항) 값을 입력하고 계정 만들기를 선택하세요. 그러면 프로필 편집 화면에서 프로필 사진과 약력을 추가하는 옵션이 제공됩니다. 확인을 선택합니다.
4. 비즈니스 정보 화면에서 국가, 통화, 전화번호, GSTIN 등과 같은 필수 필드를 작성하고 다음을 선택하여 계정 만들기 프로세스를 완료합니다.

Snapchat Ad 계정을 생성하는 단계

Snapchat Ad 계정을 생성하는 방법:

1. Ads Manager에 로그인하세요. 그런 다음, 상단 모서리의 탐색을 클릭하고 Ad Accounts를 선택합니다.

2. + New Ad Account를 선택하세요. 광고주 세부 정보를 입력하세요.

   - 광고주를 대신하여 광고를 구매하는 에이전시인지 여부를 선택하세요. 'Yes'를 선택하는 경우 연령, 성별 또는 우편번호 수준의 타겟팅을 포함할 수 있는 타겟팅 파라미터를 사용하면 광고가 거부될 수 있습니다. 최소 연령 타겟팅은 최대 21세까지 적용될 수 있습니다.
   - 광고 계정에서 주택, 크레딧 또는 고용 광고를 실행할지 여부를 선택하세요. 'Yes'를 선택하는 경우 연령, 성별 또는 우편번호 수준의 타겟팅을 포함할 수 있는 타겟팅 파라미터를 사용하면 광고가 거부될 수 있습니다. 최소 연령 타겟팅은 최대 21세까지 적용될 수 있습니다.
   - 정치 광고에 대한 광고 계정을 사용할지 여부를 선택하세요. 정치 광고를 실행하는 경우 광고 비용을 지불하는 후원 정치 단체 또는 변호 단체를 입력하세요. 정치 단체를 정확하게 입력하지 않으면 광고가 거부될 수 있습니다. 또한 광고를 제출하기 전에 연결된 필수 '정치 광고 검토 양식'을 작성해야 합니다.

3. Account Details를 선택하고 광고 계정 정보를 입력하세요.

필드	설명
명칭	광고 계정 이름.
계정 유형	자동으로 채워지지 않은 경우 계정 유형을 선택하세요.
결제 유형	펀딩 소스에 지속적으로 액세스하려면 'Revolving'을 선택하세요. 이를 통해 크레딧 라인을 보충할 수 있으며, Ads Manager에서 광고를 실행하도록 선택되기도 합니다. 지출 한도가 있는 일회성 캠페인을 실행하려면 Insertion Order를 선택할 수 있습니다. 삽입 주문(Insertion Order)과 함께 계정을 생성했지만 나중에 지출 한도를 설정하려면 지출 한도를 0달러로 두면 됩니다. 그러면 계정의 지출 한도는 $0이지만 나중에 적용할 수 있습니다.
광고 조직	광고를 구매하는 조직.
결제 센터	청구서를 수신할 결제 센터를 선택하세요. Business Help Center의 단계를 수행하여 비

필드	설명
	즈니스에 처음 청구하거나 다른 결제 센터를 추가하는 경우 자동으로 생성된 항목을 사용할 수 있습니다.
통화	통화를 선택하세요.
지출 한도	결제 유형으로 'Insertion Order'를 선택하는 경우 광고 계정 예산을 입력하세요.
시간대	시간대를 선택하세요.

4. Create Account를 선택하세요. 광고 계정이 생성되며 Ads Manager의 Ad Accounts 부분에서 찾을 수 있습니다. 광고 실행을 시작하려면 결제 방법을 입력해야 합니다. 광고 계정에 멤버를 추가할 수도 있습니다.
5. 기존 결제를 사용할지 아니면 새로 생성할지 선택하세요. 그런 다음, Save Payment Method를 선택하세요.
6. 광고 계정에 추가할 비즈니스에 초대한 멤버를 선택하세요. 할당할 수 있는 역할 및 권한에 대한 자세한 내용은 Roles and Permissions Overview를 참조하세요. 그러면 추가된 멤버가 Ads Manager에 로그인하여 이 광고 계정에 액세스할 수 있습니다. 완료되면 멤버를 저장하세요.

광고 계정에 대한 자세한 내용은 다음을 참조하세요. https://businesshelp.snapchat.com/s/article/roles-permissions?language=en_UShttps://businesshelp.snapchat.com/s/article/roles-permissions?language=en_US

Snapchat Ads 계정에서 앱 생성

Snapchat의 Marketing API에 대한 액세스를 활성화하려면 비즈니스 계정이 설정되어 있어야 합니다. 그런 다음, 아래 단계를 수행하세요.

1. Ads Manager에 로그인하세요. 그런 다음, 왼쪽 상단의 메뉴를 선택하고 Business Dashboard, Business Details를 차례로 선택하세요.
2. +OAuth App을 선택하세요.
3. 앱 이름을 입력하고 Snap 리디렉션 URI로 `https://<aws-region>.console.aws.amazon.com/gluestudio/oauth` URL을 추가하세요. 예를 들어 us-west-1 리전을 사용하는 경우 URL은 `https://us-west-1.console.aws.amazon.com/`

`gluestudio/oauth) and choose` **`Create OAuth App`**입니다. Create OAuth App을 선택하세요.

4. 앱 자격 증명(클라이언트 ID 및 클라이언트 보안 암호)가 표시됩니다. 연결을 생성하는 데 필요하므로 저장하세요.

AWS Glue Studio에서 Snowflake에 연결

> **Note**
>
> AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 Snowflake의 테이블에서 읽고 쓸 수 있습니다. 프로그래밍 방식으로 AWS Glue 작업을 통해 Snowflake를 구성하려면 Redshift 연결 섹션을 참조하세요.

AWS Glue에서는 Snowflake에 대한 기본 제공 지원을 제공합니다. AWS Glue Studio에서는 Snowflake에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스를 제공합니다.

AWS Glue Studio에서 Snowflake용 통합 연결을 생성합니다. 자세한 내용은 고려 사항 단원을 참조하십시오.

주제

- Snowflake 연결 생성
- Snowflake 소스 노드 생성
- Snowflake 대상 노드 생성
- 고급 옵션

Snowflake 연결 생성

> **Note**
>
> 통합 연결(연결 v2)은 모든 연결을 표준화하여 기본 인증 자격 증명에 `PASSWORD` 키인 `USERNAME`을 사용합니다. `sfUser`, `sfPassword`가 포함된 보안 암호로 API를 통해 v1 연결을 생성할 수 있습니다.

AWS Glue Studio에서 데이터 소스 - Snowflake 노드를 추가할 때 기존 AWS Glue Snowflake 연결을 선택하거나 새 연결을 생성할 수 있습니다. Snowflake에 연결하도록 구성된 JDBC 유형 연결이 아니라 SNOWFLAKE 유형 연결을 선택해야 합니다. 다음 절차에 따라 AWS Glue Snowflake 연결을 생성합니다.

Snowflake 연결을 생성하려면

1. Snowflake에서 사용자, *snowflakeUser* 및 암호, *snowflakePassword*를 생성합니다.
2. 이 사용자가 상호 작용하는 Snowflake 웨어하우스(*snowflakeWarehouse*)를 결정합니다. Snowflake에서 *snowflakeUser*에 대해 DEFAULT_WAREHOUSE로 설정하거나 다음 단계에서 사용하도록 기억합니다.
3. AWS Secrets Manager에서 Snowflake 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
   - 키/값 페어를 선택하면 키가 sfUser인 *snowflakeUser*에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 키가 sfPassword인 *snowflakePassword*에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 키가 sfWarehouse인 *snowflakeWarehouse*에 대한 페어를 생성합니다. Snowflake에 기본값이 설정된 경우에는 필요하지 않습니다.
4. AWS Glue 데이터 카탈로그에서 AWS Glue 연결 추가 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때 Snowflake를 선택합니다.
   - Snowflake URL을 선택할 때 Snowflake 인스턴스의 호스트 이름을 제공합니다. URL은 *account_identifier*.snowflakecomputing.com 양식의 호스트 이름을 사용합니다.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

Snowflake 소스 노드 생성

필요한 권한

Snowflake 데이터 소스를 사용하는 AWS Glue Studio 작업에는 추가 권한이 필요합니다. ETL 작업에 권한을 추가하는 방법에 대한 자세한 내용은 ETL 작업에 필요한 IAM 권한 검토를 참조하세요.

SNOWFLAKE AWS Glue 연결은 AWS Secrets Manager 보안 암호를 사용하여 보안 인증 정보를 제공합니다. AWS Glue Studio에서 작업 및 데이터 미리 보기 역할에는 이 보안 암호를 읽을 수 있는 권한이 있어야 합니다.

Snowflake 데이터 소스 추가

사전 조건:

- Snowflake 보안 인증의 AWS Secrets Manager 보안 암호
- Snowflake 유형 AWS Glue 데이터 카탈로그 연결

데이터 소스 - Snowflake 노드를 추가하려면:

1. Snowflake 데이터 소스의 연결을 선택합니다. 이 경우 연결이 이미 존재하며 기존 연결 중에서 선택할 수 있다고 가정합니다. 연결을 생성해야 하는 경우 Snowflake 연결 생성을 선택합니다. 자세한 내용은 커넥터 및 연결 사용 개요를 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다. URL, 보안 그룹, 서브넷, 가용 영역, 설명, 생성 날짜(UTC) 및 최종 업데이트(UTC) 타임스탬프를 비롯한 연결 정보가 표시됩니다.
2. 다음과 같은 Snowflake 소스 옵션을 선택합니다.

   - 단일 테이블 선택 - 단일 Snowflake 테이블에서 액세스하려는 데이터가 들어 있는 테이블입니다.
   - 사용자 지정 쿼리 입력 - 사용자 지정 쿼리를 기반으로 여러 Snowflake 테이블의 데이터 세트에 액세스할 수 있습니다.
3. 단일 테이블을 선택한 경우 Snowflake 스키마의 이름을 입력합니다.

   또는 사용자 지정 쿼리 입력을 선택합니다. 여러 Snowflake 테이블에서 사용자 지정 데이터 세트에 액세스하려면 이 옵션을 선택합니다. 이 옵션을 선택하는 경우 Snowflake 쿼리를 입력합니다.
4. 성능 및 보안 옵션(선택 사항)에서,

   - 쿼리 푸시다운 활성화 - Snowflake 인스턴스로 작업을 오프로드할지 여부를 선택합니다.
5. 사용자 지정 Snowflake 속성(선택 사항)에서 필요한 경우 파라미터와 값을 입력합니다.

Snowflake 대상 노드 생성

필요한 권한

Snowflake 데이터 소스를 사용하는 AWS Glue Studio 작업에는 추가 권한이 필요합니다. ETL 작업에 권한을 추가하는 방법에 대한 자세한 내용은 ETL 작업에 필요한 IAM 권한 검토를 참조하세요.

SNOWFLAKE AWS Glue 연결은 AWS Secrets Manager 보안 암호를 사용하여 보안 인증 정보를 제공합니다. AWS Glue Studio에서 작업 및 데이터 미리 보기 역할에는 이 보안 암호를 읽을 수 있는 권한이 있어야 합니다.

Snowflake 데이터 대상 추가

Snowflake 대상 노드를 생성하려면:

1. 기존 Snowflake 테이블을 대상으로 선택하거나 새 테이블 이름을 입력합니다.
2. 데이터 대상 - Snowflake 대상 노드를 사용하는 경우 다음 옵션 중에서 선택할 수 있습니다.

   - 추가 - 테이블이 이미 있는 경우 모든 새 데이터를 테이블에 삽입으로 덤프합니다. 테이블이 없으면 새로 생성한 후 새 데이터를 모두 삽입합니다.
   - 병합 - AWS Glue는 사용자가 지정한 조건에 따라 대상 테이블에 데이터를 추가하거나 업데이트합니다.

     다음과 같은 옵션을 선택합니다.

     - 키 및 간단한 작업 선택 - 소스 데이터와 대상 데이터 세트 사이에서 일치하는 키로 사용할 열을 선택합니다.

       일치하는 경우 다음 옵션을 지정합니다.

       - 대상 데이터 세트의 레코드를 소스의 데이터로 업데이트합니다.
       - 대상 데이터 세트에서 레코드를 삭제합니다.

       일치하지 않는 경우 다음 옵션을 지정합니다.

       - 소스 데이터를 대상 데이터 세트에 새 행으로 삽입합니다.
       - 아무 작업 안 함.
     - 사용자 지정 MERGE 명령문 입력 - 그런 다음 병합 명령문 검증을 선택하여 명령문이 유효한지 여부를 검증할 수 있습니다.

- 잘라내기 - 테이블이 이미 있는 경우 먼저 대상 테이블의 콘텐츠를 지워서 테이블 데이터를 잘라냅니다. 잘라내기에 성공하면 모든 데이터를 삽입합니다. 테이블이 없는 경우 테이블을 생성하고 모든 데이터를 삽입합니다. 잘라내기에 실패하면 작업에 실패합니다.
- 삭제 - 테이블이 이미 있는 경우 테이블 메타데이터와 데이터를 삭제합니다. 삭제에 성공하면 모든 데이터를 삽입합니다. 테이블이 없는 경우 테이블을 생성하고 모든 데이터를 삽입합니다. 삭제에 실패하면 작업에 실패합니다.

고급 옵션

AWS Glue 개발자 안내서의 Snowflake 연결을 참조하세요.

AWS Glue Studio에서 Stripe에 연결

Stripe는 비즈니스를 위한 온라인 결제 처리 및 신용카드 처리 플랫폼입니다. Stripe 플랫폼을 사용하면 비즈니스에서 온라인 결제를 수락하고, 전자 상거래에 대한 구독(반복 결제)을 생성하며, 결제 대금을 받을 수 있도록 계정을 다시 설정할 수 있습니다. 또한 Stripe에서는 비즈니스가 자체 마켓플레이스를 설정하고 결제 대금을 수금한 후 '연결'된 계정을 통해 판매자 또는 서비스 제공업체에 지불할 수 있도록 하는 다자간 결제를 지원합니다.

주제

- Stripe에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Stripe 구성
- Stripe 연결 구성
- Stripe 엔터티에서 읽기
- Stripe 연결 옵션
- 제한 사항
- 새 Stripe 계정 생성 및 클라이언트 앱 구성

Stripe에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Stripe를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Stripe에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Slack API 버전

v1.

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

아래 관리형 IAM 정책을 사용하여 다음에 대한 액세스를 허용할 수 있습니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, Amazon CloudWatch Logs, Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Stripe 구성

AWS Glue를 사용하여 Stripe에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

- 이메일 및 암호를 사용하는 Stripe 계정이 있어야 합니다. 자세한 내용은 새 Stripe 계정 생성 및 클라이언트 앱 구성 단원을 참조하십시오.
- Stripe 계정이 API 액세스에 대해 활성화되어 있습니다. Stripe API의 모든 사용은 추가 비용 없이 이용 가능합니다.

이러한 요구 사항을 충족하면 Stripe 계정에 AWS Glue를 연결할 준비가 된 것입니다.

Stripe 연결 구성

Stripe에서는 사용자 지정 인증을 지원합니다. 사용자 지정 권한 부여에 필요한 API 키를 생성하는 방법에 대한 자세한 내용은 STRIPE REST API 설명서를 참조하세요.

Stripe 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성하세요. AWS Glue에서 각 연결에 대한 보안 암호를 생성해야 합니다.
   a. 고객 관리형 연결된 앱의 경우 - 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Stripe를 선택하세요.
   b. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 IAM 역할을 선택하세요.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
```

```
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

c. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

d. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Stripe 엔터티에서 읽기

사전 조건

- 읽으려는 Stripe 객체.

지원되는 엔터티

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
밸런스	아니요	아니요	아니요	예	아니요
밸런스 트랜잭션	예	예	아니요	예	예
요금	예	예	아니요	예	예
이의	예	예	아니요	예	예
파일 링크	예	예	아니요	예	예
PaymentIntents	예	예	아니요	예	예
SetupIntents	예	예	아니요	예	예
결제 금액	예	예	아니요	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
환불	예	예	아니요	예	예
제품	예	예	아니요	예	예
가격	예	예	아니요	예	예
쿠폰	예	예	아니요	예	예
프로모션 코드	예	예	아니요	예	예
세금 코드	아니요	예	아니요	예	아니요
세율	예	예	아니요	예	예
배송 요금	예	예	아니요	예	예
세션	예	예	아니요	예	예
크레딧 노트	예	예	아니요	예	예
고객	예	예	아니요	예	예
인보이스	예	예	아니요	예	예
인보이스 항목	예	예	아니요	예	아니요
계획	예	예	아니요	예	예
견적	예	예	아니요	예	아니요
구독	예	예	아니요	예	
구독 항목	아니요	예	아니요	예	아니요
구독 일정	예	예	아니요	예	예
계정	아니요	예	아니요	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
애플리케이션 요금	예	예	아니요	예	예
국가 사양	아니요	예	아니요	예	아니요
전송	예	예	아니요	예	예
조기 사기 경고	예	예	아니요	예	예
보고서 유형	아니요	아니요	아니요	예	아니요

예제

```
stripe_read = glueContext.create_dynamic_frame.from_options(
    connection_type="stripe",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "coupons",
        "API_VERSION": "v1"
    }
)
```

Stripe 엔터티 및 필드 세부 정보

개체	필드	데이터 형식	지원되는 연산자
밸런스			
	available	나열	
	connect_reserved	나열	
	보류 중	나열	
	livemode	불	

개체	필드	데이터 형식	지원되는 연산자
	객체	String	
	instant_available	나열	
	issuing	구조체	
밸런스 트랜잭션			
	id	String	
	객체	String	
	amount	Integer	
	available_on	DateTime	=, >=, <=,<,>
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	
	설명	String	
	exchange_rate	BigDecimal	
	fee	Integer	
	fee_details	나열	
	net	Integer	
	reporting_category	String	
	source	String	=
	status	String	
	type	String	=
	cross_border_classification	String	

개체	필드	데이터 형식	지원되는 연산자
요금			
	id	String	
	객체	String	
	amount	Integer	=, <, >
	amount_captured	Integer	
	amount_refunded	Integer	
	애플리케이션	String	
	application_fee	String	
	application_fee_amount	Integer	
	balance_transaction	String	
	billing_details	구조체	
	calculated_statement_descriptor	String	
	captured	불	
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	
	customer	String	=
	설명	String	
	destination	String	
	dispute	String	

개체	필드	데이터 형식	지원되는 연산자
	disputed	불	=
	failure_balance_transaction	String	
	failure_code	String	
	failure_message	String	
	fraud_details	구조체	
	인보이스	String	
	livemode	불	
	metadata	구조체	
	on_behalf_of	String	
	순서	String	
	outcome	구조체	
	paid	불	
	payment_intent	String	=
	payment_method	String	
	payment_method_details	구조체	
	receipt_email	String	
	receipt_number	String	
	receipt_url	String	
	refunded	불	=

개체	필드	데이터 형식	지원되는 연산자
	환불	구조체	
	리뷰	String	
	shipping	구조체	
	source	구조체	
	source_transfer	String	
	statement_descriptor	String	
	statement_descriptor_suffix	String	
	status	String	
	전송	String	
	transfer_data	구조체	
	transfer_group	String	=
이의			
	id	String	
	객체	String	
	amount	Integer	=, <, >
	balance_transaction	String	
	balance_transactions	나열	
	요금	String	=
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	

개체	필드	데이터 형식	지원되는 연산자
	evidence	구조체	
	evidence_details	구조체	
	is_charge_refundable	불	
	livemode	불	
	metadata	구조체	
	payment_intent	String	=
	reason	String	=
	status	String	
	payment_method_details	구조체	
파일 링크			
	id	String	
	객체	String	
	생성 완료	DateTime	=, >=, <=,<,>
	expired	불	=
	expires_at	DateTime	
	파일	String	=
	livemode	불	
	metadata	구조체	
	url	String	
PaymentIntents			

개체	필드	데이터 형식	지원되는 연산자
	id	String	
	객체	String	
	amount	Integer	
	amount_capturable	Integer	
	amount_details	구조체	
	amount_received	Integer	
	애플리케이션	String	
	application_fee_amount	Integer	
	automatic_payment_methods	구조체	
	canceled_at	DateTime	
	cancellation_reason	String	
	capture_method	String	
	client_secret	String	
	confirmation_method	String	
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	
	customer	String	=
	설명	String	
	인보이스	String	

개체	필드	데이터 형식	지원되는 연산자
	last_payment_error	구조체	
	latest_charge	String	
	livemode	불	
	metadata	구조체	
	next_action	구조체	
	on_behalf_of	String	
	payment_method	String	
	payment_method_options	구조체	
	payment_method_types	나열	
	payment_method_configuration_details	구조체	
	처리 중	구조체	
	receipt_email	String	
	리뷰	String	
	setup_future_usage	String	
	shipping	구조체	
	source	String	
	statement_descriptor	String	
	statement_descriptor_suffix	String	

개체	필드	데이터 형식	지원되는 연산자
	status	String	
	transfer_data	구조체	
	transfer_group	String	
SetupIntents			
	id	String	
	객체	String	
	애플리케이션	String	
	cancellation_reason	String	
	client_secret	String	
	생성 완료	DateTime	=, >=, <=,<,>
	customer	String	=
	설명	String	
	flow_directions	나열	
	last_setup_error	구조체	
	latest_attempt	String	
	livemode	불	
	mandate	String	
	metadata	구조체	
	next_action	구조체	
	on_behalf_of	String	

개체	필드	데이터 형식	지원되는 연산자
	payment_method	String	
	payment_method_options	구조체	
	payment_method_types	나열	
	single_use_mandate	String	
	status	String	
	사용	String	
	automatic_payment_methods	구조체	
결제 금액			
	id	String	
	객체	String	
	amount	Integer	=, <, >
	arrival_date	DateTime	=, >=, <=,<,>
	automatic	불	
	balance_transaction	String	
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	
	설명	String	=
	destination	String	

개체	필드	데이터 형식	지원되는 연산자
	failure_balance_transaction	String	
	failure_code	String	
	failure_message	String	
	livemode	불	
	metadata	구조체	
	method	String	
	original_payout	String	
	reversed_by	String	
	reconciliation_status	String	
	source_type	String	
	statement_descriptor	String	
	status	String	
	type	String	
	application_fee	String	
	application_fee_amount	Integer	
환불			
	id	String	
	객체	String	
	amount	Integer	

개체	필드	데이터 형식	지원되는 연산자
	balance_transaction	String	
	요금	String	=
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	
	metadata	구조체	
	destination_details	구조체	
	payment_intent	String	=
	reason	String	
	receipt_number	String	
	source_transfer_reversal	String	
	status	String	
	transfer_reversal	String	
제품			
	id	String	
	객체	String	
	활성화	불	=
	attributes	나열	
	생성 완료	DateTime	=, >=, <=,<,>
	default_price	String	
	설명	String	

개체	필드	데이터 형식	지원되는 연산자
	이미지	나열	
	livemode	불	
	metadata	구조체	
	name	String	
	package_dimensions	구조체	
	shippable	불	
	statement_descriptor	String	
	tax_code	String	
	type	String	=
	unit_label	String	
	updated	DateTime	
	url	String	
	기능	나열	
가격			
	id	String	
	객체	String	
	활성화	불	=
	billing_scheme	String	
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	=

개체	필드	데이터 형식	지원되는 연산자
	custom_unit_amount	구조체	
	livemode	불	
	lookup_key	String	
	metadata	구조체	
	nickname	String	
	product	String	=
	recurring	구조체	
	tax_behavior	String	
	tiers_mode	String	
	transform_quantity	구조체	
	type	String	=
	unit_amount	Integer	
	unit_amount_decimal	String	
쿠폰			
	Id	String	
	객체	String	
	amount_off	Integer	
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	=
	duration	String	=

개체	필드	데이터 형식	지원되는 연산자
	duration_in_months	Integer	=,<,>
	livemode	불	
	max_redemptions	Integer	=, <, >
	metadata	구조체	
	name	String	
	percent_off	Double	=
	redeem_by	DateTime	=, >=, <=, <, >
	times_redeemed	Integer	
	유효함	불	
프로모션 코드			
	Id	String	
	객체	String	
	활성화	불	=
	code	String	=
	coupon	구조체	
	생성 완료	DateTime	=,>=,<=,<,>
	customer	String	
	expires_at	DateTime	
	livemode	불	
	max_redemptions	Integer	

개체	필드	데이터 형식	지원되는 연산자
	metadata	구조체	
	제한 사항	구조체	
	times_redeemed	Integer	
세금 코드			
	Id	String	
	객체	String	
	설명	String	
	name	String	
세율			
	Id	String	
	객체	String	
	활성화	불	=
	country	String	
	생성 완료	DateTime	=, >=, <=, <, >
	설명	String	
	display_name	String	
	inclusive	불	=
	jurisdiction	String	
	jurisdiction_level	String	
	livemode	불	

개체	필드	데이터 형식	지원되는 연산자
	metadata	구조체	
	percentage	Double	
	effective_percentage	Double	
	state	String	
	tax_type	String	
배송 요금			
	Id	String	
	객체	String	
	활성화	불	=
	생성 완료	DateTime	=, >=, <=, <, >
	delivery_estimate	구조체	
	display_name	String	
	fixed_amount	구조체	
	livemode	불	
	metadata	구조체	
	tax_behavior	String	
	tax_code	String	
	type	String	
세션			
	id	String	

개체	필드	데이터 형식	지원되는 연산자
	객체	String	
	after_expiration	구조체	
	allow_promotion_codes	불	
	amount_subtotal	Integer	
	amount_total	Integer	
	automatic_tax	구조체	
	billing_address_collection	String	
	cancel_url	String	
	client_reference_id	String	
	consent	구조체	
	consent_collection	구조체	
	생성 완료	DateTime	=, >=, <=, <, >
	currency	String	
	custom_text	구조체	
	customer	String	
	customer_creation	String	
	customer_details	구조체	
	customer_email	String	
	expires_at	DateTime	

개체	필드	데이터 형식	지원되는 연산자
	인보이스	String	
	invoice_creation	구조체	
	livemode	불	
	locale	String	
	metadata	구조체	
	mode	String	
	payment_intent	String	=
	payment_link	String	
	payment_method_collection	String	
	payment_method_options	구조체	
	payment_method_types	나열	
	payment_status	String	
	phone_number_collection	구조체	
	recovered_from	String	
	setup_intent	String	
	shipping_address_collection	구조체	
	shipping_cost	구조체	

개체	필드	데이터 형식	지원되는 연산자
	shipping_details	구조체	
	shipping_options	나열	
	status	String	
	submit_type	String	
	구독	String	
	success_url	String	
	tax_id_collection	구조체	
	total_details	구조체	
	url	String	
	ui_mode	String	
크레딧 노트			
	id	String	
	객체	String	
	amount	Integer	
	생성 완료	DateTime	=, >=, <=, <, >
	currency	String	
	customer	String	=
	customer_balance_transaction	String	
	discount_amount	Integer	
	discount_amounts	나열	

개체	필드	데이터 형식	지원되는 연산자
	인보이스	String	=
	lines	구조체	
	livemode	불	
	memo	String	
	metadata	구조체	
	number	String	
	out_of_band_amount	Integer	
	pdf	String	
	reason	String	
	refund	String	
	status	String	
	subtotal	Integer	
	subtotal_excluding_tax	Integer	
	tax_amounts	나열	
	총합	Integer	
	total_excluding_tax	Integer	
	type	String	
	voided_at	DateTime	
	amount_shipping	Integer	
	effective_at	DateTime	

개체	필드	데이터 형식	지원되는 연산자
	shipping_cost	구조체	
고객			
	id	String	
	객체	String	
	address	구조체	
	balance	Integer	
	생성 완료	DateTime	
	currency	String	=, >=, <=, <, >
	default_source	String	
	delinquent	불	=
	설명	String	
	discount	구조체	
	이메일	String	=
	invoice_prefix	String	
	invoice_settings	구조체	
	livemode	불	
	metadata	구조체	
	name	String	
	next_invoice_sequence	Integer	
	phone	String	

개체	필드	데이터 형식	지원되는 연산자
	preferred_locales	나열	
	shipping	구조체	
	tax_exempt	String	
	test_clock	String	
인보이스			
	id	String	
	객체	String	
	account_country	String	
	account_name	String	
	account_tax_ids	나열	
	amount_due	Integer	
	amount_paid	Integer	
	amount_remaining	Integer	
	애플리케이션	String	
	application_fee_amount	Integer	
	attempt_count	Integer	
	attempted	불	=
	auto_advance	불	=
	automatic_tax	구조체	
	billing_reason	String	

개체	필드	데이터 형식	지원되는 연산자
	요금	String	
	collection_method	String	=
	생성 완료	DateTime	=, >=, <=, <, >
	currency	String	
	custom_fields	나열	
	customer	String	=
	customer_address	구조체	
	customer_email	String	
	customer_name	String	
	customer_phone	String	
	customer_shipping	구조체	
	customer_tax_exempt	String	
	customer_tax_ids	나열	
	default_payment_method	String	
	default_source	String	
	default_tax_rates	나열	
	설명	String	
	discount	구조체	
	discounts	나열	
	due_date	DateTime	=, >=, <=, <, >

개체	필드	데이터 형식	지원되는 연산자
	ending_balance	Integer	
	footer	String	
	from_invoice	구조체	
	hosted_invoice_url	String	
	invoice_pdf	String	
	last_finalization_error	구조체	
	latest_revision	String	
	lines	구조체	
	livemode	불	
	metadata	구조체	
	next_payment_attempt	DateTime	
	number	String	
	on_behalf_of	String	
	paid	불	=
	paid_out_of_band	불	
	payment_intent	String	
	payment_settings	구조체	
	period_end	DateTime	=, >=, <=, <, >
	period_start	DateTime	=, >=, <=, <, >

개체	필드	데이터 형식	지원되는 연산자
	post_payment_credit_notes_amount	Integer	
	pre_payment_credit_notes_amount	Integer	
	quote	String	
	receipt_number	String	
	rendering	구조체	
	rendering_options	구조체	
	starting_balance	Integer	
	statement_descriptor	String	
	status	String	=
	status_transitions	구조체	
	구독	String	
	subscription_details	구조체	
	subtotal	Integer	=, <, >
	subtotal_excluding_tax	Integer	
	tax	Integer	
	test_clock	String	
	총합	Integer	=, <, >
	total_discount_amounts	나열	

개체	필드	데이터 형식	지원되는 연산자
	total_excluding_tax	Integer	
	total_tax_amounts	나열	
	transfer_data	구조체	
	webhooks_delivered_at	DateTime	
	automatically_finalizes_at	DateTime	
	effective_at	DateTime	
	발행자	구조체	
인보이스 항목			
	id	String	
	객체	String	
	amount	Integer	=, <, >
	currency	String	
	customer	String	=
	date	DateTime	
	설명	String	
	discountable	불	
	discounts	나열	
	인보이스	String	=
	livemode	불	

개체	필드	데이터 형식	지원되는 연산자
	metadata	구조체	
	기간	구조체	
	계획	구조체	
	가격	구조체	
	proration	불	=
	양	Integer	
	구독	String	
	subscription_item	String	
	tax_rates	나열	
	test_clock	String	
	unit_amount	Integer	
	unit_amount_decimal	String	
계획			
	id	String	
	객체	String	
	활성화	불	=
	aggregate_usage	String	
	amount	Integer	
	amount_decimal	String	
	billing_scheme	String	

개체	필드	데이터 형식	지원되는 연산자
	생성 완료	DateTime	=, >=, <=, <, >
	currency	String	=
	interval	String	=
	interval_count	Integer	
	livemode	불	
	metadata	구조체	
	nickname	String	
	product	String	=
	tiers_mode	String	
	transform_usage	구조체	
	trial_period_days	Integer	=, <, >
	usage_type	String	
	측정	String	
견적			
	id	String	
	객체	String	
	amount_subtotal	Integer	
	amount_total	Integer	
	애플리케이션	String	
	application_fee_amount	Integer	

개체	필드	데이터 형식	지원되는 연산자
	application_fee_percent	Double	
	automatic_tax	구조체	
	collection_method	String	
	computed	구조체	
	생성 완료	DateTime	
	currency	String	
	customer	String	=
	default_tax_rates	나열	
	설명	String	
	discounts	나열	
	expires_at	DateTime	
	footer	String	
	from_quote	구조체	
	header	String	
	인보이스	String	
	invoice_settings	구조체	
	livemode	불	
	metadata	구조체	
	number	String	
	on_behalf_of	String	

개체	필드	데이터 형식	지원되는 연산자
	status	String	=
	status_transitions	구조체	
	구독	String	
	subscription_data	구조체	
	subscription_schedule	String	
	test_clock	String	
	total_details	구조체	
	transfer_data	구조체	
구독			
	id	String	
	객체	String	
	애플리케이션	String	
	application_fee_percent	Double	
	automatic_tax	구조체	
	billing_cycle_anchor	DateTime	
	billing_thresholds	구조체	
	cancel_at	DateTime	
	cancel_at_period_end	불	
	canceled_at	DateTime	
	collection_method	String	=

개체	필드	데이터 형식	지원되는 연산자
	생성 완료	DateTime	=, >=, <=,<,>
	currency	String	
	current_period_end	DateTime	=, >=, <=
	current_period_start	DateTime	=, >=, <=
	customer	String	=
	days_until_due	Integer	
	default_payment_method	String	
	default_source	String	
	default_tax_rates	나열	
	설명	String	
	discount	구조체	
	ended_at	DateTime	
	항목	구조체	
	latest_invoice	String	
	livemode	불	
	metadata	구조체	
	next_pending_invoice_item_invoice	DateTime	
	pause_collection	구조체	
	payment_settings	구조체	

개체	필드	데이터 형식	지원되는 연산자
	pending_invoice_item_interval	구조체	
	pending_setup_intent	String	
	pending_update	구조체	
	계획	구조체	
	양	Integer	
	schedule	String	
	start_date	DateTime	
	status	String	=
	test_clock	String	
	transfer_data	구조체	
	trial_end	DateTime	
	trial_start	DateTime	
구독 항목			
	Id	String	
	객체	String	
	billing_thresholds	구조체	
	생성 완료	DateTime	=, >=, <=, <, >
	metadata	구조체	
	계획	구조체	
	가격	구조체	

개체	필드	데이터 형식	지원되는 연산자
	구독	String	
	tax_rates	나열	
	discounts	나열	
구독 일정			
	객체	String	
	애플리케이션	String	
	canceled_at	DateTime	
	completed_at	DateTime	
	생성 완료	DateTime	
	current_phase	구조체	
	customer	String	=
	default_settings	구조체	
	end_behavior	String	
	livemode	불	
	metadata	구조체	
	단계	나열	
	released_at	DateTime	
	released_subscription	String	
	renewal_interval	String	
	status	String	

개체	필드	데이터 형식	지원되는 연산자
	구독	String	
	test_clock	String	
계정			
	details_submitted	불	
	tos_acceptance	구조체	
	type	String	
	metadata	구조체	
	id	String	
	객체	String	
	default_currency	String	
	역량	구조체	
	charges_enabled	불	
	설정	구조체	
	요구 사항	구조체	
	payouts_enabled	불	
	future_requirements	구조체	
	external_accounts	구조체	
	컨트롤러	구조체	
	country	String	
	이메일	String	

개체	필드	데이터 형식	지원되는 연산자
	생성 완료	DateTime	=, >=, <=, <, >
	business_profile	구조체	
	business_type	String	
	company	구조체	
애플리케이션 요금			
	id	String	
	객체	String	
	account	String	
	amount	Integer	=, <, >
	amount_refunded	Integer	=, <, >
	애플리케이션	String	
	balance_transaction	String	
	요금	String	=
	생성 완료	DateTime	
	currency	String	
	livemode	불	
	originating_transaction	String	
	refunded	불	=
	환불	구조체	
	fee_source	구조체	

개체	필드	데이터 형식	지원되는 연산자
국가 사양			
	id	String	
	객체	String	
	default_currency	String	
	supported_bank_account_currencies	구조체	
	supported_payment_currencies	나열	
	supported_payment_methods	나열	
	supported_transfer_countries	나열	
	verification_fields	구조체	
전송			
	id	String	
	객체	String	
	amount	Integer	=, <, >
	amount_reversed	Integer	
	balance_transaction	String	
	생성 완료	DateTime	=, >=, <=, <, >
	currency	String	=
	설명	String	

개체	필드	데이터 형식	지원되는 연산자
	destination	String	=
	destination_payment	String	
	livemode	불	
	metadata	구조체	
	reversals	구조체	
	reversed	불	
	source_transaction	String	
	source_type	String	
	transfer_group	String	=
조기 사기 경고			
	id	String	
	객체	String	
	actionable	불	
	요금	String	=
	생성 완료	DateTime	=, >=, <=, <, >
	fraud_type	String	
	livemode	불	
	payment_intent	String	=
보고서 유형			
	id	String	

개체	필드	데이터 형식	지원되는 연산자
	객체	String	
	data_available_end	DateTime	
	data_available_start	DateTime	
	default_columns	나열	
	livemode	불	
	name	String	
	updated	DateTime	
	version	Integer	

분할 쿼리

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원본 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  날짜의 경우 Spark SQL 쿼리에 사용된 Spark 날짜 형식을 허용합니다. 유효한 값의 예제: "2024-07-01T00:00:00.000Z".
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

엔터티 수준 분할 필드 지원 세부 정보는 아래 표에 캡처되어 있습니다.

Entity Name	분할 필드	데이터 형식
밸런스 트랜잭션	생성 완료	DateTime

Entity Name	분할 필드	데이터 형식
요금	생성 완료	DateTime
이의	생성 완료	DateTime
파일 링크	생성 완료	DateTime
PaymentIntents	생성 완료	DateTime
SetupIntents	생성 완료	DateTime
결제 금액	생성 완료	DateTime
환불	생성 완료	DateTime
제품	생성 완료	DateTime
가격	생성 완료	DateTime
쿠폰	생성 완료	DateTime
프로모션 코드	생성 완료	DateTime
세율	생성 완료	DateTime
배송 요금	생성 완료	DateTime
세션	생성 완료	DateTime
크레딧 노트	생성 완료	DateTime
고객	생성 완료	DateTime
인보이스	생성 완료	DateTime
계획	생성 완료	DateTime
구독	생성 완료	DateTime
구독 일정	생성 완료	DateTime

Entity Name	분할 필드	데이터 형식
계정	생성 완료	DateTime
애플리케이션 요금	생성 완료	DateTime
전송	생성 완료	DateTime
조기 사기 경고	생성 완료	DateTime

예제

```
stripe_read = glueContext.create_dynamic_frame.from_options(
    connection_type="stripe",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "coupons",
        "API_VERSION": "v1",
        "PARTITION_FIELD": "created"
        "LOWER_BOUND": "2024-05-01T20:55:02.000Z"
        "UPPER_BOUND": "2024-07-11T20:55:02.000Z"
        "NUM_PARTITIONS": "10"
    }
)
```

Stripe 연결 옵션

다음은 Stripe에 대한 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기/쓰기에 사용됩니다. Stripe에서의 객체 이름.
- API_VERSION(문자열) - (필수) 읽기/쓰기에 사용됩니다. 사용할 Stripe Rest API 버전. 예: v1.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리를 파티셔닝하는 데 사용할 필드.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).

- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

제한 사항

다음은 Stripe 커넥터에 대한 제한 사항입니다.

- 커넥터에서 지원하는 필드 기반 분할만 해당됩니다.
- 커넥터에서 레코드 기반 분할을 지원하지 않으며 총 레코드 수를 검색하도록 프로비저닝되지 않습니다.
- 프라이머리 키 데이터 유형은 문자열이므로 ID 기반 분할은 커넥터에서 지원하지 않습니다.

새 Stripe 계정 생성 및 클라이언트 앱 구성

Stripe 계정 생성

1. https://dashboard.stripe.com/register 링크를 선택하세요.
2. 이메일, 전체 이름, 암호를 입력하고 Create Account를 선택하세요.
3. 계정으로 로그인한 후 Open Gmail을 선택하여 계정을 확인하세요.
4. 이메일에 수신된 확인 링크를 클릭하여 계정을 확인하세요.
5. 이메일 주소 확인을 클릭하면 다른 페이지로 리디렉션됩니다.
6. Activate payments를 클릭하여 계정을 활성화하면 Activate payments(https://dashboard.stripe.com/welcome) 페이지로 리디렉션되고 유효한 세부 정보를 모두 입력한 후 Continue 버튼을 선택하세요.

Slack 개발자 앱 생성

1. Stripe에 로그인하세요.
2. 아래 그림의 맨 위에 표시된 Developers를 선택하세요.
3. Developers 아래에서 API keys를 선택하세요.
4. API 키를 가져오려면 Reveal test key를 선택하세요.

AWS Glue Studio의 Teradata 밴티지에 연결

AWS Glue에서는 Teradata Vantage를 기본으로 지원합니다. AWS Glue Studio에는 Teradata에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스가 있습니다.

AWS Glue Studio에서 Teradata Vantage용 통합 연결을 생성합니다. 자세한 내용은 고려 사항 섹션을 참조하세요.

주제

- Teradata Vantage 연결 생성
- Teradata 소스 노드 생성
- Teradata 대상 노드 생성
- 고급 옵션

Teradata Vantage 연결 생성

AWS Glue에서 Teradata Vantage에 연결하려면 Teradata 보안 인증 정보를 생성하여 AWS Secrets Manager 암호에 저장한 다음 해당 암호를 AWS Glue Teradata 연결에 연결해야 합니다.

사전 조건:

- Amazon VPC를 통해 Teradata 환경에 액세스하는 경우, AWS Glue 작업이 Teradata 환경과 통신할 수 있도록 Amazon VPC를 구성하십시오. 퍼블릭 인터넷을 통해 Teradata 환경에 액세스하는 것은 권장하지 않습니다.

  Amazon VPC에서 작업을 실행하는 동안 AWS Glue가 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 Amazon VPC가 Teradata 인스턴스와 이 위치 간의 네트워크 트래픽을 허용하도록 구성되어 있는지 확인해야 합니다. 작업을 수행하려면 Teradata 클라이언트 포트와 TCP 연결을 설정해야 합니다. Teradata 포트에 대한 자세한 내용은 Teradata 설명서를 참조하십시오.

  네트워크 레이아웃에 따라 보안 VPC 연결에는 Amazon VPC 및 기타 네트워킹 서비스를 변경해야 할 수 있습니다. AWS 연결에 대한 자세한 내용은 Teradata 설명서의 AWS연결 옵션을 참조하십시오.

AWS Glue Teradata 연결을 구성하는 방법:

1. Teradata 구성에서 AWS Glue가 *teradataUser* 및 *teradataPassword*와 연결할 사용자 및 암호를 식별하거나 생성합니다. 자세한 내용은 Teradata 설명서의 Vantage 보안 개요를 참조하십시오.
2. AWS Secrets Manager에서 Teradata 보안 인증 정보를 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
   - 키/값 페어를 선택하면 값 *teradataUsername*이 포함된 키 `user`에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *teradataPassword*가 포함된 키 `password`에 대한 페어를 생성합니다.
3. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때 Teradata를 선택합니다.
   - JDBC URL을 제공할 때는 인스턴스의 URL을 제공하십시오. 또한 JDBC URL에 쉼표로 구분된 특정 연결 매개변수를 하드코딩할 수 있습니다. URL의 형식: `jdbc:teradata://`*teradataHostname*`/`*ParameterName*`=`*ParameterValue*`,`*ParameterName*

     지원되는 URL 파라미터는 다음과 같습니다.
     - `DATABASE` - 기본으로 액세스하는 호스트의 데이터베이스 이름입니다.
     - `DBS_PORT` - 비표준 포트에서 실행할 때 사용되는 데이터베이스 포트입니다.
   - 보안 인증 정보 유형을 선택할 때에는 AWS Secrets Manager을 선택한 다음 AWS보안 암호를 *secretName*으로 설정합니다.
4. 다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.
   - Amazon VPC에서 AWS에 호스팅된 Teradata 인스턴스의 경우
     - Teradata 보안 보안 인증 정보를 정의하는 AWS Glue 연결에 Amazon VPC 연결 정보를 제공해야 합니다. 연결을 만들거나 업데이트할 때 네트워크 옵션에서 VPC, 서브넷 및 보안 그룹을 설정합니다.

Teradata 소스 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Teradata Vantage 연결 생성”에서 설명한 것처럼 AWS Secrets Manager 암호로 구성된 AWS Glue Teradata Vantage 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 읽으려는 Teradata 테이블, *tableName* 또는 쿼리 *targetQuery*.

Teradata 데이터 소스 추가

데이터 소스 - Teradata 노드를 추가하는 방법:

1. Teradata 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 새 연결 생성을 선택합니다. 자세한 내용은 이전the section called “Teradata Vantage 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. Teradata 소스 옵션 선택:
   - 단일 테이블 선택 - 단일 테이블에서 모든 데이터에 액세스할 수 있습니다.
   - 사용자 지정 쿼리 입력 - 사용자 지정 쿼리를 기반으로 여러 테이블의 데이터 세트에 액세스할 수 있습니다.
3. 단일 테이블을 선택한 경우 *tableName*을 입력합니다.

   사용자 지정 쿼리 입력을 선택한 경우 SQL SELECT 쿼리를 입력합니다.
4. 사용자 지정 Teradata 속성에서 필요한 경우 파라미터와 값을 입력합니다.

Teradata 대상 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Teradata Vantage 연결 생성”에서 설명한 것처럼 AWS Secrets Manager 암호로 구성된 AWS Glue Teradata Vantage 연결입니다.
- 연결에 사용되는 보안 암호를 읽을 작업에 대한 적절한 권한.
- 쓰고 싶은 Teradata 테이블, *tableName*

Teradata 데이터 대상 추가

데이터 대상 - Teradata 노드를 추가하는 방법:

1. Teradata 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 Teradata 연결 생성을 선택합니다. 자세한 내용은 커넥터 및 연결 사용 개요를 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. *tableName*을 제공하여 테이블 이름을 구성합니다.
3. 사용자 지정 Teradata 속성에서 필요한 경우 파라미터와 값을 입력합니다.

고급 옵션

Teradata 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “Teradata Vantage 연결”를 참조하세요.

Twilio에 연결

Twilio는 웹 서비스 API를 사용하여 전화 걸기와 받기, 문자 메시지 보내기와 받기, 기타 커뮤니케이션 기능을 수행할 수 있는 프로그래밍 가능한 커뮤니케이션 도구를 제공합니다. Twilio의 API는 커뮤니케이션을 위한 플랫폼을 지원합니다. 이러한 API 뒤에는 전 세계 커뮤니케이션 네트워크를 연결하고 최적화하는 소프트웨어 계층이 있어 사용자가 전 세계 누구에게나 전화를 걸고 메시지를 보낼 수 있습니다. Twilio 사용자는 Twilio 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, Twilio를 ETL 작업에서의 데이터 소스로 사용할 수 있습니다. 이러한 작업을 실행하여 Twilio 및 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

주제

- AWS Glue의 Twilio 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Twilio 구성
- Twilio 연결 구성
- Twilio 엔터티에서 읽기
- Twilio 연결 옵션

- Twilio 커넥터의 제한 사항 및 참고 사항

AWS Glue의 Twilio 지원

AWS Glue에서는 다음과 같이 Twilio를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Twilio에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Twilio API 버전

다음 Twilio API 버전이 지원됩니다.

- v1
- 2010-04-01

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
 }
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Twilio 구성

AWS Glue를 사용하여 Twilio에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- 사용자 이름과 암호를 사용하는 Twilio 계정이 있습니다.
- Twilio 계정이 API 액세스에 대해 활성화되어 있습니다.

이러한 요구 사항을 충족하면 Twilio 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Twilio에서 다른 작업을 수행하지 않아도 됩니다.

Twilio 연결 구성

Twilio는 기본 인증을 위한 사용자 이름과 암호를 지원합니다. 기본 인증은 클라이언트가 보호된 리소스에 액세스하기 위해 자격 증명을 직접 제공하는 간단한 인증 방법입니다. AWS Glue는 사용자 이름(계정 SID) 및 암호(인증 토큰)를 사용하여 Twilio API를 인증합니다.

기본 권한 부여 흐름에 대한 퍼블릭 Twilio 설명서는 Basic Authentication | Twilio를 참조하세요.

Twilio 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - 기본 인증의 경우: 보안 암호에는 계정 SID(사용자 이름) 및 인증 토큰(암호)이 있는 연결된 앱 소비자 보안 암호가 포함되어야 합니다.

> **Note**
>
> AWS Glue에서 연결에 대한 보안 암호를 생성해야 합니다.

2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Twilio를 선택합니다.
   b. 연결하려는 Twilio 인스턴스의 `Edge_Location` 항목을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.
3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Twilio 엔터티에서 읽기

사전 조건

읽으려는 Twilio 객체입니다. 객체 이름(예: `SMS-Message` 또는 `SMS-CountryPricing`)이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	인터페이스	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
SMS-Message	REST	예	예	아니요	예	예
SMS-CountryPricing	REST	아니요	아니요	아니요	예	아니요
Voice-Call	REST	예	예	아니요	예	아니요
Voice-Application	REST	예	예	아니요	예	아니요
Voice-OutgoingCallerID	REST	예	예	아니요	예	아니요
Voice-Queue	REST	예	예	아니요	예	아니요
Conversations-Conversation	REST	예	예	아니요	예	아니요
Conversations-User	REST	아니요	예	아니요	예	아니요
Conversations-Role	REST	아니요	예	아니요	예	아니요
Conversations-Configuration	REST	아니요	아니요	아니요	예	아니요
Conversations-Addr	REST	예	예	아니요	예	아니요

엔터티	인터페이스	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
essConfiguration						
Conversations-WebhookConfiguration	REST	아니요	아니요	아니요	예	아니요
Conversations-ParticipantConversation	REST	아니요	아니요	아니요	예	아니요
Conversations-Credential	REST	아니요	예	아니요	예	아니요
Conversations-ConversationService	REST	아니요	예	아니요	예	아니요

예시:

```
twilio_read = glueContext.create_dynamic_frame.from_options(
    connection_type="twilio",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "sms-message",
        "API_VERSION": "2010-04-01",
        "Edge_Location": "sydney.us1"
    }
```

Twilio 엔터티 및 필드 세부 정보:

엔터티	필드	데이터 유형	지원되는 연산자
SMS-Message	account_sid	String	N/A
	api_version	String	N/A
	body	String	N/A
	date_created	Datetime	N/A
	date_sent	Datetime	>=, <=, =
	date_updated	Datetime	N/A
	direction	String	N/A
	error_code	Integer	N/A
	error_message	String	N/A
	from	Integer	=
	messaging_service_sid	String	N/A
	num_media	String	N/A
	num_segments	String	N/A
	price	String	N/A
	price_unit	Struct	N/A
	sid	Integer	N/A
	status	String	N/A
	subresource_uris	Map	N/A
	to	Integer	=
	uri	Datetime	N/A

엔터티	필드	데이터 유형	지원되는 연산자
SMS-CountryPricing	country	String	N/A
	iso_country	String	N/A
	url	String	N/A
	outbound_sms_prices	List	N/A
	inbound_sms_prices	List	N/A
	price_unit	String	N/A
Voice-Call	account_sid	String	N/A
	annotation	String	N/A
	answered_by	String	N/A
	api_version	String	N/A
	caller_name	String	N/A
	date_created	Datetime	N/A
	date_updated	Datetime	N/A
	direction	String	N/A
	duration	String	N/A
	end_time	Datetime	>=, <=, =
	forwarded_from	String	N/A
	from	String	=
	from_formatted	String	N/A
	group_sid	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	parent_call_sid	String	N/A
	phone_number_sid	String	N/A
	price	String	N/A
	price_unit	String	N/A
	sid	String	N/A
	start_time	Datetime	>=, <=, =
	status	String	=
	subresource_uris	String	N/A
	to	String	=
	to_formatted	String	N/A
	trunk_sid	String	N/A
	uri	String	N/A
	queue_time	String	N/A
Voice-Application	account_sid	String	N/A
	api_version	String	N/A
	date_created	Datetime	N/A
	date_updated	Datetime	N/A
	friendly_name	String	=
	message_status_callback	String	N/A
	sid	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	sms_fallback_method	String	N/A
	sms_fallback_url	String	N/A
	sms_method	String	N/A
	sms_status_callback	String	N/A
	sms_url	String	N/A
	status_callback	String	N/A
	status_callback_method	String	N/A
	uri	String	N/A
	voice_caller_id_lookup	Boolean	N/A
	voice_fallback_method	String	N/A
	voice_fallback_url	String	N/A
	voice_method	String	N/A
	voice_url	String	N/A
public_application_connect_enabled	Boolean	N/A	
Voice-OutgoingCallerID	sid	String	N/A
	date_created	Datetime	N/A
	date_updated	Datetime	N/A
	account_sid	String	N/A
	friendly_name	String	=

엔터티	필드	데이터 유형	지원되는 연산자
	phone_number	String	=
	uri	String	N/A
Voice-Queue	date_created	Datetime	N/A
	date_updated	Datetime	N/A
	current_size	Integer	N/A
	friendly_name	String	N/A
	uri	String	N/A
	account_sid	String	N/A
	average_wait_time	Integer	N/A
	sid	String	N/A
	max_size	Integer	N/A
Conversations-Conversation	account_sid	String	N/A
	chat_service_sid	String	N/A
	messaging_service_sid	String	N/A
	sid	String	N/A
	friendly_name	String	N/A
	unique_name	String	N/A
	attributes	String	N/A
	state	String	=
	date_created	Datetime	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	date_updated	Datetime	N/A
	timers	Struct	N/A
	url	String	N/A
	links	Struct	N/A
	bindings	Struct	N/A
	start_date	Datetime	=
	end_date	Datetime	=
	Timers.DateInactive	String	N/A
	Timers.DateClosed	String	N/A
Conversations-User	sid	String	N/A
	account_sid	String	N/A
	chat_service_sid	String	N/A
	role_sid	String	N/A
	identity	String	N/A
	friendly_name	String	N/A
	attributes	String	N/A
	is_online	Boolean	N/A
	is_notifiable	Boolean	N/A
	date_created	Datetime	N/A
	date_updated	Datetime	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	url	String	N/A
	links	Struct	N/A
Conversations-Role	sid	String	N/A
	account_sid	String	N/A
	chat_service_sid	String	N/A
	friendly_name	String	N/A
	type	String	N/A
	permissions	String	N/A
	date_created	Datetime	N/A
	date_updated	Datetime	N/A
	url	String	N/A
Conversations-Configuration	account_sid	Long	N/A
	default_chat_service_sid	String	N/A
	default_messaging_service_sid	String	N/A
	default_inactive_timer	String	N/A
	default_closed_timer	String	N/A
	url	String	N/A
	links	Map	N/A
Conversations-AddressConfiguration	sid	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	account_sid	String	N/A
	type	String	N/A
	address	String	N/A
	friendly_name	String	N/A
	auto_creation	Struct	N/A
	date_created	Datetime	N/A
	date_updated	Datetime	N/A
	url	String	N/A
	address_country	String	N/A
	AutoCreation.Enabled	Boolean	N/A
	AutoCreation.Type	String	N/A
	AutoCreation.ConversationServiceSid	String	N/A
	AutoCreation.WebhookUrl	String	N/A
	AutoCreation.WebhookMethod	String	N/A
	AutoCreation.WebhookFilters	List	N/A
	AutoCreation.StudioFlowSid	String	N/A
	AutoCreation.StudioRetryCount	Integer	N/A

엔터티	필드	데이터 유형	지원되는 연산자
Conversations-WebhookConfiguration	account_sid	String	N/A
	method	String	N/A
	filters	List	N/A
	pre_webhook_url	String	N/A
	post_webhook_url	String	N/A
	target	String	N/A
	url	String	N/A
Converations-ParticipantConversation	account_sid	String	N/A
	chat_service_sid	String	N/A
	participant_sid	String	N/A
	participant_user_sid	String	N/A
	participant_identity	String	N/A
	participant_messaging_binding	Struct	N/A
	Conversation_sid	String	N/A
	conversation_unique_name	String	N/A
	conversation_friendly_name	String	N/A
	conversation_attributes	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	conversation_date_created	Datetime	N/A
	conversation_date_updated	Datetime	N/A
	conversation_created_by	String	N/A
	conversation_state	String	N/A
	conversation_timers	Struct	N/A
	links	Map	N/A
	address	String	=
	identity	String	=
Conversation-Credentials	sid	String	N/A
	account_sid	String	N/A
	friendly_name	String	N/A
	type	String	N/A
	sandbox	String	N/A
	date_created	Datetime	N/A
	dated_updated	Datetime	N/A
	url	String	N/A
	certificate	String	N/A
	private_key	String	N/A
	api_key	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	secret	String	N/A
Conversations-ConversationService	sid	String	N/A
	account_sid	String	N/A
	friendly_name	String	N/A
	date_created	Datetime	N/A
	date_updated	Datetime	N/A
	url	String	N/A
	links	Map	N/A

분할 쿼리

분할을 지원하는 필드:

Twilio에서 DateTime 데이터 형식 필드는 필드 기반 분할을 지원합니다.

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  "2024-05-01T20:55:02.000Z"
  ```

- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예시:

```
twilio_read = glueContext.create_dynamic_frame.from_options(
    connection_type="twilio",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "sms-message",
        "API_VERSION": "2010-04-01",
        "PARTITION_FIELD": "date_sent"
        "LOWER_BOUND": "2024-05-01T20:55:02.000Z"
        "UPPER_BOUND": "2024-06-01T20:55:02.000Z"
        "NUM_PARTITIONS": "10"
    }
```

Twilio 연결 옵션

다음은 Twilio의 연결 옵션입니다.

- ENTITY_NAME(문자열) - (필수) 읽기에 사용됩니다. Twilio에서의 객체 이름입니다.
- EDGE_LOCATION(문자열)-(필수) 유효한 Twilio 엣지 로케이션입니다.
- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용할 Twilio Rest API 버전입니다. Twilio는 'v1'과 '2010-04-01'의 API 버전 두 가지를 지원합니다.
- SELECTED_FIELDS(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- INSTANCE_URL(문자열) - (필수) 읽기에 사용됩니다. 유효한 Twilio 인스턴스 URL입니다.

Twilio 커넥터의 제한 사항 및 참고 사항

다음은 Twilio 커넥터의 제한 사항입니다.

- 레코드 기반 분할을 지원하지 않으며 Twilio에서 총 레코드 수를 검색하도록 프로비저닝되지 않습니다.
- `date_sent`, `start_time` 및 `end_time` 필드는 날짜/시간 데이터 형식이지만 필터링할 때는 날짜 값만 지원합니다(시간 구성 요소는 고려되지 않음).
- 'from' 또는 'to' 필드를 필터링하는 작업은 값에 접두사(예: 프로토콜 또는 레이블)가 포함되지 않은 경우에만 작동합니다. 접두사가 있는 경우 해당 필드에 대한 필터링이 작동하지 않습니다. 예를 들어 "to": "whatsapp:+14xxxxxxxxxx"를 필터로 전달하면 Twilio는 응답을 반환하지 않습니다. "to": "+14xxxxxxxx"로 전달해야 하고, 레코드가 있는 경우 반환됩니다.
- `conversation-participant-conversation` 엔터티를 쿼리할 때 'identity' 필드 필터는 필수입니다.

AWS Glue Studio의 Vertica 입력에 연결

AWS Glue에서는 Vertica를 기본으로 지원합니다. AWS Glue Studio에는 Vertica에 연결하고, 데이터 통합 작업을 작성하며, AWS Glue Studio 서버리스 Spark 런타임에서 실행할 수 있는 시각적 인터페이스가 있습니다.

AWS Glue Studio는 Vertica용 통합 연결을 생성합니다. 자세한 내용은 고려 사항 단원을 참조하십시오.

주제

- Vertica 연결 생성
- Vertica 소스 노드 생성
- Vertica 대상 노드 생성
- 고급 옵션

Vertica 연결 생성

사전 조건:

- *`tempS3Path`*에서 참조하여 데이터베이스에서 읽고 쓸 때 임시 스토리지로 사용하는 Amazon S3 버킷 또는 폴더.

> **Note**
>
> AWS Glue 작업 데이터 미리 보기에서 Vertica를 사용하는 경우 임시 파일이 *tempS3Path*에서 자동으로 제거되지 않을 수 있습니다. 임시 파일을 제거하려면 데이터 미리 보기 창에서 세션 종료를 선택하여 데이터 미리 보기 세션을 바로 종료합니다.
> 데이터 미리 보기 세션이 바로 종료되도록 보장할 수 없는 경우 이전 데이터를 제거하도록 Amazon S3 수명 주기 구성을 설정하는 것이 좋습니다. 최대 작업 런타임에 여백을 더한 기준으로 49시간이 지난 데이터는 제거하는 것이 좋습니다. Amazon S3 수명 주기 구성에 대한 자세한 내용은 Amazon S3 설명서에서 스토리지 수명 주기 관리를 참조하십시오.

- Amazon S3 경로에 대한 적절한 권한이 있는 IAM 정책을 AWS Glue 작업 역할과 연결할 수 있습니다.
- Vertica 인스턴스가 Amazon VPC에 있는 경우, 퍼블릭 인터넷을 통과하는 트래픽 없이 AWS Glue 작업이 Vertica 인스턴스와 통신할 수 있도록 Amazon VPC를 구성하십시오.

  Amazon VPC에서 작업을 실행하는 동안 AWS Glue가 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 Vertica 인스턴스와 이 위치 간의 네트워크 트래픽을 허용하도록 Amazon VPC를 구성해야 합니다. 작업을 수행하려면 Vertica 클라이언트 포트(기본값 5433)와의 TCP 연결을 설정해야 합니다. 네트워크 레이아웃에 따라 보안 그룹 규칙, 네트워크 ACL, NAT 게이트웨이 및 피어링 연결을 변경해야 할 수도 있습니다.

Vertica에 대한 연결 구성하는 방법:

1. AWS Secrets Manager에서 Vertica 보안 인증 정보, *verticaUsername*, *verticaPassword*을 사용하여 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

   - 키/값 페어를 선택하면 값 *verticaUsername*이 포함된 키 `user`에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *verticaPassword*이 포함된 키 `password`에 대한 페어를 생성합니다.
2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때 Vertica를 선택합니다.
   - Vertica 호스트를 선택할 때 Vertica 설치의 호스트 이름을 제공합니다.

- Vertica 포트를 선택하면 해당 포트를 통해 Vertica 설치를 사용할 수 있습니다.
- AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

3. 다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.

   - Amazon VPC에서 AWS에 호스팅된 Vertica 인스턴스의 경우
     - Vertica 보안 보안 인증 정보를 정의하는 Amazon VPC 연결 정보를 AWS Glue 연결에 제공하십시오. 연결을 만들거나 업데이트할 때 네트워크 옵션에서 VPC, 서브넷 및 보안 그룹을 설정합니다.

AWS Glue 작업을 실행하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업과 권한과 연결된 IAM 역할을 *tempS3Path*에 부여합니다.
- AWS Glue 작업 권한과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.

Vertica 소스 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Vertica 연결 생성”에서 설명한 대로 Vertica 유형 AWS Glue 데이터 카탈로그 연결인 *connectionName*과 임시 Amazon S3 위치인 *tempS3Path*를 사용합니다.
- 읽으려는 Vertica 테이블, *tableName* 또는 쿼리 *targetQuery*.

Vertica 데이터 소스 추가

데이터 소스 - Vertica 노드 추가하는 방법:

1. Vertica 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 Vertica 연결 생성을 선택합니다. 자세한 내용은 이전the section called “Vertica 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.
2. 테이블이 포함된 데이터베이스를 선택합니다.
3. Amazon S3에서 스테이징 영역을 선택하고 *tempS3Path*에 S3A URI를 입력합니다.
4. Vertica 소스를 선택합니다.

   - 단일 테이블 선택 - 단일 테이블에서 모든 데이터에 액세스할 수 있습니다.

- 사용자 지정 쿼리 입력 - 사용자 지정 쿼리를 기반으로 여러 테이블의 데이터 세트에 액세스할 수 있습니다.

5. 단일 테이블을 선택한 경우 *tableName*을 입력하고 선택적으로 스키마를 선택합니다.

   사용자 지정 쿼리를 선택한 경우 SQL SELECT 쿼리를 입력하고 선택적으로 스키마를 선택합니다.

6. 사용자 지정 Vertica 속성에서 필요한 경우 파라미터와 값을 입력합니다.

Vertica 대상 노드 생성

필수 전제 조건

- 이전 섹션 the section called “Vertica 연결 생성”에서 설명한 대로 Vertica 유형 AWS Glue 데이터 카탈로그 연결인 *connectionName*과 임시 Amazon S3 위치인 *tempS3Path*를 사용합니다.

Vertica 데이터 대상 추가

데이터 대상 - Vertica 노드를 추가하는 방법:

1. Vertica 데이터 소스의 연결을 선택합니다. 생성했으므로 드롭다운에서 사용할 수 있을 것입니다. 연결을 생성해야 하는 경우 Vertica 연결 생성을 선택합니다. 자세한 내용은 이전the section called “Vertica 연결 생성” 섹션을 참조하세요.

   연결을 선택한 후에는 속성 보기를 클릭하여 연결 속성을 볼 수 있습니다.

2. 테이블이 포함된 데이터베이스를 선택합니다.
3. Amazon S3에서 스테이징 영역을 선택하고 *tempS3Path*에 S3A URI를 입력합니다.
4. *tableName*을 입력하고 선택적으로 스키마를 선택합니다.
5. 사용자 지정 Vertica 속성에서 필요한 경우 파라미터와 값을 입력합니다.

고급 옵션

Vertica 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “수직 연결”를 참조하세요.

WooCommerce에 연결

WooCommerce는 WordPress 기반 웹 사이트를 위해 빌드된 유연한 오픈 소스 소프트웨어 솔루션입니다. 일반적으로 온라인 전자 상거래 상점을 생성하는 데 사용됩니다. 이 소프트웨어 솔루션을 사용하면 누구나 일반 웹 사이트를 완전한 기능을 갖춘 온라인 상점으로 전환할 수 있습니다.

주제

- AWS Glue의 WooCommerce 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- WooCommerce 구성
- WooCommerce 연결 구성
- WooCommerce 엔터티에서 읽기
- WooCommerce 연결 옵션

AWS Glue의 WooCommerce 지원

AWS Glue에서는 다음과 같이 WooCommerce를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 WooCommerce에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 WooCommerce API 버전

다음 WooCommerce API 버전이 지원됩니다.

- v3

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
```

```
"Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

WooCommerce 구성

AWS Glue를 사용하여 WooCommerce에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- `consumerKey`와 `consumerSecret`가 있는 WooCommerce 계정이 있습니다.
- WooCommerce 계정에 유효한 라이선스가 있는 API 액세스 권한이 있습니다.

이러한 요구 사항을 충족하면 WooCommerce 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 WooCommerce에서 다른 작업을 수행하지 않아도 됩니다.

WooCommerce 연결 구성

WooCommerce에서는 사용자 지정 인증을 지원합니다. 사용자 지정 권한 부여에 필요한 API 키 생성에 대한 퍼블릭 WooCommerce 설명서는 인증-WooCommerce REST API 설명서를 참조하세요.

WooCommerce 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   - 고객 관리형 연결된 앱의 경우 보안 암호에는 연결된 앱의 소비자 보안 암호와 `consumerSecret` 및 `consumerKey`를 키로 포함해야 합니다. 참고: AWS Glue에서 연결당 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 WooCommerce를 선택합니다.
   b. 연결하려는 WooCommerce 인스턴스의 `INSTANCE_URL` 항목을 제공합니다.
   c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

   d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.
   e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

WooCommerce 엔터티에서 읽기

사전 조건

읽으려는 WooCommerce 객체입니다. 쿠폰, 주문, 제품 등과 같은 객체 이름이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Coupon	예	예	예	예	예
Coupon Total	아니요	아니요	아니요	예	아니요
Customers Total	아니요	아니요	아니요	예	아니요
Order	예	예	예	예	예
Orders Total	아니요	아니요	아니요	예	아니요
Payment Gateway	아니요	아니요	아니요	예	아니요
Product	예	예	예	예	예
Product attribute	예	예	예	예	예
Product category	예	예	예	예	예
Product review	예	예	예	예	예

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Product shipping class	예	예	예	예	예
Product tag	예	예	예	예	예
Product variation	예	예	예	예	예
Products Total	아니요	아니요	아니요	예	아니요
Report (List)	아니요	아니요	아니요	예	아니요
Reviews Total	아니요	아니요	아니요	예	아니요
Sales Report	예	아니요	아니요	예	아니요
Shipping Method	아니요	아니요	아니요	예	아니요
Shipping Zone	아니요	아니요	아니요	예	아니요
Shipping Zone Location	아니요	아니요	아니요	예	아니요
Shipping Zone Method	아니요	아니요	아니요	예	아니요
Tax Rate	예	예	예	예	예
Tax Class	아니요	아니요	아니요	예	아니요

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Top Sellers Report	예	아니요	아니요	예	아니요

예시:

```
woocommerce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="glue.spark.woocommerce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "coupon",
        "API_VERSION": "v3",
        "INSTANCE_URL": "instanceUrl"
    }
```

WooCommerce 엔터티 및 필드 세부 정보:

엔터티	필드	데이터 유형	지원되는 연산자
coupon	id	Integer	N/A
	code	String	EQUAL_TO
	amount	String	N/A
	status	String	N/A
	date_created	DateTime	N/A
	date_created_gmt	DateTime	N/A
	date_modified	DateTime	N/A
	date_modified_gmt	DateTime	N/A
	discount_type	String	N/A
	description	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	date_expires	String	N/A
	date_expires_gmt	String	N/A
	usage_count	Integer	N/A
	individual_use	Boolean	N/A
	product_id	List	N/A
	excluded_product_ids	List	N/A
	usage_limit	Integer	N/A
	usage_limit_per_user	Integer	N/A
	limit_usage_to_x_items	Integer	N/A
	free_shipping	Boolean	N/A
	product_categories	List	N/A
	excluded_product_categories	List	N/A
	exclude_sale_items	Boolean	N/A
	minimum_amount	String	N/A
	maximum_amount	String	N/A
	email_restrictions	List	N/A
	used_by	List	N/A
	meta_data	List	N/A
	context	String	EQUAL_TO

엔터티	필드	데이터 유형	지원되는 연산자
	search	String	EQUAL_TO
	after	DateTime	EQUAL_TO
	before	DateTime	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	modified_after	DateTime	EQUAL_TO
	modified_before	DateTime	EQUAL_TO
	dates_are_gmt	Boolean	EQUAL_TO
coupon-total	slug	String	N/A
	name	String	N/A
	total	Integer	N/A
customer-total	slug	String	N/A
	name	String	N/A
	total	Integer	N/A
order	id	Integer	N/A
	parent_id	Integer	N/A
	number	String	N/A
	order_key	String	N/A
	created_via	String	N/A
	status	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	currency	String	N/A
	version	String	N/A
	date_created	DateTime	N/A
	date_modified	DateTime	N/A
	discount_total	String	N/A
	discount_tax	String	N/A
	shipping_total	String	N/A
	shipping_tax	String	N/A
	cart_tax	String	N/A
	total	String	N/A
	total_tax	String	N/A
	prices_include_tax	Boolean	N/A
	customer_id	Integer	N/A
	customer_ip_address	String	N/A
	customer_user_agent	String	N/A
	customer_note	String	N/A
	billing	Struct	N/A
	shipping	Struct	N/A
	payment_method	String	N/A
	payment_method_title	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	transaction_id	String	N/A
	date_paid	DateTime	N/A
	date_completed	DateTime	N/A
	cart_hash	String	N/A
	meta_data	List	N/A
	line_items	List	N/A
	tax_lines	List	N/A
	shipping_lines	List	N/A
	fee_lines	List	N/A
	coupon_lines	List	N/A
	refunds	List	N/A
	payment_url	String	N/A
	is_editable	Boolean	N/A
	needs_payment	Boolean	N/A
	needs_processing	Boolean	N/A
	date_created_gmt	DateTime	N/A
	date_modified_gmt	DateTime	N/A
	date_completed_gmt	DateTime	N/A
	date_paid_gmt	DateTime	N/A
	currency_symbol	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	set_paid	Boolean	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	after	DateTime	EQUAL_TO
	before	DateTime	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	customer	Integer	EQUAL_TO
	product	Integer	EQUAL_TO
	dp	Integer	EQUAL_TO
	modified_before	DateTime	EQUAL_TO
	modified_after	DateTime	EQUAL_TO
	dates_are_gmt	Boolean	EQUAL_TO
order-total	slug	String	N/A
	name	String	N/A
	total	Integer	N/A
payment-gateway	title	String	N/A
	description	String	N/A
	order	String	N/A
	enabled	Boolean	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	method_title	String	N/A
	method_description	String	N/A
	method_supports	List	N/A
	settings	String	N/A
	needs_setup	Boolean	N/A
	post_install_scripts	List	N/A
	settings_url	String	N/A
	connection_url	String	N/A
	setup_help_text	String	N/A
	required_settings_keys	List	N/A
product	id	Integer	N/A
	name	String	N/A
	type	String	EQUAL_TO
	permalink	String	N/A
	date_created	DateTime	N/A
	date_created_gmt	DateTime	N/A
	date_modified	DateTime	N/A
	date_modified_gmt	DateTime	N/A
	catalog_visibility	String	N/A
	description	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	short_description	String	N/A
	price	String	N/A
	regular_price	String	N/A
	sale_price	String	N/A
	date_on_sale_from	DateTime	N/A
	date_on_sale_from_gmt	DateTime	N/A
	date_on_sale_to	DateTime	N/A
	date_on_sale_to_gmt	DateTime	N/A
	price_html	String	N/A
	purchasable	Boolean	N/A
	total_sales	Integer	N/A
	virtual	Boolean	N/A
	downloadable	Boolean	N/A
	downloads	List	N/A
	download_limit	Integer	N/A
	download_expiry	Integer	N/A
	external_url	String	N/A
	button_text	String	N/A
	tax_status	String	N/A
	manage_stock	Boolean	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	stock_quantity	Integer	N/A
	backorders	String	N/A
	backorders_allowed	Boolean	N/A
	backordered	Boolean	N/A
	sold_individually	Boolean	N/A
	weight	String	N/A
	dimensions	Struct	N/A
	shipping_required	Boolean	N/A
	shipping_taxable	Boolean	N/A
	shipping_class_id	Integer	N/A
	reviews_allowed	Boolean	N/A
	average_rating	String	N/A
	rating_count	Integer	N/A
	related_ids	List	N/A
	upsell_ids	List	N/A
	cross_sell_ids	List	N/A
	parent_id	Integer	N/A
	purchase_note	String	N/A
	categories	List	N/A
	tags	List	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	images	List	N/A
	attributes	List	N/A
	default_attributes	List	N/A
	variation	List	N/A
	grouped_products	List	N/A
	menu_order	Integer	N/A
	meta_data	List	N/A
	low_stock_amount	Integer	N/A
	jetpack_publicize_connections	List	N/A
	jetpack-related-posts	List	N/A
	jetpack_likes_enabled	Boolean	N/A
	jetpack_sharing_enabled	Boolean	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	after	DateTime	EQUAL_TO
	before	DateTime	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	slug	String	EQUAL_TO

엔터티	필드	데이터 유형	지원되는 연산자
	status	String	EQUAL_TO
	sku	String	EQUAL_TO
	featured	Boolean	EQUAL_TO
	tag	String	EQUAL_TO
	shipping_class	String	EQUAL_TO
	tax_class	String	EQUAL_TO
	on_sale	Boolean	EQUAL_TO
	stock_status	String	EQUAL_TO
	has_options	Boolean	N/A
	modified_after	DateTime	EQUAL_TO
	modified_before	DateTime	EQUAL_TO
	dates_are_gmt	Boolean	EQUAL_TO
	category	String	EQUAL_TO
	attribute	String	EQUAL_TO
	min_price	String	EQUAL_TO
	max_price	String	EQUAL_TO
product-attribute	id	Integer	N/A
	name	String	N/A
	slug	String	N/A
	type	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	order_by	String	N/A
	has_archives	Boolean	N/A
	context	String	EQUAL_TO
product-attribute-term	id	Integer	N/A
	name	String	N/A
	slug	String	N/A
	description	String	N/A
	menu_order	Integer	N/A
	count	Integer	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	hide_empty	Boolean	EQUAL_TO
	parent	Integer	EQUAL_TO
	product	Integer	EQUAL_TO
product-category	id	Integer	N/A
	name	String	N/A
	slug	String	EQUAL_TO
	description	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	display	String	N/A
	image	Struct	N/A
	menu_order	Integer	N/A
	count	Integer	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	hide_empty	Boolean	EQUAL_TO
	parent	Integer	EQUAL_TO
	product	Integer	EQUAL_TO
product-review	id	Integer	N/A
	date_created	DateTime	N/A
	date_created_gmt	DateTime	N/A
	product_id	Integer	N/A
	product_name	String	N/A
	product_permalink	String	N/A
	review	String	N/A
	rating	Integer	N/A
	verified	Boolean	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	reviewer	String	N/A
	reviewer_email	String	N/A
	reviewer_avatar_urls	Struct	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	after	DateTime	EQUAL_TO
	before	DateTime	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	status	String	EQUAL_TO
product-shipping-class	id	Integer	N/A
	name	String	N/A
	slug	String	EQUAL_TO
	description	String	N/A
	count	Integer	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	hide_empty	String	EQUAL_TO

엔터티	필드	데이터 유형	지원되는 연산자
	product	Integer	EQUAL_TO
product-tag	id	Integer	N/A
	name	String	N/A
	slug	String	EQUAL_TO
	description	String	N/A
	count	Integer	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	hide_empty	Boolean	EQUAL_TO
	product	Integer	EQUAL_TO
product-total	slug	String	N/A
	name	String	N/A
	total	Integer	N/A
product-variation	id	Integer	N/A
	date_created	DateTime	N/A
	date_created_gmt	DateTime	N/A
	date_modified	DateTime	N/A
	date_modified_gmt	DateTime	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	description	String	N/A
	permalink	String	N/A
	price	String	N/A
	regular_price	String	N/A
	sale_price	String	N/A
	date_on_sale_from	DateTime	N/A
	date_on_sale_from_gmt	DateTime	N/A
	date_on_sale_to	DateTime	N/A
	date_on_sale_to_gmt	DateTime	N/A
	purchasable	Boolean	N/A
	virtual	Boolean	N/A
	downloadable	Boolean	N/A
	downloads	List	N/A
	download_limit	Integer	N/A
	download_expiry	Integer	N/A
	tax_status	String	N/A
	manage_stock	Boolean	N/A
	stock_quantity	Integer	N/A
	backorders	String	N/A
	backorders_allowed	Boolean	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	backordered	Boolean	N/A
	low_stock_amount	Integer	N/A
	weight	String	N/A
	dimensions	Struct	N/A
	shipping_class	String	N/A
	shipping_class_id	Integer	N/A
	image	Struct	N/A
	attributes	List	N/A
	menu_order	Integer	N/A
	meta_data	List	N/A
	context	String	EQUAL_TO
	search	String	EQUAL_TO
	after	DateTime	EQUAL_TO
	before	DateTime	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	slug	String	EQUAL_TO
	status	String	EQUAL_TO
	sku	String	EQUAL_TO
	tax_class	String	EQUAL_TO

엔터티	필드	데이터 유형	지원되는 연산자
	on_sale	Boolean	EQUAL_TO
	min_price	String	EQUAL_TO
	max_price	String	EQUAL_TO
	stock_status	String	EQUAL_TO
report	slug	String	N/A
	description	String	N/A
review-total	slug	String	N/A
	name	String	N/A
	total	Integer	N/A
sales-report	total_sales	String	N/A
	net_sales	String	N/A
	average_sales	String	N/A
	total_orders	Integer	N/A
	total_items	Integer	N/A
	total_tax	String	N/A
	total_shipping	String	N/A
	total_refunds	Integer	N/A
	total_discount	String	N/A
	totals_grouped_by	String	N/A
	totals	Struct	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	total_customers	Integer	N/A
	context	String	EQUAL_TO
	period	String	EQUAL_TO
	date_min	Date	EQUAL_TO
	date_max	Date	EQUAL_TO
shipping-method	id	String	N/A
	title	String	N/A
	description	String	N/A
shipping-zone	id	Integer	EQUAL_TO
	name	String	N/A
	order	Integer	N/A
shipping-zone-location	code	String	N/A
	type	String	N/A
shipping-zone-method	instance-id	Integer	N/A
	id	Integer	EQUAL_TO
	title	String	N/A
	order	Integer	N/A
	enabled	Boolean	N/A
	method_id	String	N/A
	method_title	String	N/A

엔터티	필드	데이터 유형	지원되는 연산자
	method_description	String	N/A
	settings	Struct	N/A
tax-class	slug	String	N/A
	name	String	N/A
tax-rate	id	Integer	N/A
	country	String	N/A
	state	String	N/A
	postcode	String	N/A
	city	String	N/A
	postcodes	List	N/A
	cities	List	N/A
	rate	String	N/A
	name	String	N/A
	priority	Integer	N/A
	compound	Boolean	N/A
	shipping	Boolean	N/A
	context	String	EQUAL_TO
	order	String	EQUAL_TO
	orderby	String	EQUAL_TO
	class	String	EQUAL_TO

엔터티	필드	데이터 유형	지원되는 연산자
top-seller-report	name	String	N/A
	product_id	Integer	N/A
	quantity	Integer	N/A
	context	String	EQUAL_TO
	period	String	EQUAL_TO
	date_min	Date	EQUAL_TO
	date_max	Date	EQUAL_TO

> **Note**
>
> 커넥터의 응답에서 Struct 및 List 데이터 유형은 String 데이터 유형으로 변환되며, DateTime 데이터 유형은 타임스탬프로 변환됩니다.

분할 쿼리

레코드 기반 분할:

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

레코드 기반 분할에서는 존재하는 총 레코드 수를 WooCommerce API에서 쿼리하고 제공된 NUM_PARTITIONS 번호로 나눕니다. 그런 다음, 결과 레코드 수를 각 하위 쿼리에서 동시에 가져옵니다.

- NUM_PARTITIONS: 파티션 수.

다음 엔터티는 레코드 기반 분할을 지원합니다.

- coupon

- order
- product
- product-attribute
- product-attribute-term
- product-category
- product-review
- product-shipping-class
- product-tag
- product-variation
- tax-rate

예시:

```
woocommerce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="glue.spark.woocommerce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "coupon",
        "API_VERSION": "v3",
        "INSTANCE_URL": "instanceUrl"
        "NUM_PARTITIONS": "10"
    }
```

레코드 기반 분할:

원본 쿼리가 Spark 태스크에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 분할됩니다.

- NUM_PARTITIONS: 파티션 수.

예시:

```
WooCommerce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="WooCommerce",
    connection_options={
        "connectionName": "connectionName",
        "REALMID": "1234567890123456789",
```

```
        "ENTITY_NAME": "Bill",
        "API_VERSION": "v3",
        "NUM_PARTITIONS": "10"
    }
```

WooCommerce 연결 옵션

다음은 WooCommerce의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. WooCommerce에서의 객체 이름입니다.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용할 WooCommerce Rest API 버전입니다.
- `REALM_ID`(문자열)-요청을 보내는 개별 WooCommerce Online 회사를 식별하는 ID입니다.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- `INSTANCE_URL`(문자열)-(필수) 형식이 https://<instance>.wpcomstaging.com이며 유효한 WooCommerce 인스턴스 URL
- `NUM_PARTITIONS`(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.

Zendesk에 연결

Zendesk는 고객 서비스 포털, 지식 기반 및 온라인 커뮤니티를 구축하기 위한 사용자 지정 가능한 도구를 제공하는 클라우드 기반 헬프 데스크 관리 솔루션입니다.

주제

- AWS Glue Zendesk 지원
- 연결 생성 및 사용 API 작업이 포함된 정책
- Zendesk 구성
- Zendesk 연결 구성
- Zendesk 엔터티에서 읽기
- Zendesk 연결 옵션
- 제한 사항

AWS Glue Zendesk 지원

AWS Glue 는 다음과 같이 Zendesk를 지원합니다.

소스로 지원되나요?

예. 작업을 사용하여 AWS Glue ETL Zendesk에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Zendesk API 버전

다음 Zendesk API 버전이 지원됩니다.

- v2

연결 생성 및 사용 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려면 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole - 다양한 AWS Glue 프로세스가 사용자를 대신하여 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon 가 포함됩니다EC2. 이 정책에 지정된 리소스의 이름 지정 규칙을 따르는 경우 AWS Glue 프로세스에 필요한 권한이 있습니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS 관리 콘솔을 사용할 때 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue 콘솔 사용자에게 연결됩니다.

Zendesk 구성

AWS Glue 를 사용하여 Zendesk에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Zendesk 계정이 있습니다. 자세한 내용은 Zendesk 계정 생성 단원을 참조하십시오.
- Zendesk 계정에 API 액세스할 수 있습니다.
- Zendesk 계정을 사용하면 연결된 앱을 설치할 수 있습니다.

이러한 요구 사항을 충족하면 Zendesk 계정에 AWS Glue 연결할 준비가 된 것입니다.

Zendesk 계정 생성

Zendesk 계정을 생성하려면:

1. https://www.zendesk.com/in/등록으로 이동/
2. 작업 이메일, 이름, 성, 전화번호, 직책, 회사 이름, 회사 내 직원 수, 암호 및 선호하는 언어와 같은 세부 정보를 입력합니다. 그런 다음 평가판 등록 완료를 선택합니다.
3. 계정이 생성되면 수신한 확인 링크를 작성하여 이메일 주소를 확인합니다.
4. 작업 이메일 주소가 확인되면 Zendesk 계정으로 리디렉션됩니다. 원하는 플랜에 대해 Zendesk 구매 옵션을 선택합니다. 참고: Zendesk 커넥터의 경우 Suite Enterprise 플랜을 구매하는 것이 좋습니다.

클라이언트 앱 및 OAuth 2.0 보안 인증 생성

클라이언트 앱 및 OAuth 2.0 보안 인증 정보를 생성하려면:

1. OAuth 2.0 앱을 생성하려는 Zendesk 계정에 https://www.zendesk.com/in/로그인/
2. 기어 아이콘을 클릭합니다. 관리자 센터로 이동 링크를 선택하여 관리자 센터 페이지를 엽니다.
3. 왼쪽 사이드바에서 앱 및 통합을 선택한 다음 APIs > Zendesk API를 선택합니다.
4. Zendesk API 페이지에서 OAuth 클라이언트 탭을 선택합니다.
5. 오른쪽에서 Oauth Client 추가를 선택합니다.
6. 클라이언트를 생성하려면 다음 필드를 작성합니다.
   a. 클라이언트 이름 - 앱의 이름을 입력합니다. 이 이름은 사용자가 애플리케이션에 대한 액세스 권한을 부여하라는 요청을 받고 Zendesk에 액세스할 수 있는 타사 앱 목록을 확인할 때 표시되는 이름입니다.
   b. 설명 - 선택 사항. 사용자에게 액세스 권한을 부여하라는 메시지가 표시될 때 사용자에게 표시되는 앱에 대한 간단한 설명입니다.
   c. 회사 - 선택 사항입니다. 애플리케이션에 대한 액세스 권한을 부여하라는 메시지가 표시될 때 사용자에게 표시되는 회사 이름입니다. 이 정보는 액세스 권한을 부여하는 대상을 이해하는 데 도움이 될 수 있습니다.
   d. 로고 - 선택 사항. 애플리케이션에 대한 액세스 권한을 부여하라는 메시지가 표시될 때 사용자에게 표시되는 로고입니다. 이미지는 JPG, GIF또는 일 수 있습니다PNG. 최상의 결과를 얻으려면 사각형 이미지를 업로드하세요. 권한 부여 페이지에 맞게 크기가 조정됩니다.
   e. 고유 식별자 - 이 필드는 앱에 입력한 이름의 형식이 변경된 버전으로 자동 채워집니다. 원하는 경우 변경할 수 있습니다.
   f. 리디렉션 URLs - Zendesk가 사용자의 애플리케이션 액세스 권한 부여 결정을 보내는 데 사용해야 URLs 하는 URL 또는 를 입력합니다.

      예: https://us-east-1.console.aws.amazon.com/gluestudio/oauth
7. 저장을 클릭합니다.
8. 페이지가 새로 고쳐지면 새 미리 채워진 보안 암호 필드가 하단에 나타납니다. 사양에 지정된 'client_secret' 값입니다OAuth2. 보안 암호 값을 클립보드에 복사하고 안전한 곳에 저장합니다. 참고: 문자는 텍스트 상자의 너비를 넘어 확장될 수 있으므로 복사하기 전에 모든 항목을 선택해야 합니다.
9. 저장을 클릭합니다.

Zendesk 연결 구성

Zendesk 커넥터는 권한 부여 코드 권한 부여 유형을 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 기본적으로 Zendesk 인스턴스 URL을 제외한 OAuth 관련 정보를 제공할 필요가 없는 AWS Glue 자체 연결된 앱(AWS Glue 관리형 클라이언트 애플리케이션)에 의존할 수 있습니다. AWS Glue 콘솔은 사용자를 Zendesk로 리디렉션합니다. 사용자가 로그인하고 Zendesk 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
- 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Zendesk에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Zendesk로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 액세스 토큰을 생성합니다. 액세스 토큰은 만료되지 않습니다.

권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Zendesk 설명서는 권한 부여 유형용 OAuth 토큰을 참조하세요.

Zendesk 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. AuthorizationCode 권한 부여 유형의 경우 고객 관리형 연결된 앱에서는 시크릿에 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 시크릿이 포함되어야 합니다.
   b. 참고: AWS Glue에서 연결당 시크릿을 생성해야 합니다.
2. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Zendesk를 선택합니다.
   b. 연결하려는 Zendesk의 INSTANCE_URL을 제공합니다.
   c. Zendesk 환경을 제공합니다.
   d. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
```

```
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

e. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

f. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

3. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Zendesk 엔터티에서 읽기

사전 조건

읽으려는 Zendesk 객체. 다음 표에 언급된 대로 티켓, 사용자 또는 문서와 같은 객체 이름이 필요합니다.

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
티켓	Y	Y	Y	Y	N
User	Y	Y	Y	Y	N
조직	Y	Y	Y	Y	N
Article	Y	Y	N	Y	N
티켓 이벤트	Y	Y	N	Y	N
티켓 지표 이벤트	Y	Y	N	Y	N
티켓 설명	Y	Y	Y	Y	N
티켓 필드	Y	Y	N	Y	N

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
티켓 지표	Y	Y	N	Y	N
티켓 활동	Y	Y	N	Y	N
티켓 건너뛰기	N	Y	N	Y	N
그룹	Y	Y	Y	Y	N
그룹 멤버십	N	Y	Y	Y	N
만족도 등급	Y	Y	N	Y	N
보기	Y	Y	Y	Y	N
트리거	Y	Y	Y	Y	N
트리거 카테고리	N	Y	Y	Y	N
매크로	Y	Y	Y	Y	N
자동화	N	Y	Y	Y	N

예제:

```
Zendesk_read = glueContext.create_dynamic_frame.from_options(
    connection_type="Zendesk",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v2"
    }
```

Zendesk 엔터티 및 필드 세부 정보:

개체	필드	데이터 유형	지원되는 연산자	설명
문서	url	String		
	id	Long		
	author_id	Long		
	본문	String		
	comments_ disabled	불		
	draft	불		
	edited_at	DateTime		
	html_url	String		
	label_names	나열		
	locale	String	EQUAL_TO	
	outdated	불		
	outdated_locales	나열		
	permissio n_group_id	Long		
	position	Integer		
	promoted	불		
	section_id	Long		
	source_locale	String		
	name	String		
	title	String		

개체	필드	데이터 유형	지원되는 연산자	설명
	user_segment_id	Long		
	content_tags_id	나열		
	vote_count	Integer		
	vote_sum	Integer		
	created_at	DateTime		
	updated_at	DateTime	EQUAL_TO	
	label_name	String	EQUAL_TO	
그룹	url	String		
	id	Long		
	is_public	불		
	name	String		
	설명	String		
	기본값	불		
	삭제됨	불		
	created_at	DateTime		
	updated_at	DateTime		
	exclude_deleted	불	EQUAL_TO	
자동화	url	String		
	id	Long		
	title	String		

개체	필드	데이터 유형	지원되는 연산자	설명
	활성화	불		
	created_at	DateTime		
	updated_at	DateTime		
	기본값	불		
	actions	나열		
	positions	Integer		
	조건	구조체		
	raw_title	String		
group-membership	url	String		
	id	Long		
	user_id	Long		
	group_id	Long		
	기본값	불		
	created_at	DateTime		
	updated_at	DateTime		
매크로	url	String		
	id	Long		
	title	String		
	활성화	불	EQUAL_TO	
	created_at	DateTime		

개체	필드	데이터 유형	지원되는 연산자	설명
	updated_at	DateTime		
	기본값	불		
	actions	나열		
	position	Integer		
	설명	String		
	raw_title	String		
	제한	구조체		
	access	String	EQUAL_TO	
	category	Integer	EQUAL_TO	
	group_id	Long	EQUAL_TO	
	only_viewable	불	EQUAL_TO	
조직	url	String		
	id	Long		
	external_id	String		
	name	String		
	domain_names	나열		
	details	String		
	notes	String		
	group_id	Long		
	shared_tickets	불		

개체	필드	데이터 유형	지원되는 연산자	설명
	shared_comments	불		
	tags	나열		
	organization_fields	구조체		
	created_at	DateTime		
	updated_at	DateTime	EQUAL_TO	
	DML_STATUS	String		레코드의 생성, 업데이트 및 삭제 상태를 추적하는 데 사용되는 사용자 정의 필드.
satisfaction-rating	url	String		
	id	Long		
	assignee_id	Long		
	설명	String		
	group_id	Long		
	reason	String		
	reason_code	Integer		
	reason_id	Long		
	requester_id	Long		
	점수	String	EQUAL_TO	
	ticket_id	Integer		

개체	필드	데이터 유형	지원되는 연산자	설명
	created_at	DateTime		
	updated_at	DateTime	EQUAL_TO	
	start_time	DateTime	EQUAL_TO	
	end_time	DateTime	EQUAL_TO	
	DML_STATUS	String		레코드의 생성, 업데이트 및 삭제 상태를 추적하는 데 사용되는 사용자 정의 필드.
ticket-activity	액터	구조체		
	actor_id	Long		
	created_at	DateTime		
	id	Long		
	객체	구조체		
	대상	구조체		
	title	String		
	updated_at	DateTime		
	url	String		
	사용자	구조체		
	user_id	Long		
	동사	String		
	since	DateTime	EQUAL_TO	

개체	필드	데이터 유형	지원되는 연산자	설명
ticket-comment	id	Long		
	type	String		
	author_id	Long		
	본문	String		
	html_body	String		
	plain_body	String		
	퍼블릭	불		
	attachments	나열		
	audit_id	Long		
	via	구조체		
	created_at	DateTime		
	metadata	구조체		
	ticket_id	Integer	EQUAL_TO	
	include_inline_images	불	EQUAL_TO	
ticket-events	id	Long		
	ticket_id	Long		
	타임스탬프	Long		
	created_at	DateTime		
	updater_id	Long		
	child_events	나열		

개체	필드	데이터 유형	지원되는 연산자	설명
	via	String		
	시스템	구조체		
	event_type	String		
	comment_p resent	불		
	comment_public	불		
	via_reference_id	Long		
	created_at	DateTime	EQUAL_TO	
	DML_STATUS	String		레코드의 생성, 업데이트 및 삭제 상태를 추적하는 데 사용되는 사용자 정의 필드.
ticket-field	url	String		
	id	Long		
	type	String		
	title	String		
	raw_title	String		
	설명	String		
	raw_description	String		
	position	Integer		
	활성화	불		

개체	필드	데이터 유형	지원되는 연산자	설명
	필수	불		
	collapsed_for_agents	불		
	regexp_for_validation	String		
	title_in_portal	String		
	raw_title_in_portal	String		
	visible_in_portal	불		
	editable_on_portal	불		
	required_in_portal	불		
	tag	String		
	created_at	DateTime		
	updated_at	DateTime		
	removable	불		
	agent_description	String		
	custom_field_options	나열		
	custom_statuses	나열		
	relationship_filter	구조체		

개체	필드	데이터 유형	지원되는 연산자	설명
	relationship_target_type	String		
	sub_type_id	Integer		
	system_field_options	나열		
	locale	String	EQUAL_TO	
ticket-metric-events	id	Long		
	시간	DateTime	EQUAL_TO	
	ticket_id	Integer		
	지표	String		
	instance-id	Integer		
	type	String		
	DML_STATUS	String	EQUAL_TO	레코드의 생성, 업데이트 및 삭제 상태를 추적하는 데 사용되는 사용자 정의 필드.
ticket-metric	url	String		
	id	Long		
	ticket_id	Integer		
	created_at	DateTime		
	updated_at	DateTime		
	group_stations	Integer		

개체	필드	데이터 유형	지원되는 연산자	설명
	assignee_stations	Integer		
	reopens	Integer		
	replies	Integer		
	assignee_updated_at	DateTime		
	requester_updated_at	DateTime		
	initially_assigned_at	DateTime		
	assigned_at	DateTime		
	solved_at	DateTime		
	last_comment_added_at	DateTime		
	reply_time_in_minutes	구조체		
	first_resolution_time_in_minutes	구조체		
	full_resolution_time_in_minutes	구조체		
	agent_wait_time_in_minutes	구조체		

개체	필드	데이터 유형	지원되는 연산자	설명
	requester_wait_time_in_minutes	구조체		
	on_hold_time_in_seconds	구조체		
	reply_time_in_seconds	구조체		
	custom_status_updated_at	DateTime		
ticket-skip	created_at	DateTime		
	id	Long		
	reason	String		
	ticket	구조체		
	ticket_id	Integer		
	updated_at	DateTime		
	user_id	Long		
tickets	url	String		
	id	Long		
	external_id	String	EQUAL_TO	
	type	String		
	subject	String		
	raw_subject	String		

개체	필드	데이터 유형	지원되는 연산자	설명
	설명	String		
	우선순위	String		
	status	String		
	수신자	String		
	요청자	구조체		
	requester_id	Long		
	submitter_id	Long		
	assignee_id	Long		
	organization_id	Long		
	group_id	Long		
	collaborator_ids	나열		
	emails_cc_ids	나열		
	follower_ids	나열		
	forum_topic_id	Ling		
	problem_id	Long		
	has_incidents	불		
	due_at	DateTime		
	tags	나열		
	via	구조체		
	custom_fields	나열		

개체	필드	데이터 유형	지원되는 연산자	설명
	satisfaction_rating	구조체		
	sharing_agreement_ids	나열		
	followup_ids	나열		
	via_followup_source_id	Long		
	ticket_form_id	Long		
	brand_id	Long		
	allow_channelback	불		
	allow_attachments	불		
	is_public	불		
	from_messaging_channel	불		
	created_at	DateTime		
	updated_at	DateTime	EQUAL_TO	
	assignee_email	String		
	attribute_value_ids	나열		
	collaborators	나열		
	설명	구조체		

개체	필드	데이터 유형	지원되는 연산자	설명
	custom_status_id	Long		
	email_ccs	구조체		
	followers	구조체		
	macro_id	Long		
	macros_ids	나열		
	metadata	구조체		
	safe_update	불		
	updated_stamp	DateTime		
	via_id	Long		
	voice_comment	구조체		
	DML_STATUS	String		레코드의 생성, 업데이트 및 삭제 상태를 추적하는 데 사용되는 사용자 정의 필드.
trigger-category	url	String		
	id	String		
	name	String		
	updated_at	DateTime		
	created_at	DateTime		
	position	Integer		
트리거	url	String		

개체	필드	데이터 유형	지원되는 연산자	설명
	id	Long		
	title	String		
	활성화	불	EQUAL_TO	
	updated_at	DateTime		
	created_at	DateTime		
	기본값	불		
	actions	나열		
	조건	구조체		
	설명	String		
	position	Integer		
	raw_title	String		
	category_id	String	EQUAL_TO	
사용자	url	String		
	id	Long		
	external_id	String	EQUAL_TO	
	이메일	String		
	활성화	불		
	별칭	String		
	chat_only	불		
	custom_roll_id	Long		

개체	필드	데이터 유형	지원되는 연산자	설명
	roll_type	Integer		
	details	String		
	last_login_at	DateTime		
	locale	String		
	locale_id	Integer		
	moderator	불		
	notes	String		
	name	String		
	only_private_comments	불		
	organization_id	Long		
	default_group_id	Long		
	phone	String		
	photo	구조체		
	remote_photo_url	String		
	restricted_agent	불		
	역할	String	EQUAL_TO	
	shared	불		
	shared_agent	불		
	tag	나열		

개체	필드	데이터 유형	지원되는 연산자	설명
	서명	String		
	suspended	불		
	ticket_restriction	String		
	time_zone	String		
	iana_time_zone			
	two_factor_auth_enabled			
	user_fields			
	verified	불		
	report_csv	불		
	created_at	DateTime		
	updated_at	DateTime	EQUAL_TO	
	permission_set	Long	EQUAL_TO	
	shared_phone_number	불		
	DML_STATUS	String		레코드의 생성, 업데이트 및 삭제 상태를 추적하는 데 사용되는 사용자 정의 필드.
view	url	String		
	id	Long		
	title	String		

개체	필드	데이터 유형	지원되는 연산자	설명
	활성화	불	EQUAL_TO	
	updated_at	DateTime		
	created_at	DateTime		
	기본값	불		
	position	Integer		
	설명	String		
	execution	구조체		
	제한	구조체		
	raw_title	String		
	조건	구조체		
	access	String	EQUAL_TO	
	group_id	Long	EQUAL_TO	

> **Note**
>
> 구조체 및 목록 데이터 유형은 커넥터의 응답에서 문자열 데이터 유형으로 변환됩니다.

쿼리 파티셔닝

파티션은 Zendesk에서 지원되지 않습니다.

Zendesk 연결 옵션

다음은 Zendesk의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Zendesk의 객체 이름입니다.

- API_VERSION(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Zendesk Rest API 버전입니다. 예: v2.
- SELECTED_FIELDS(목록<문자열>) - 기본값: 비어 있음(SELECT*). 읽기에 사용됩니다. 객체에 대해 선택할 열. 예: id, name, url, created_at
- FILTER_PREDICATE(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다. 예: group_id = 100
- QUERY(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리. 예: “SELECT id,url FROM users WHERE role=\"end-user\"”
- PARTITION_FIELD(문자열) - 읽기에 사용됩니다. 쿼리를 파티셔닝하는 데 사용할 필드. 기본 필드는 증분 내보내기를 지원하는 엔터티update_at의 경우 입니다API(created_at의 경우 ticket-events , 의 time 경우 ticket-metric-events).
- LOWER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- UPPER_BOUND(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외). 선택 사항입니다. 이 옵션은 작업 옵션에 제공되지 않은 경우 커넥터에서 처리됩니다. 기본값 - “2024-05-01T20:55:02.000Z
- NUM_PARTITIONS(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수. 선택 사항입니다. 이 옵션은 작업 옵션에 제공되지 않은 경우 커넥터에서 처리됩니다. 기본값: 1.
- IMPORT_DELETED_RECORDS(문자열) - 기본값: FALSE. 읽기에 사용됩니다. 쿼리하는 동안 삭제 레코드를 가져오려면 다음을 수행합니다.
- ACCESS_TOKEN - 요청에 사용할 액세스 토큰입니다.
- INSTANCE_URL - 사용자가 작업을 실행하려는 URL 인스턴스의 입니다. 예: https://{subdomain}/.zendesk.com

제한 사항

다음은 Zendesk 커넥터의 제한 사항입니다.

- 오프셋 기반 페이지 매김은 가져올 수 있는 페이지 수를 100개로 제한하지만 가져올 수 있는 총 레코드 수가 10,000개이므로 권장하지 않습니다. 그러나 Zendesk 커넥터에 구현된 커서 기반 페이지 매김은 이러한 제한을 극복합니다. Zendesk 를 통해 EQUAL_TO 필터 연산자만 지원됩니다API.

  이러한 제한으로 인해 Zendesk 커넥터에는 파티셔닝이 지원되지 않습니다.
- '티켓 이벤트' 엔터티의 경우 속도 제한은 분당 요청 10개입니다. 작업을 실행하는 AWS Glue ETL 동안 429(요청이 너무 많음) 오류가 발생할 수 있습니다.

Zoho CRM에 연결

Zoho CRM은 단일 리포지토리와 같이 작동하여 영업, 마케팅 및 고객 지원 활동을 함께 수행하고 프로세스, 정책 및 인력을 하나의 플랫폼에서 간소화합니다. Zoho CRM은 모든 규모와 유형의 비즈니스가 갖고 있는 특정 요구 사항을 충족하도록 쉽게 사용자 지정할 수 있습니다.

Zoho CRM의 개발자 플랫폼에서는 비즈니스/엔터프라이즈가 작업을 자동화하고, 엔터프라이즈 스택에서 데이터를 통합하며, 웹 및 모바일을 위한 사용자 지정 솔루션을 생성할 수 있도록 로우 코드 및 프로 코드 도구를 올바르게 조합하여 제공합니다.

주제

- Zoho CRM에 대한 AWS Glue의 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Zoho CRM 구성
- Zoho CRM 연결 구성
- Zoho CRM 엔터티에서 읽기
- Zoho CRM 연결 옵션
- Zoho CRM 커넥터에 대한 제한 사항 및 참고 사항

Zoho CRM에 대한 AWS Glue의 지원

AWS Glue에서는 다음과 같이 Zoho CRM을 지원합니다.

소스로 지원되나요?

예 - 동기화 및 비동기. AWS Glue ETL 작업을 사용하여 Zoho CRM에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Zoho CRM API 버전

다음 Zoho CRM API 버전이 지원됩니다.

- v7

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Zoho CRM 구성

AWS Glue를 사용하여 Zoho CRM에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Zoho CRM 계정이 있습니다.
- Zoho CRM 계정이 API 액세스에 대해 활성화되어 있습니다.
- OAuth 자격 증명을 가져올 등록된 API 클라이언트가 API 콘솔 아래에 있습니다.

Zoho CRM 연결 구성

권한 부여 유형은 AWS Glue에서 Zoho CRM과 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다. 선택한 항목은 연결을 생성하기 전에 충족해야 하는 요구 사항에 영향을 미칩니다. Zoho CRM에서는 OAuth 2.0에 대한 AUTHORIZATION_CODE 권한 부여 유형만 지원합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. AWS Glue 콘솔은 사용자를 Zoho CRM으로 리디렉션합니다. 사용자가 로그인하고 Zoho CRM 인스턴스에 액세스하도록 요청된 권한을 Glue에 허용해야 합니다.
- 사용자는 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Zoho CRM에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID, 권한 부여 URL, 토큰 URL 및 인스턴스 URL을 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Zoho CRM으로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰의 유효 기간은 1시간이며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Zoho CRM 설명서는 Authentication을 참조하세요.

Zoho CRM 연결을 구성하는 방법:

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Zoho CRM을 선택하세요.
   b. 연결하려는 Zoho CRM 인스턴스의 INSTANCE_URL 항목을 제공합니다.
   c. 사용자 클라이언트 애플리케이션 클라이언트 ID를 제공합니다.
   d. 드롭다운에서 적절한 권한 부여 URL을 선택하세요.
   e. 드롭다운에서 적절한 토큰 URL을 선택하세요.
   f. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

g. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

h. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.
3. AWS Glue 작업 구성에서 추가 네트워크 연결로 `connectionName`을 제공합니다.

Zoho CRM 엔터티에서 읽기

사전 조건

읽으려는 Zoho CRM 객체입니다. 객체 이름이 필요합니다.

동기화 소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Product	예	예	예	예	예
Quote	예	예	예	예	예
Purchase Order	예	예	예	예	예
Solution	예	예	예	예	예

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Call	예	예	예	예	예
Task	예	예	예	예	예
Event	예	예	예	예	예
Invoice	예	예	예	예	예
Account	예	예	예	예	예
Contact	예	예	예	예	예
Vendor	예	예	예	예	예
캠페인	예	예	예	예	예
Deal	예	예	예	예	예
Lead	예	예	예	예	예
Custom Module	예	예	예	예	예
Sales Order	예	예	예	예	예
Price Books	예	예	예	예	예
사례	예	예	예	예	예

예시:

```
zoho_read = glueContext.create_dynamic_frame.from_options(
    connection_type="ZOHO",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v7",
        "INSTANCE_URL": "https://www.zohoapis.in/"
```

```
    }
```

비동기 소스에 대해 지원되는 엔터티:

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Product	예	아니요	아니요	예	아니요
Quote	예	아니요	아니요	예	아니요
Purchase Order	예	아니요	아니요	예	아니요
Solution	예	아니요	아니요	예	아니요
Call	예	아니요	아니요	예	아니요
Task	예	아니요	아니요	예	아니요
Event	예	아니요	아니요	예	아니요
Invoice	예	아니요	아니요	예	아니요
Account	예	아니요	아니요	예	아니요
Contact	예	아니요	아니요	예	아니요
Vendor	예	아니요	아니요	예	아니요
캠페인	예	아니요	아니요	예	아니요
Deal	예	아니요	아니요	예	아니요
Lead	예	아니요	아니요	예	아니요
Custom Module	예	아니요	아니요	예	아니요
Sales Order	예	아니요	아니요	예	아니요

개체	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Price Books	예	아니요	아니요	예	아니요
사례	예	아니요	아니요	예	아니요

예시:

```
zoho_read = glueContext.create_dynamic_frame.from_options(
    connection_type="ZOHO",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v7",
        "INSTANCE_URL": "https://www.zohoapis.in/",
        "TRANSFER_MODE": "ASYNC"
    }
```

Zoho CRM 필드 세부 정보:

Zoho CRM에서는 지원되는 엔터티에 대해 메타데이터를 동적으로 가져오도록 엔드포인트를 제공합니다. 따라서 운영자 지원은 데이터 유형 수준에서 캡처됩니다.

개체	데이터 유형	지원되는 연산자
Zoho 엔터티(모든 엔터티)	Integer	!=, =, <, <=, >, >=, BETWEEN
	String	Like, =, !=
	BigInteger	!=, =, <, <=, >, >=, BETWEEN
	Boolean	=
	배정밀도 실수	!=, =, <, <=, >, >=, BETWEEN
	BigDecimal	!=, =, <, <=, >, >=, BETWEEN
	날짜	!=, =, <, <=, >, >=, BETWEEN

개체	데이터 유형	지원되는 연산자
	DateTime	!=, =, <, <=, >, >=, BETWEEN
	Struct	N/A
	나열	N/A

분할 쿼리

파티셔닝은 비동기 모드에서 지원되지 않습니다.

필터 기반 파티셔닝(동기 모드):

Spark에서 동시성을 활용하려는 경우 추가 Spark 옵션(PARTITION_FIELD, LOWER_BOUND, UPPER_BOUND, NUM_PARTITIONS)을 제공할 수 있습니다. 이러한 파라미터를 사용하면 Spark 작업에서 동시에 실행할 수 있는 NUM_PARTITIONS개의 하위 쿼리로 원래 쿼리가 분할됩니다.

- PARTITION_FIELD: 쿼리 분할에 사용할 필드의 이름입니다.
- LOWER_BOUND: 선택한 파티션 필드의 하한 값(경계 포함).

  Datetime 필드의 경우 Spark SQL 쿼리에 사용된 Spark 타임스탬프 형식을 허용합니다.

  유효한 값의 예제:

  ```
  "2024-09-30T01:01:01.000Z"
  ```
- UPPER_BOUND: 선택한 파티션 필드의 상한 값(경계 제외).
- NUM_PARTITIONS: 파티션 수.

예시:

```
zoho_read = glueContext.create_dynamic_frame.from_options(
    connection_type="zohocrm",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "entityName",
        "API_VERSION": "v7",
        "PARTITION_FIELD": "Created_Time"
        "LOWER_BOUND": "2022-01-01T01:01:01.000Z"
```

```
        "UPPER_BOUND": "2024-01-01T01:01:01.000Z"
        "NUM_PARTITIONS": "10"
    }
```

Zoho CRM 연결 옵션

다음은 Zoho CRM에 대한 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Zoho CRM에서의 객체 이름.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용할 CRM Rest API 버전.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 객체에 대해 선택할 열.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.
- `PARTITION_FIELD`(문자열) - 읽기에 사용됩니다. 쿼리 분할에 사용할 필드입니다.
- `LOWER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 하한 값(경계 포함).
- `UPPER_BOUND`(문자열) - 읽기에 사용됩니다. 선택한 파티션 필드의 상한 값(경계 제외).
- `NUM_PARTITIONS`(정수) - 기본값: 1. 읽기에 사용됩니다. 읽을 파티션 수.
- `INSTANCE_URL`(문자열) - (필수) 읽기에 사용됩니다. 유효한 Zoho CRM 인스턴스 URL.
- `TRANSFER_MODE`(문자열) - 쿼리를 비동기 모드에서 실행해야 하는지 여부를 나타내는 데 사용됩니다.

Zoho CRM 커넥터에 대한 제한 사항 및 참고 사항

다음은 Zoho CRM 커넥터에 대한 제한 사항입니다.

- API 버전 v7에서는 최대 100,000개의 레코드를 가져올 수 있습니다. Zoho 설명서를 참조하세요.
- 이벤트 엔터티의 경우 Zoho 설명서에 언급된 대로 "Meeting" 레이블이 표시됩니다.
- 모두 선택 기능의 경우:
  - GET 및 POST 직접 호출 모두에 대해 SaaS에서 최대 50개의 필드를 가져올 수 있습니다.
  - 처음 50개 필드에 속하지 않는 일부 특정 필드의 데이터를 가져오려면 선택한 필드 목록을 수동으로 제공해야 합니다.
  - 50개가 넘는 필드를 선택하면 50개 필드를 초과하는 모든 필드가 잘리고 Amazon S3에 null 데이터가 포함됩니다.

  - 필터 표현식의 경우 사용자가 제공한 50개 필드 목록에 "id" 및 "Created_Time"이 포함되지 않은 경우 사용자에게 이러한 필드를 포함하라는 사용자 지정 예외가 발생합니다.
- 필터 연산자는 데이터 유형이 동일하더라도 필드마다 다를 수 있습니다. 따라서 SaaS 플랫폼에서 오류를 트리거하는 모든 필드에 대해 다른 연산자를 수동으로 지정해야 합니다.
- 정렬 기준 기능의 경우:
  - 필터 표현식을 사용하지 않으면 단일 필드를 기준으로만 데이터를 정렬할 수 있지만 필터 표현식이 적용되는 경우 여러 필드로 기준으로 데이터를 정렬할 수 있습니다.
  - 선택한 필드에 대해 정렬 순서가 지정되지 않은 경우 데이터는 기본적으로 오름차순으로 검색됩니다.
- Zoho CRM 커넥터에 대해 지원되는 리전은 미국, 유럽, 인도, 호주 및 일본입니다.
- 비동기 읽기 기능 제한 사항:
  - 비동기 모드에서는 제한 순서 기준 및 파티셔닝이 지원되지 않습니다.
  - 비동기 모드에서는 페이지당 레코드가 200,000개인 최대 500페이지의 데이터를 전송할 수 있습니다.
  - 1분 간격의 경우 10개의 요청만 다운로드할 수 있습니다. 다운로드 한도를 초과하면 HTTP 429 오류가 반환되고 모든 다운로드 요청이 1분 동안 일시 중지된 후 처리가 재개됩니다.
  - 대량 작업을 완료한 후 1일 동안만 다운로드 가능한 파일에 액세스할 수 있습니다. 그런 다음 엔드포인트를 통해 파일에 액세스할 수 없습니다.
  - 엔드포인트를 통해 최대 200개의 선택 필드를 제공할 수 있습니다. 엔드포인트에 200개 이상의 선택 필드를 지정하면 시스템에서 해당 모듈에 사용 가능한 모든 필드를 자동으로 내보냅니다.
  - 모든 모듈에서 생성된 외부 필드는 대량 읽기 API에서 지원되지 않습니다.
  - 정렬 및 `Group_by` 절은 이 API 엔드포인트를 통해 지원되지 않습니다.
  - 규정 준수 설정에서 API를 통한 데이터 액세스 제한 옵션이 비활성화된 경우에만 민감한 상태 데이터가 있는 필드의 값이 검색됩니다. 이 옵션이 활성화되면 결과에서 값이 비어 있게 됩니다.
  - 필터링/기준 제한
    - 쿼리에 사용할 수 있는 최대 기준 수는 25개입니다.
    - 여러 줄 텍스트 필드의 필터링/기준은 지원되지 않습니다.

Zoom Meetings에 연결

Zoom Meetings는 화상 회의, 오디오 회의, 웨비나, 회의 녹화, 라이브 채팅에 사용할 수 있는 클라우드 기반 화상 회의 플랫폼입니다.

주제

- AWS Glue의 Zoom Meetings 지원
- 연결을 생성하고 사용하기 위한 API 작업이 포함된 정책
- Zoom Meetings 구성
- Zoom Meetings 클라이언트 앱 구성
- Zoom Meetings 연결 구성
- Zoom Meetings 엔터티에서 읽기
- Zoom Meetings 연결 옵션
- Zoom Meetings 제한 사항

AWS Glue의 Zoom Meetings 지원

AWS Glue에서는 다음과 같이 Zoom Meetings를 지원합니다.

소스로 지원되나요?

예. AWS Glue ETL 작업을 사용하여 Zoom Meetings에서 데이터를 쿼리할 수 있습니다.

대상으로서 지원되나요?

아니요.

지원되는 Zoom Meetings API 버전

다음 Zoom Meetings API 버전이 지원됩니다.

- v2

연결을 생성하고 사용하기 위한 API 작업이 포함된 정책

다음 샘플 정책은 연결을 생성하고 사용하는 데 필요한 AWS IAM 권한을 설명합니다. 새 역할을 생성하는 경우 다음을 포함하는 정책을 생성합니다.

```
{
 "Version": "2012-10-17",
    "Statement": [
     {
```

```
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens",
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

위 메서드를 사용하지 않으려는 경우 대신 다음 관리형 IAM 정책을 사용합니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

Zoom Meetings 구성

AWS Glue를 사용하여 Zoom Meetings에서 데이터를 전송하려면 먼저 다음 요구 사항을 충족해야 합니다.

최소 요구 사항

다음은 최소 요구 사항입니다.

- Zoom Meetings 계정이 있습니다.
- Zoom 계정이 API 액세스에 대해 활성화되어 있습니다.
- Zoom Meetings 계정에서 OAuth2 앱을 생성했습니다. 이 통합에서는 계정에 대해 인증된 직접 호출을 수행하는 경우 AWS Glue에서 데이터에 안전하게 액세스하는 데 사용하는 자격 증명을 제공합니다. 자세한 내용은 the section called “Zoom Meetings 클라이언트 앱 구성” 단원을 참조하십시오.

이러한 요구 사항을 충족하면 Zoom Meetings 계정에 AWS Glue를 연결할 준비가 된 것입니다. 일반적인 연결의 경우 Zoom Meetings에서 다른 작업을 수행하지 않아도 됩니다.

Zoom Meetings 클라이언트 앱 구성

1. Zoom App Marketplace에 로그인합니다.
2. 개발 > 앱 빌드를 선택합니다.
3. OAuth 2.0 기반 앱의 일반 앱을 선택합니다.
4. 기본 정보 페이지에서 앱 이름, 앱 관리 방식, 앱 자격 증명, OAuth 정보와 같은 앱에 대한 정보를 추가하거나 업데이트합니다.
5. 앱 관리 방식 선택 섹션에서 앱 관리 방식을 확인합니다.
   a. 관리자 관리형: 계정 관리자가 앱을 추가하고 관리합니다.
   b. 사용자 관리형: 개별 사용자가 앱을 추가하고 관리합니다. 앱은 사용자의 승인된 데이터에만 액세스할 수 있습니다.
6. 앱 자격 증명: 빌드 흐름은 앱에 대한 앱 자격 증명(클라이언트 ID 및 클라이언트 보안 암호)을 자동으로 생성합니다.
7. OAuth 정보 섹션에서 앱에 대한 OAuth를 설정합니다.
   a. OAuth 리디렉션 URL(필수): 리디렉션 URL 또는 엔드포인트를 입력하여 앱과 Zoom 간에 OAuth를 설정합니다.
   b. 엄격 모드 URL 사용(선택 사항)
   c. 하위 도메인 확인(선택 사항)
   d. OAuth 허용 목록(필수): Zoom이 OAuth 흐름에 유효한 리디렉션으로 허용해야 하는 고유한 URL을 추가합니다.
8. 범위 페이지에서 앱이 직접적으로 호출할 수 있는 Zoom API 메서드를 선택합니다. 범위는 사용자가 사용할 수 있는 정보와 기능을 정의합니다. 다음과 같은 세분화된 범위를 선택합니다.
   - user:read:list_users:admin
   - zoom_rooms:read:list_rooms:admin
   - group:read:list_members:admin
   - group:read:administrator:admin
   - group:read:list_groups:admin
   - report:read:admin
   - role:read:list_roles, role:read:list_roles:admin

범위가 추가되면 계속을 선택하고 앱을 사용할 준비가 된 것입니다.

OAuth 2.0 설정에 대한 자세한 내용은 Integrations(OAuth apps)를 참조하세요.

Zoom Meetings 연결 구성

Zoom Meetings에서는 OAuth2에 대한 AUTHORIZATION_CODE 권한 부여 유형을 지원합니다. 권한 부여 유형은 AWS Glue에서 Zoom Meetings과 통신하여 데이터에 대한 액세스를 요청하는 방법을 결정합니다.

- 이 권한 부여 유형은 사용자를 인증하기 위해 사용자를 서드파티 권한 부여 서버로 리디렉션하는 방식에 의존하므로 '3각' OAuth로 간주됩니다. AWS Glue 콘솔을 통해 연결을 생성할 때 사용됩니다. 연결을 생성하는 사용자는 Zoom Meetings 클라이언트 애플리케이션에 대한 클라이언트 ID 및 클라이언트 보안 암호와 같은 OAuth 관련 정보를 제공해야 합니다. AWS Glue 콘솔은 사용자를 Zoom으로 리디렉션합니다. 사용자가 로그인하고 Zoom Meetings 인스턴스에 액세스하도록 요청된 권한을 AWS Glue에 허용해야 합니다.
- 사용자는 여전히 AWS Glue 콘솔을 통해 연결을 생성할 때에도 Zoom Meetings에서 자체 연결된 앱을 생성하고 자체 클라이언트 ID와 클라이언트 보안 암호를 제공하기로 선택할 수 있습니다. 이 시나리오에서는 여전히 Zoom Meetings으로 리디렉션되어 로그인하고 리소스에 액세스할 수 있는 권한을 AWS Glue에 부여합니다.
- 이 권한 부여 유형은 새로 고침 토큰과 액세스 토큰을 생성합니다. 액세스 토큰은 수명이 짧으며 새로 고침 토큰을 사용하여 사용자 상호 작용 없이 자동으로 새로 고칠 수 있습니다.
- 권한 부여 코드 OAuth 흐름을 위한 연결된 앱 생성에 대한 퍼블릭 Zoom Meetings 설명서는 Using OAuth 2.0을 참조하세요.

Zoom Meetings 연결을 구성하는 방법:

1. AWS Secrets Manager에서 다음 세부 정보로 보안 암호를 생성합니다.
   a. 고객 관리형 연결된 앱의 경우 보안 암호는 키 역할을 하는 `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET`과 함께 연결된 앱 소비자 보안 암호를 포함해야 합니다.
   b. 참고: AWS Glue에서 연결의 보안 암호를 생성해야 합니다.

1. AWS Glue Glue Studio의 데이터 연결에서 아래 단계에 따라 연결을 생성하세요.
   a. 연결 유형을 선택할 때 Zoom Meetings를 선택합니다.

b. 연결하려는 Zoom Meetings 환경을 제공합니다.

c. 다음 작업에 대한 권한이 있고 AWS Glue에서 수임할 수 있는 AWS IAM 역할을 선택합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

d. 토큰을 넣기 위해 AWS Glue에서 이 연결에 사용할 `secretName`을 선택합니다.

e. 네트워크를 사용하려는 경우 네트워크 옵션을 선택합니다.

2. AWS Glue 작업 권한과 연결된 IAM 역할에 `secretName`을 읽을 수 있는 권한을 부여합니다.

Zoom Meetings 엔터티에서 읽기

사전 조건

읽으려는 Zoom Meetings 객체입니다. 객체 이름(예: `Group` 또는 `Zoom Rooms`)이 필요합니다.

소스에 대해 지원되는 엔터티:

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Zoom Rooms	아니요	예	아니요	예	아니요
Group	아니요	아니요	아니요	예	아니요

엔터티	필터링 가능	제한 지원	정렬 기준 지원	Select * 지원	분할 지원
Group Member	예	예	아니요	예	아니요
Group Admin	아니요	예	아니요	예	아니요
Report(daily)	예	아니요	아니요	예	아니요
Roles	아니요	아니요	아니요	예	아니요
Users	예	예	아니요	예	아니요

예시:

```
zoom_read = glueContext.create_dynamic_frame.from_options(
    connection_type="zoom",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "organization",
        "API_VERSION": "v2"
    }
)
```

Zoom Meetings 엔터티 및 필드 세부 정보:

Zoom Meetings에서는 선택한 엔터티 아래에서 사용 가능한 필드를 동적으로 로드합니다. 필드의 데이터 유형에 따라 다음 필터 연산자를 지원합니다.

엔터티	필드	데이터 유형	지원되는 연산자
Zoom Room	status	String	=
	type	String	=
	unassigned_rooms	Boolean	=
	location_id	String	=

엔터티	필드	데이터 유형	지원되는 연산자	
	room_id	String	N/A	
	activation_code	String	N/A	
	id	String	N/A	
	name	String	N/A	
	tag_ids	String	N/A	
	query_name	String	N/A	
Daily Report	month	Date	=	
	date	Date	N/A	
	meeting_minutes	Integer	N/A	
	meetings	Integer	N/A	
	new_users	Integer	N/A	
	participants	Integer	N/A	
	group_id	String	N/A	
User	created_at	DateTime	N/A	
	dept	String	N/A	
	email	String	N/A	
	employee_ unique_id	String	N/A	
	first_name	String	N/A	
	group_ids	List	N/A	
	host_key	String	N/A	

엔터티	필드	데이터 유형	지원되는 연산자	
	id	String	N/A	
	im_group_ids	String	N/A	
	last_client_version	String	N/A	
	last_login_time	DateTime	N/A	
	last_name	String	N/A	
	plan_united_type	String	N/A	
	custom_attributes	List	N/A	
	pmi	BigInteger	N/A	
	role_id	String	=	
	status	String	=	
	timezone	String	N/A	
	type	Integer	N/A	
	verified	Integer	N/A	
	user_created_at	DateTime	N/A	
	display_name	String	N/A	
	phone_number	String	N/A	
	language	String	N/A	
	license	String	=	
Group	id	String	N/A	

엔터티	필드	데이터 유형	지원되는 연산자	
	name	String	N/A	
	total_members	Integer	N/A	
Group Member	email	String	N/A	
	first_name	String	N/A	
	id	String	N/A	
	last_name	String	N/A	
	type	Integer	N/A	
	primary_group	Boolean	N/A	
	member_id	String	N/A	
Group Admin	id	String	N/A	
	email	String	N/A	
	name	String	N/A	
role	description	String	N/A	
	id	String	N/A	
	name	String	N/A	
	total_members	Integer	N/A	
	type	String	=	

분할 쿼리

Zoom Meetings는 필터 기반 분할 또는 레코드 기반 분할을 지원하지 않습니다.

Zoom Meetings 연결 옵션

다음은 Zoom Meetings의 연결 옵션입니다.

- `ENTITY_NAME`(문자열) - (필수) 읽기에 사용됩니다. Zoom Meetings 엔터티의 이름입니다. 예: `group`.
- `API_VERSION`(문자열) - (필수) 읽기에 사용됩니다. 사용하려는 Zoom Meetings Rest API 버전입니다. Zoom Meetings에서는 현재 버전 v2만 지원하므로 값은 `v2`입니다.
- `SELECTED_FIELDS`(List<String>) - 기본값: 비어 있습니다(SELECT *). 읽기에 사용됩니다. 선택한 엔터티에 대해 선택할 열의 쉼표로 구분된 목록입니다.
- `FILTER_PREDICATE`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. Spark SQL 형식이어야 합니다.
- `QUERY`(문자열) - 기본값: 비어 있습니다. 읽기에 사용됩니다. 전체 Spark SQL 쿼리.

Zoom Meetings 제한 사항

다음은 Zoom Meetings의 제한 사항 또는 참고 사항입니다.

- Zoom Meetings는 orderby를 지원하지 않습니다.
- 필요한 기준을 충족할 수 있는 필드가 없으므로 Zoom Meetings는 필터 기반 분할을 지원하지 않습니다.
- 페이지 매김 제한 및 오프셋 기반 페이지 매김이 지원되지 않으므로 Zoom Meetings는 레코드 기반 분할을 지원하지 않습니다.

자체 JDBC 드라이버를 사용하여 JDBC 연결 추가

JDBC 연결을 사용할 때 자체 JDBC 드라이버를 사용할 수 있습니다. AWS Glue 크롤러가 사용하는 기본 드라이버에서 데이터베이스에 연결할 수 없는 경우 자체 JDBC 드라이버를 사용할 수 있습니다. 예를 들어 Postgres 데이터베이스에서 SHA-256을 사용하려고 하지만 이전 Postgres 드라이버에서 이를 지원하지 않는 경우 자체 JDBC 드라이버를 사용할 수 있습니다.

지원되는 데이터 소스

지원되는 데이터 소스	지원되지 않는 데이터 소스
MySQL	Snowflake

지원되는 데이터 소스	지원되지 않는 데이터 소스
Postgres	
Oracle	
Redshift	
SQL Server	
Aurora*	

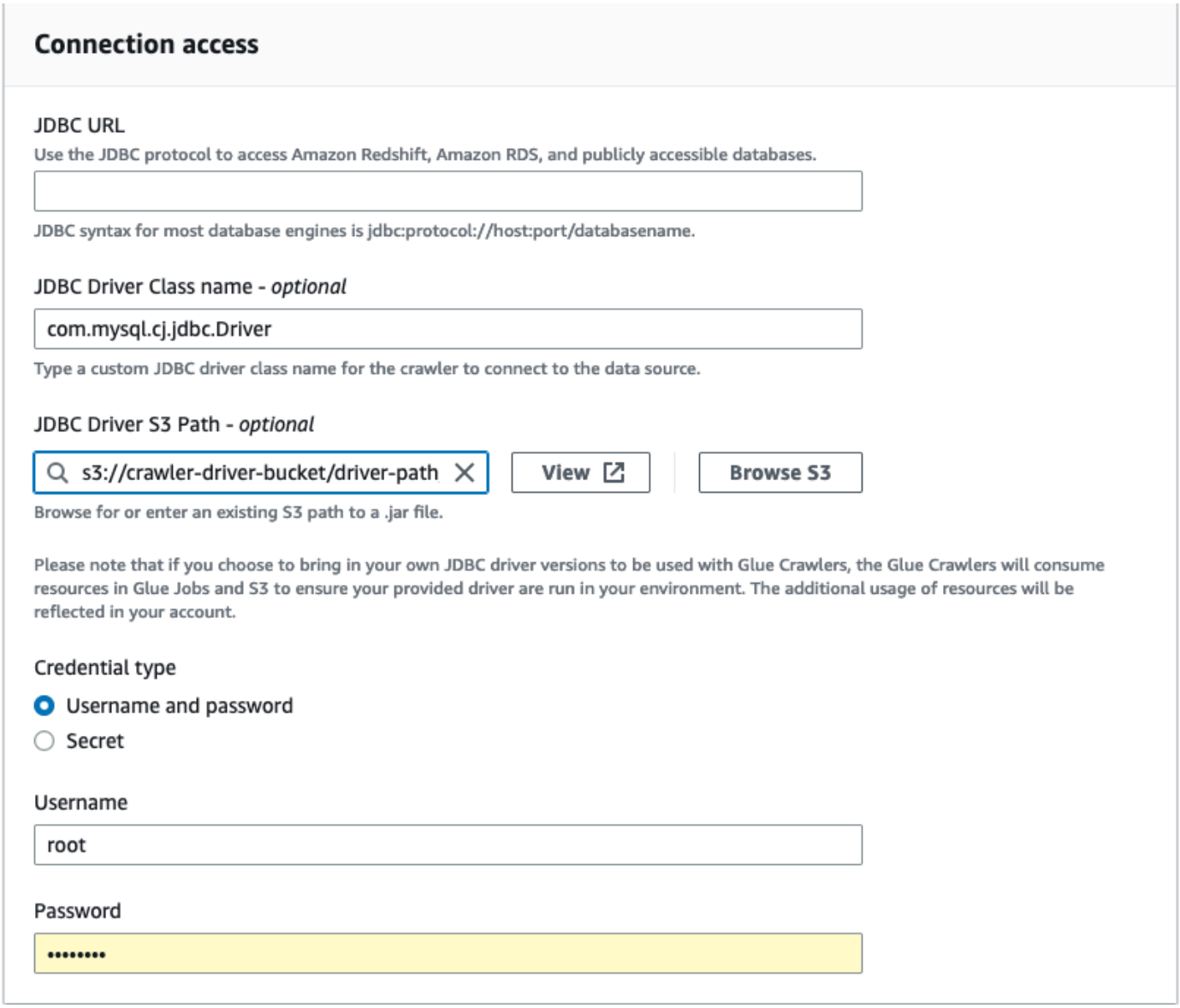
*기본 JDBC 드라이버를 사용하는 경우 지원됩니다. 일부 드라이버 기능은 사용할 수 없습니다.

JDBC 연결에 JDBC 드라이버 추가

> **Note**
>
> 자체 JDBC 드라이버 버전을 가져오기로 선택한 경우 AWS Glue 크롤러는 AWS Glue작업 및 Amazon S3 버킷의 리소스를 소비하여 제공된 드라이버가 사용자 환경에서 실행되도록 합니다. 리소스의 추가 사용량은 계정에 반영됩니다. AWS Glue 크롤러 및 작업 비용은 청구에서 AWS Glue 카테고리에 속합니다. 또한 자체 JDBC 드라이버를 제공한다고 해서 크롤러가 해당 드라이버의 모든 기능을 사용할 수 있는 것은 아닙니다.

JDBC 연결에 자체 JDBC 드라이버를 추가하려면:

1. JDBC 드라이버 파일을 Amazon S3 위치에 추가합니다. 버킷 및/또는 폴더를 생성하거나 기존 버킷 및/또는 폴더를 사용할 수 있습니다.
2. AWS Glue 콘솔의 왼쪽 메뉴에서 데이터 카탈로그 아래 연결을 선택한 다음 새 연결을 생성합니다.
3. 연결 속성 필드를 작성하고 연결 유형으로 JDBC를 선택합니다.
4. 연결 액세스에서 JDBC URL 및 JDBC 드라이버 클래스 - 선택 사항을 입력합니다. 드라이버 클래스 이름은 AWS Glue 크롤러에서 지원하는 데이터 소스의 이름이어야 합니다.

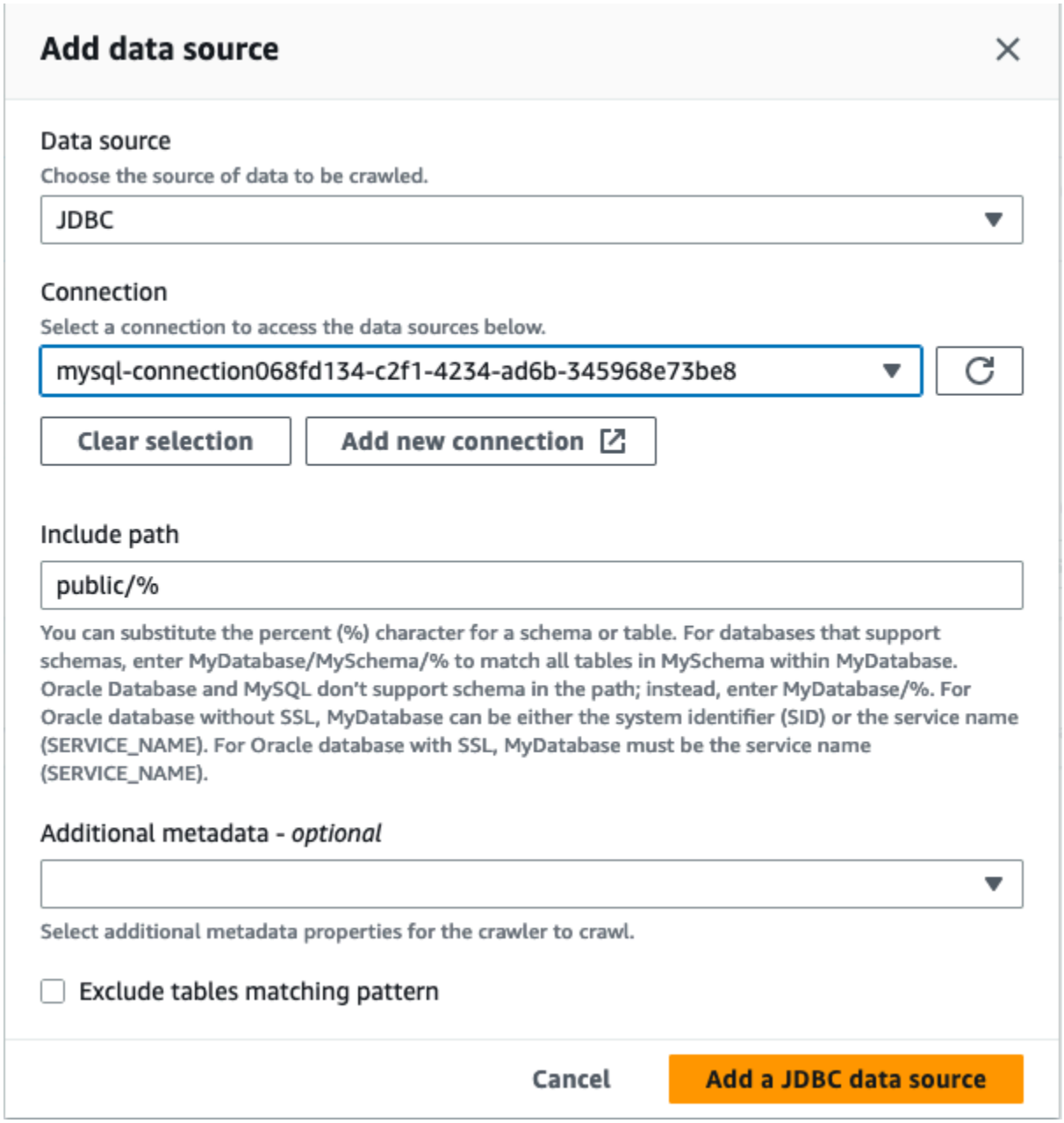


5. JDBC 드라이버 Amazon S3 경로 - 선택 사항 필드에서 JDBC 드라이버가 있는 Amazon S3 경로를 선택합니다.
6. 사용자 이름과 암호 또는 보안 암호를 입력하는 경우 보안 인증 유형 필드를 작성합니다. 완료되면 연결 생성을 선택합니다.

> **Note**
>
> 현재 테스트 연결은 지원되지 않습니다. 사용자가 제공한 JDBC 드라이버로 데이터 소스를 크롤링하는 경우 크롤러는 이 단계를 건너뛰습니다.

7. 새로 생성된 연결을 크롤러에 추가합니다. AWS Glue 콘솔 왼쪽 메뉴에서 데이터 카탈로그 아래 크롤러를 선택한 다음 새 크롤러를 생성합니다.

8. 크롤러 추가 마법사의 2단계에서 데이터 소스 추가를 선택합니다.



9. JDBC를 데이터 소스로 선택하고 이전 단계에서 생성한 연결을 선택합니다. 완료

10. AWS Glue 크롤러와 함께 자체 JDBC 드라이버를 사용하려면 크롤러가 사용하는 역할에 다음 권한을 추가합니다.

    - `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun` 작업에 대한 권한을 부여합니다.
    - IAM 작업, `iam:PassRole`에 대한 권한 부여

- Amazon S3 작업, `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`에 대한 권한을 부여합니다.
- 서비스 보안 주체에 IAM 정책의 버킷 및 폴더에 대한 액세스 권한을 부여합니다.

IAM 정책 예제:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/driver-parent-folder/driver.jar",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

AWS Glue 크롤러는 _glue_job_crawler 및 _crawler라는 두 개의 폴더를 생성합니다.

드라이버 jar이 `s3://amzn-s3-demo-bucket/driver.jar"` 폴더에 있는 경우 다음 리소스를 추가하세요.

```
"Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/_glue_job_crawler/*",
     "arn:aws:s3:::amzn-s3-demo-bucket/_crawler/*"
            ]
```

드라이버 jar이 `s3://amzn-s3-demo-bucket/tmp/driver/subfolder/driver.jar"` 폴더에 있는 경우 다음 리소스를 추가하세요.

```
"Resource": [
               "arn:aws:s3:::amzn-s3-demo-bucket/tmp/_glue_job_crawler/*",
    "arn:aws:s3:::amzn-s3-demo-bucket/tmp/_crawler/*"
            ]
```

11. VPC를 사용하는 경우 인터페이스 엔드포인트를 생성하여 AWS Glue 엔드포인트에 대한 액세스를 허용하고 이를 라우팅 테이블에 추가해야 합니다. 자세한 내용은 AWS Glue에 대한 인터페이스 VPC 엔드포인트 생성을 참조하세요.
12. 데이터 카탈로그에서 암호화를 사용하는 경우 AWS KMS 인터페이스 엔드포인트를 생성하고 이를 라우팅 테이블에 추가합니다. 자세한 내용은 AWS KMS용 VPC 엔드포인트 생성을 참조하세요.

AWS Glue Studio에서 사용자 지정 커넥터 및 연결 사용

AWS Glue는 JDBC 연결을 사용하여 가장 일반적으로 사용되는 데이터 스토어(예: Amazon Redshift, Amazon Aurora, Microsoft SQL Server, MySQL, MongoDB 및 PostgreSQL)에 대한 기본 제공 지원을 제공합니다. AWS Glue를 사용하면 추출, 변환, 로드 작업에서 사용자 정의 JDBC 드라이버를 사용할 수도 있습니다. SaaS 애플리케이션과 같이 기본적으로 지원되지 않는 데이터 스토어의 경우 커넥터를 사용할 수 있습니다.

커넥터는 AWS Glue Studio의 데이터 스토어에 액세스하는 데 도움이 되는 선택적 코드 패키지입니다. AWS Marketplace에서 제공하는 여러 커넥터를 구독할 수 있습니다.

ETL 작업을 생성할 때 기본적으로 지원되는 데이터 스토어, AWS Marketplace의 커넥터 또는 고유한 사용자 정의 커넥터를 사용할 수 있습니다. 커넥터를 사용하는 경우 먼저 커넥터에 대한 연결을 생성해야 합니다. 특정 데이터 스토어에 연결하는 데 필요한 속성을 포함하는 연결입니다. ETL 작업에서 데이터 원본 및 데이터 대상과의 연결을 사용합니다. 커넥터와 연결은 함께 작동하여 데이터 스토어에 쉽게 액세스할 수 있습니다.

커넥터 연결을 생성할 때 사용할 수 있는 연결은 다음과 같습니다.

- Amazon Aurora - 내장된 보안, 백업 및 복원, 인메모리 가속화 기능을 갖춘 확장 가능한 고성능 관계형 데이터베이스 엔진.
- Amazon DocumentDB - MongoDB 및 SQL API를 지원하는 확장 가능하고 가용성이 뛰어난 완전 관리형 도큐먼트 데이터베이스 서비스입니다.

- Amazon Redshift - MongoDB 및 SQL API를 지원하는 확장 가능하고 가용성이 뛰어난 완전 관리형 도큐먼트 데이터베이스입니다.
- Azure SQL - 확장 가능하고 안정적이며 안전한 데이터 스토리지 및 관리 기능을 제공하는 Microsoft Azure의 클라우드 기반 관계형 데이터베이스 서비스입니다.
- Cosmos DB - 확장 가능한 고성능 데이터 스토리지 및 쿼리 기능을 제공하는 Microsoft Azure의 전 세계에 분산된 클라우드 데이터베이스 서비스입니다.
- Google BigQuery - 대규모 데이터 세트에서 빠른 SQL 쿼리를 실행하기 위한 서버리스 클라우드 데이터 웨어하우스입니다.
- JDBC - 데이터 연결 및 상호 작용에 Java API를 사용하는 관계형 데이터베이스 관리 시스템(RDBMS)입니다.
- Kafka - 실시간 데이터 스트리밍 및 메시징에 사용되는 오픈 소스 스트림 처리 플랫폼입니다.
- MariaDB - 커뮤니티에서 개발한 MySQL 포크로 향상된 성능, 확장성 및 기능을 제공합니다.
- MongoDB - 높은 확장성, 유연성 및 성능을 제공하는 크로스 플랫폼 문서 지향 데이터베이스입니다.
- MongoDB Atlas - MongoDB 배포의 관리 및 확장을 간소화하는 MongoDB에서 제공하는 클라우드 기반 서비스형 데이터베이스(DBaaS)입니다.
- Microsoft SQL Server - 강력한 데이터 스토리지, 분석 및 보고 기능을 제공하는 Microsoft의 관계형 데이터베이스 관리 시스템(RDBMS)입니다.
- Mixpanel-기업이 사용자가 웹 사이트, 모바일 애플리케이션 및 기타 디지털 제품과 상호 작용하는 방식을 분석하는 데 도움이 되는 분석 플랫폼입니다.
- MySQL - 웹 애플리케이션에서 널리 사용되고 신뢰성과 확장성으로 잘 알려진 오픈 소스 관계형 데이터베이스 관리 시스템(RDBMS)입니다.
- 네트워크 - 네트워크 데이터 소스는 데이터 통합 플랫폼에서 액세스할 수 있는 네트워크 액세스 가능 리소스 또는 서비스를 나타냅니다.
- OpenSearch - OpenSearch 데이터 소스는 OpenSearch가 연결하고 데이터를 수집할 수 있는 애플리케이션입니다.
- Oracle - 오라클의 관계형 데이터베이스 관리 시스템(RDBMS)으로 강력한 데이터 스토리지, 분석 및 보고 기능을 제공합니다.
- PostgreSQL — 강력한 데이터 스토리지, 분석 및 보고 기능을 제공하는 오픈 소스 관계형 데이터베이스 관리 시스템(RDBMS)입니다.
- Salesforce – Salesforce는 영업, 고객 서비스, 전자 상거래 등에 도움이 되는 고객 관계 관리(CRM) 소프트웨어를 제공합니다. Salesforce 사용자인 경우 Salesforce 계정에 AWS Glue를 연결할 수 있습니다. 그런 다음, ETL 작업에서 Salesforce를 데이터 소스 또는 대상으로 사용할 수 있습니다. 이러

한 작업을 실행하여 Salesforce와 AWS 서비스 또는 기타 지원되는 애플리케이션 간에 데이터를 전송합니다.

- SAP HANA - 빠른 데이터 처리, 고급 분석, 실시간 데이터 통합을 제공하는 인메모리 데이터베이스 및 분석 플랫폼입니다.
- Snowflake - 확장 가능한 고성능 데이터 스토리지 및 분석 서비스를 제공하는 클라우드 기반 데이터 웨어하우스입니다.
- Teradata - 고성능 데이터 저장, 분석 및 보고 기능을 제공하는 관계형 데이터베이스 관리 시스템(RDBMS)입니다.
- Vertica - 빅데이터 분석을 위해 설계된 컬럼 중심의 분석 데이터 웨어하우스로, 빠른 쿼리 성능, 고급 분석 및 확장성을 제공합니다.

사용자 정의 커넥터 생성

고유한 커넥터를 구축하고 커넥터 코드를 AWS Glue Studio에 업로드할 수도 있습니다.

사용자 지정 커넥터는 AWS Glue Spark 런타임 API를 통해 AWS Glue Studio에 통합됩니다. AWS Glue Spark 런타임을 사용하면 Spark, Athena 또는 JDBC 인터페이스와 호환되는 모든 커넥터를 연결할 수 있습니다. 사용자 정의 커넥터에서 사용할 수 있는 연결 옵션을 전달할 수 있습니다.

AWS Glue 연결로 모든 연결 속성을 캡슐화하고 ETL 작업에 연결 이름을 제공할 수 있습니다. Data Catalog 연결과의 통합을 통해 단일 Spark 애플리케이션 또는 다른 애플리케이션의 여러 호출에서 동일한 연결 속성을 사용할 수 있습니다.

연결에 대한 추가 옵션을 지정할 수 있습니다. AWS Glue Studio가 생성하는 작업 스크립트에는 연결을 사용하여 지정된 연결 옵션으로 커넥터를 플러그 인하는 `Datasource` 항목이 포함되어 있습니다. 예:

```
Datasource = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"dbTable":"Account","connectionName":"my-custom-
jdbc-
connection"}, transformation_ctx = "DataSource0")
```

AWS Glue Studio에 사용자 지정 커넥터를 추가하려면

1. 사용자 정의 커넥터에 대한 코드를 생성합니다. 자세한 내용은 사용자 정의 커넥터 개발 단원을 참조하십시오.
2. 커넥터에 AWS Glue 기능 지원을 추가합니다. 다음은 이러한 기능의 몇 가지 예와 AWS Glue Studio에서 생성한 작업 스크립트 내에서 기능이 사용되는 방식입니다.

- 데이터 유형 매핑 - 커넥터가 기본 데이터 스토어에서 열을 읽는 동안 열을 유형 변환할 수 있습니다. 예를 들어, `{"INTEGER":"STRING"}`의 `dataTypeMapping`은 레코드를 구문 분석하고 `DynamicFrame`을 구성할 때 `Integer` 유형의 모든 열을 `String` 유형의 열로 변환합니다. 이는 사용자가 선택한 유형으로 열을 캐스팅하는 데 도움이 됩니다.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type
= "custom.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"}",
connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
```

- 병렬 읽기를 위한 분할 – AWS Glue는 열에서 데이터를 분할하여 데이터 스토어에서 병렬 데이터 읽기를 허용합니다. 파티션 열, 하위 파티션 경계, 상위 파티션 경계 및 파티션 수를 지정해야 합니다. 이 기능을 사용하면 데이터 병렬 처리와 Spark 애플리케이션에 할당된 여러 Spark 실행기를 사용할 수 있습니다.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type
= "custom.jdbc", connection_options = {"upperBound":"200","numPartitions":"4",
"partitionColumn":"id","lowerBound":"0","connectionName":"test-connection-jdbc"},
transformation_ctx = "DataSource0")
```

- 자격 증명 저장에 AWS Secrets Manager 사용 – Data Catalog 연결에는 AWS Secrets Manager에 저장된 보안 암호에 대한 `secretId`가 포함될 수도 있습니다. AWS 보안 암호는 인증 및 자격 증명 정보를 안전하게 저장하고 런타임 시 AWS Glue에 제공할 수 있습니다. 또는 다음과 같이 Spark 스크립트에서 `secretId`를 지정할 수 있습니다.

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type
= "custom.jdbc", connection_options = {"connectionName":"test-connection-jdbc",
 "secretId"-> "my-secret-id"}, transformation_ctx = "DataSource0")
```

- 행 조건자 및 열 프로젝션으로 소스 데이터 필터링 – AWS Glue Spark 런타임을 사용하면 SQL 쿼리를 푸시다운하여 행 조건자 및 열 프로젝션으로 소스에서 데이터를 필터링할 수도 있습니다. 이를 통해 ETL 작업은 푸시다운을 지원하는 데이터 스토어에서 필터링된 데이터를 더 빠르게 로드할 수 있습니다. JDBC 데이터 원본으로 푸시다운된 SQL 쿼리의 예는 `SELECT id, name, department FROM department WHERE id < 200.`입니다.

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"query":"SELECT id, name, department FROM
 department
WHERE id < 200","connectionName":"test-connection-jdbc"}, transformation_ctx =
"DataSource0")
```

- 작업 북마크 – AWS Glue는 JDBC 소스에서 데이터의 증분 로드를 지원합니다. AWS Glue는 데이터 스토어에서 마지막으로 처리된 레코드를 추적하고 후속 ETL 작업 실행에서 새 데이터 레코드를 처리합니다. 작업 북마크는 이 열이 순차적으로 증가하거나 감소하는 경우 기본 키를 북마크 키의 기본 열로 사용합니다. 작업 북마크에 대한 자세한 내용은 AWS Glue Developer Guide의 Job Bookmarks를 참조하세요.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"jobBookmarkKeys":["empno"],
 "jobBookmarkKeysSortOrder"
:"asc", "connectionName":"test-connection-jdbc"}, transformation_ctx =
 "DataSource0")
```

3. 사용자 정의 커넥터를 JAR 파일로 패키징하고 파일을 Amazon S3에 업로드합니다.
4. 사용자 정의 커넥터를 테스트합니다. 자세한 내용은 Glue Custom Connectors: Local Validation Tests Guide에서 GitHub의 지침을 참조하세요.
5. AWS Glue Studio 콘솔의 콘솔 탐색 창에서 커넥터(Connectors)를 선택합니다.
6. [커넥터(Connectors)] 페이지에서 [사용자 정의 커넥터 생성(Create custom connector)]을 선택합니다.
7. [사용자 정의 커넥터 생성(Create custom connector)] 페이지에서 다음 정보를 입력합니다.
   - Amazon S3에서 사용자 정의 코드 JAR 파일의 위치에 대한 경로입니다.
   - AWS Glue Studio에서 사용할 커넥터의 이름입니다.
   - [JDBC], [Spark] [또는 Athena] 중 하나일 수 있는 커넥터 유형입니다.
   - AWS Glue Studio가 커넥터를 사용하기 위해 호출하는 사용자 지정 코드 내의 진입점 이름입니다.
     - JDBC 커넥터의 경우 이 필드는 JDBC 드라이버의 클래스 이름이어야 합니다.
     - Spark 커넥터의 경우 이 필드는 `format` 연산자로 Spark 데이터 원본을 로드할 때 사용하는 정규화된 데이터 원본 클래스 이름 또는 해당 별칭이어야 합니다.
   - (JDBC만 해당) 데이터 스토어에 대한 JDBC 연결에서 사용하는 기본 URL입니다.
   - (선택 사항) 사용자 정의 커넥터에 대한 설명입니다.
8. [커넥터 생성(Create connector)]을 선택합니다.
9. 커넥터에 대한 연결 생성에 설명된 대로 [커넥터(Connectors)] 페이지에서 이 커넥터를 사용하는 연결을 생성합니다.

AWS Glue Studio에 커넥터 추가

커넥터는 데이터 스토어와 AWS Glue 간의 통신을 용이하게 하는 코드입니다. AWS Marketplace에서 제공되는 커넥터를 구독하거나 사용자 정의 커넥터를 생성할 수 있습니다.

AWS Marketplace 커넥터 구독

AWS Glue Studio를 사용하면 AWS Marketplace에서 커넥터를 쉽게 추가할 수 있습니다.

AWS Marketplace에서 AWS Glue Studio로 커넥터를 추가하려면

1. AWS Glue Studio 콘솔의 콘솔 탐색 창에서 커넥터(Connectors)를 선택합니다.
2. Connectors(커넥터) 페이지에서 Go to AWS Marketplace(이동)을 선택합니다.
3. AWS Marketplace의 [추천 제품(Featured products)]에서 사용하려는 커넥터를 선택합니다. 추천 커넥터 중 하나를 선택하거나 검색을 사용할 수 있습니다. 커넥터의 이름이나 유형을 검색하고 옵션을 사용하여 검색 결과를 구체화할 수 있습니다.

   주요 커넥터 중 하나를 사용하려면 [제품 보기(View product)]를 선택합니다. 검색을 사용하여 커넥터를 찾은 경우 커넥터 이름을 선택합니다.
4. 커넥터의 제품 페이지에서 탭을 사용하여 커넥터에 대한 정보를 봅니다. 이 커넥터를 구매하기로 결정했다면 [계속 구독(Continue to Subscribe)]을 선택합니다.
5. 결제 정보를 제공한 다음 [계속 구성(Continue to Configure)]을 선택합니다.
6. [이 소프트웨어 구성(Configure this software)] 페이지에서 배포 방법과 사용할 커넥터 버전을 선택합니다. 그런 뒤 [계속 시작(Continue to Launch)]을 선택합니다.
7. [이 소프트웨어 시작(Launch this software)] 페이지에서 커넥터 공급자가 제공한 [사용 지침(Usage Instructions)]을 검토할 수 있습니다. 계속할 준비가 되면 Activate connection in AWS Glue Studio(연결 활성화)를 선택합니다.

   잠시 후 콘솔에 AWS Glue Studio의 마켓플레이스 연결 생성(Create marketplace connection) 페이지가 표시됩니다.
8. 커넥터에 대한 연결 생성에 설명된 대로 이 커넥터를 사용하는 연결을 생성합니다.

   또는 [커넥터만 활성화(Activate connector only)]를 선택하여 지금 연결 생성을 건너뛸 수 있습니다. 커넥터를 사용하려면 나중에 연결을 생성해야 합니다.

커넥터에 대한 연결 생성

AWS Glue 연결은 특정 데이터 스토어에 대한 연결 정보를 저장하는 Data Catalog 객체입니다. 연결에는 로그인 자격 증명, URI 문자열, 가상 프라이빗 클라우드(VPC) 정보 등이 저장됩니다. Data Catalog에서 연결을 생성하면 작업을 생성할 때마다 모든 연결 세부 정보를 지정해야 하는 수고를 덜 수 있습니다.

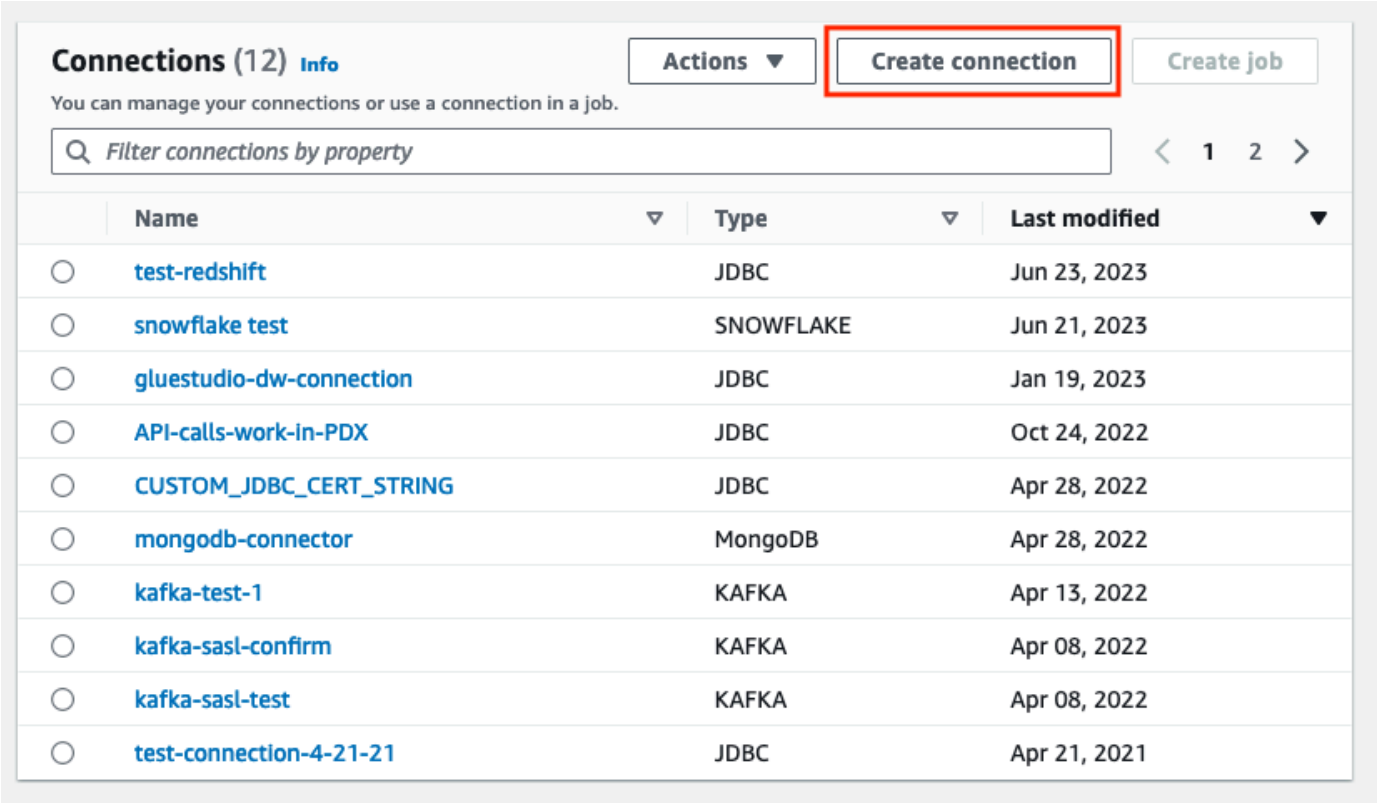
커넥터에 대한 연결을 생성하려면

1. AWS Glue Studio 콘솔의 콘솔 탐색 창에서 커넥터를 선택합니다. 아웃바운드 연결 섹션에서 연결 생성을 선택합니다.
2. 데이터 연결 만들기 마법사의 1단계에서 연결을 만들려는 데이터 소스를 선택합니다. 사용 가능한 데이터 소스를 보는 방법에는 다음과 같이 여러 가지가 있습니다.

   - 탭을 선택하여 사용 가능한 데이터 소스를 필터링합니다. 기본적으로 모든 커넥터가 선택됩니다.
   - 목록을 토글하여 데이터 소스를 목록으로 보거나 그리드로 다시 전환하여 그리드 레이아웃에서 사용 가능한 커넥터를 볼 수 있습니다.
   - 검색 창을 사용하여 데이터 소스의 목록의 범위를 좁힙니다. 입력할 때 일치하는 검색이 표시되고 일치하지 않는 소스는 보기에서 제거됩니다.

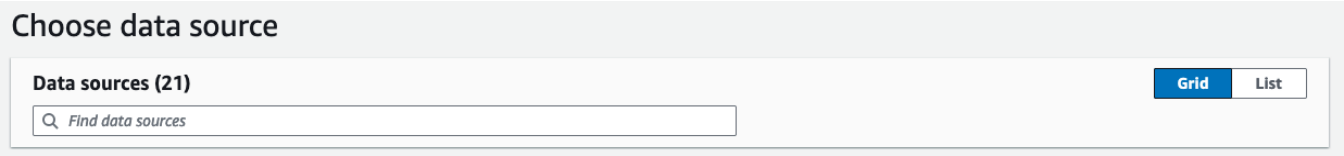
   데이터 소스를 선택한 후 다음을 선택합니다.
3. 마법사의 2단계에서 연결을 구성합니다.

   연결 세부 정보를 입력합니다. 선택한 커넥터 유형에 따라 추가 정보를 입력하라는 메시지가 나타납니다.



4. 데이터 연결 만들기 마법사의 1단계에서 연결을 만들려는 데이터 소스를 선택합니다. 사용 가능한 데이터 소스는 여러 가지 방법으로 확인할 수 있습니다. 기본적으로 사용 가능한 모든 데이터 소스가 그리드 레이아웃으로 표시됩니다. 다른 방법:

   - 목록을 토글하여 데이터 소스를 목록으로 보거나 그리드로 다시 전환하여 그리드 레이아웃에서 사용 가능한 커넥터를 볼 수 있습니다.
   - 검색 창을 사용하여 데이터 소스의 목록의 범위를 좁힙니다. 입력할 때 일치하는 검색이 표시되고 일치하지 않는 소스는 보기에서 제거됩니다.

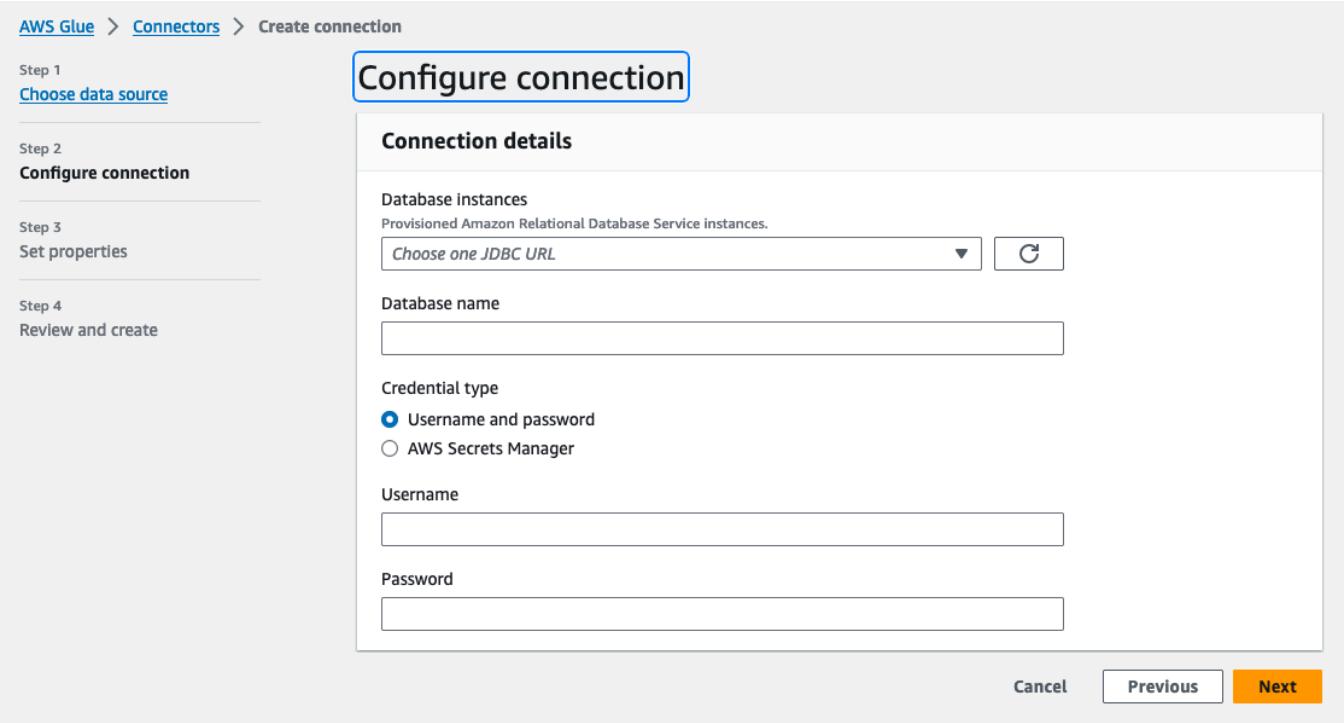


   데이터 소스를 선택한 후 다음을 선택합니다.

5. 마법사의 2단계에서 연결을 구성합니다.

   연결 세부 정보를 입력합니다. 선택한 커넥터 유형에 따라 추가 연결 정보를 입력해야 할 수도 있습니다. 여기에는 다음이 포함됩니다.

- 연결 세부 정보 - 이 필드는 연결 중인 데이터 소스에 따라 달라집니다. 예를 들어 Amazon DocumentDB 데이터베이스에 연결하는 경우 Amazon DocumentDB 를 입력합니다URL. 에 연결하는 경우 데이터베이스 인스턴스를 Amazon Aurora선택하고 데이터베이스 이름을 입력합니다. 다음은 에 필요한 연결 세부 정보입니다 Amazon Aurora.



- 보안 인증 정보 유형 - 사용자 이름과 암호 또는 AWS Secrets Manager을 선택합니다. 요청된 인증 정보를 입력합니다.
- 를 사용하는 커넥터의 경우 데이터 스토어에 JDBC URL 대한 를 생성하는 데 필요한 정보를 JDBC입력합니다.
- 가상 프라이빗 클라우드(VPC)를 사용하는 경우 에 대한 네트워크 정보를 입력합니다VPC.

6. 마법사의 3단계에서 연결 속성을 설정합니다. 이 단계의 선택적 부분으로 설명과 태그를 추가할 수 있습니다. 이름은 필수이며 기본값으로 미리 채워집니다. Next(다음)를 선택합니다.
7. 연결 소스, 세부 정보 및 속성을 검토하십시오. 변경을 하려면 마법사의 편집 단계를 선택합니다. 준비되면 연결 생성을 선택합니다.

   연결 생성을 선택합니다.

   [커넥터(Connectors)] 페이지로 돌아가고 정보 배너가 생성된 연결을 나타냅니다. 이제 AWS Glue Studio 작업에서 연결을 사용할 수 있습니다.

Kafka 연결 생성

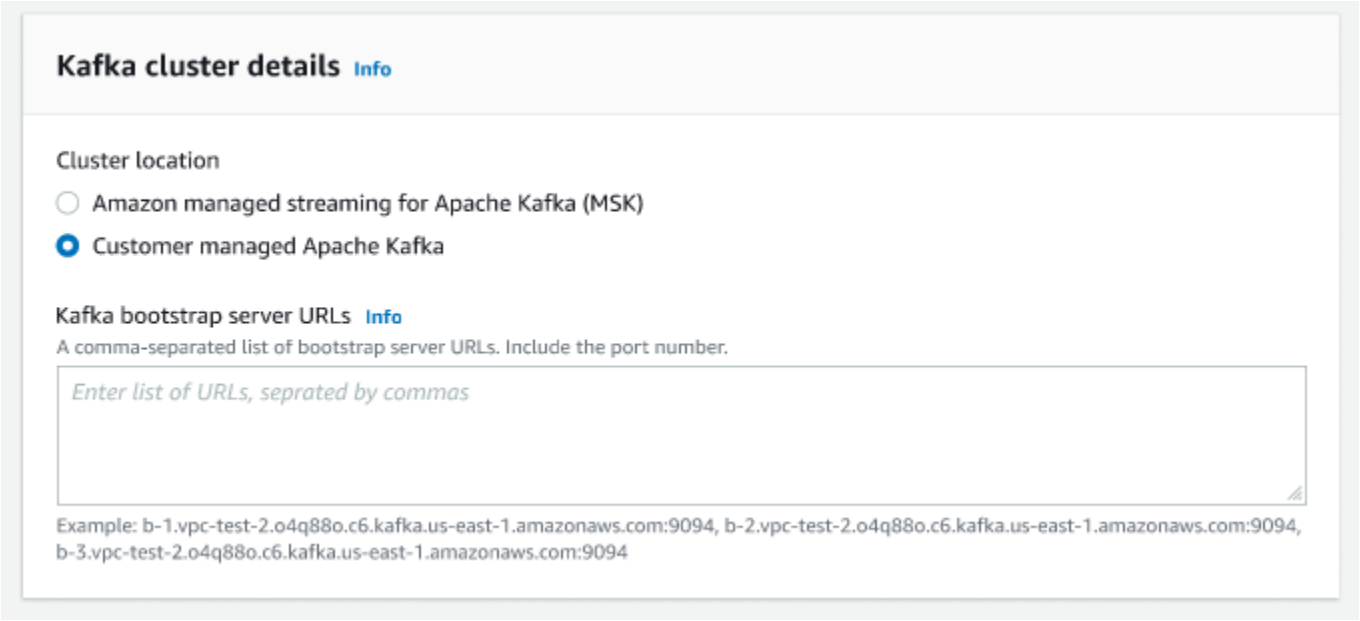
Kafka 연결을 생성할 때 드롭다운 메뉴에서 Kafka를 선택하면 구성할 추가 설정이 표시됩니다.

- Kafka 클러스터 세부 정보
- 인증
- 암호화(Encryption)
- 네트워크 옵션

Kafka 클러스터 세부 정보 구성(Configure Kafka cluster details)

1. 클러스터 위치를 선택합니다. Amazon 관리형 Apache Kafka(MSK) 클러스터용 스트리밍 또는 고객 관리형 Apache Kafka 클러스터 중에서 선택할 수 있습니다. Amazon Managed 스트리밍 for Apache Kafka에 대한 자세한 내용은 Amazon Managed 스트리밍 for Apache Kafka(MSK)를 참조하세요.

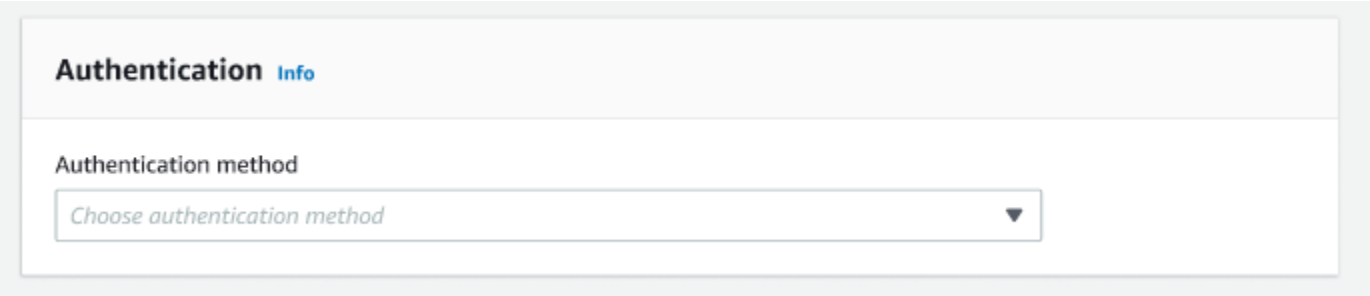
> **Note**
>
> Amazon Managed Streaming for Apache Kafka 는 TLS 및 SASL/SCRAM-SHA-512 인증 방법만 지원합니다.



2. Kafka 부트스트랩 서버의 URLs 를 입력합니다. 각 서버를 쉼표로 구분하여 둘 이상 입력할 수 있습니다. 를 URL 추가하여 의 끝에 포트 번호를 포함합니다:<port number>.

   예: b-1.vpc-test-2.034a88o.kafka-us-east-1.amazonaws.com:9094

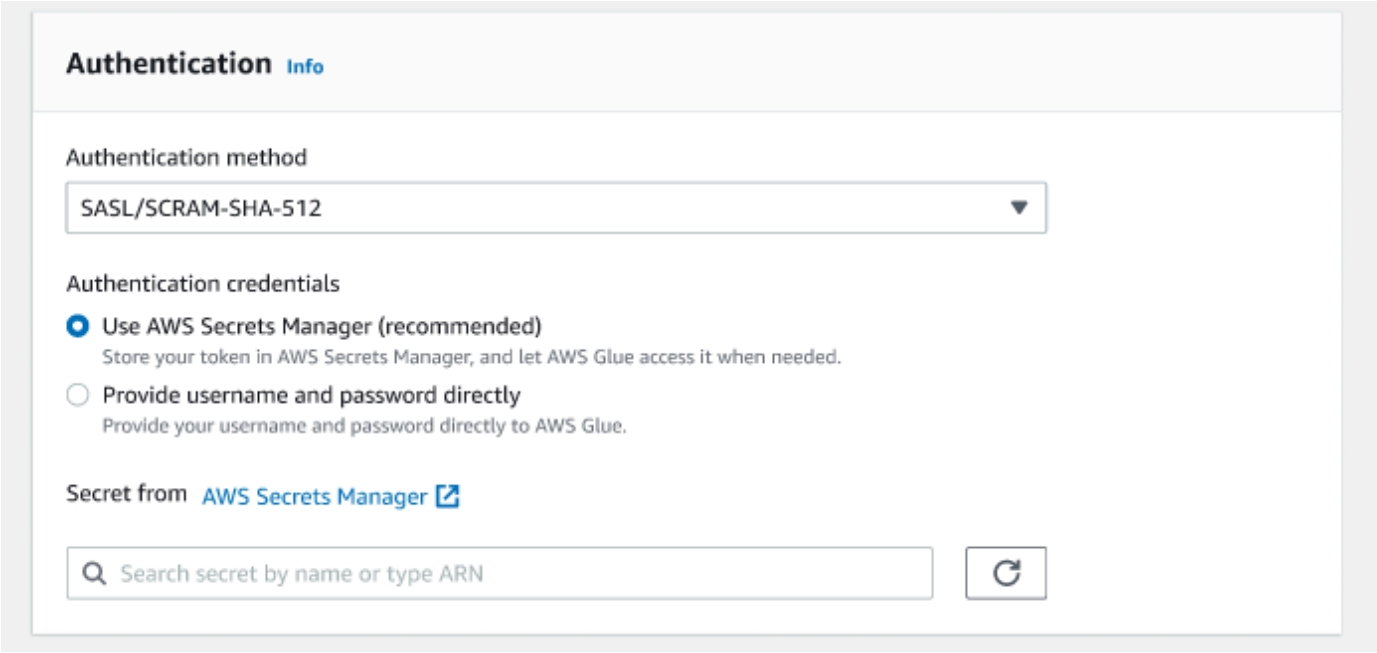
인증 방법 선택(Select authentication method)



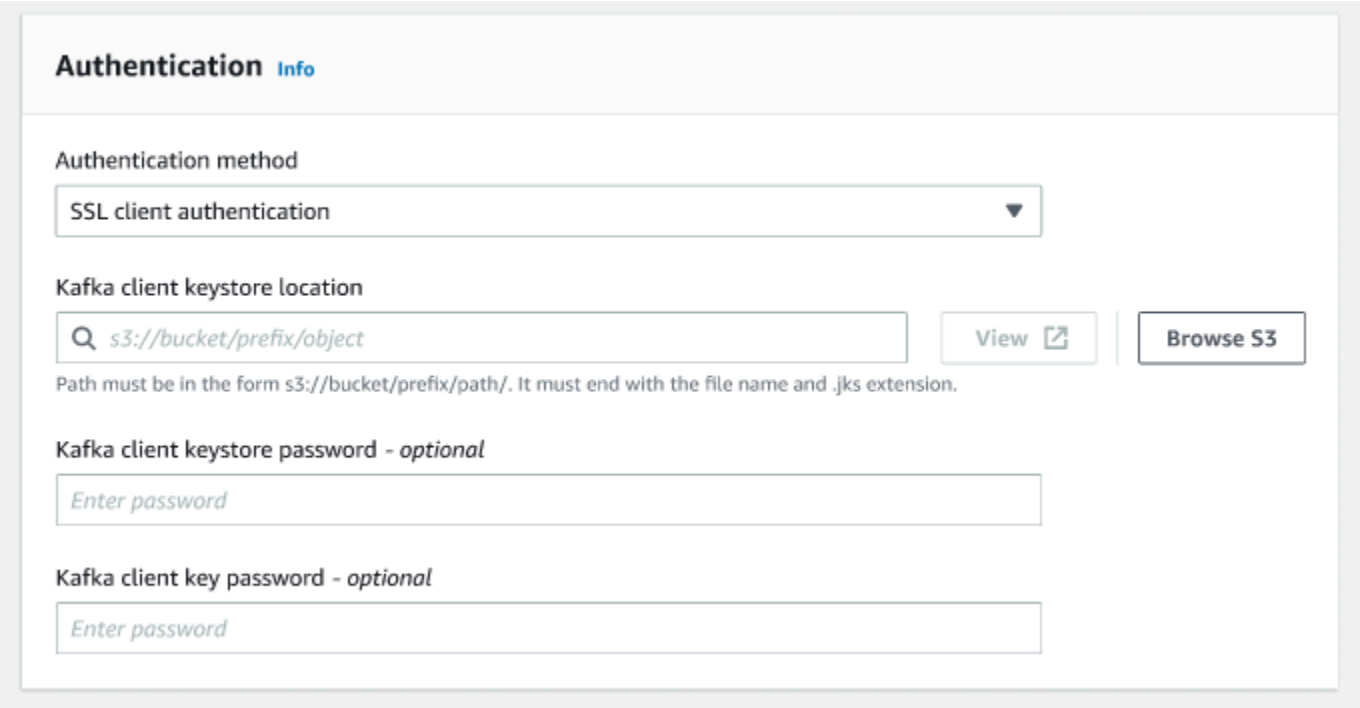
AWS Glue 는 인증을 위한 Simple Authentication and Security Layer(SASL) 프레임워크를 지원합니다. SASL 프레임워크는 다양한 인증 메커니즘을 지원하며 SCRAM (사용자 이름 및 암호), GSSAPI (Kerberos 프로토콜) 및 PLAIN (사용자 이름 및 암호) 프로토콜을 AWS Glue 제공합니다.

드롭다운 메뉴에서 인증 방법을 선택할 때 다음 클라이언트 인증 방법을 선택할 수 있습니다.

- 없음(None) - 인증이 없습니다. 이는 테스트 목적으로 연결을 생성하는 경우에 유용합니다.
- SASL/SCRAM-SHA-512 - 이 인증 방법을 선택하여 인증 자격 증명을 지정합니다. 두 가지 옵션을 사용할 수 있습니다.
  - AWS Secrets Manager 사용(권장) - 이 옵션을 선택하면 보안 인증 정보를 AWS Secrets Manager 에 저장하고 필요한 경우 정보에 AWS Glue 액세스하도록 허용할 수 있습니다. SSL 또는 SASL 인증 자격 증명을 저장하는 보안 암호를 지정합니다.



  - 사용자 이름 및 암호를 직접 제공합니다.
- SASL/GSSAPI (Kerberos) - if you select this option, you can select the location of the keytab file, krb5.conf file and enter the Kerberos principal name and Kerberos service name. The locations for the keytab file and krb5.conf file must be in an Amazon S3 location. Since MSK does not yet support SASL/GSSAPI, 이 옵션은 고객 관리형 Apache Kafka 클러스터에만 사용할 수 있습니다. 자세한 내용은 MIT Kerberos 설명서: 키탭 을 참조하세요.
- SASL/PLAIN - 이 인증 방법을 선택하여 인증 자격 증명을 지정합니다. 두 가지 옵션을 사용할 수 있습니다.
  - AWS Secrets Manager 사용(권장) - 이 옵션을 선택하면 보안 인증 정보를 AWS Secrets Manager에 저장하고 필요한 경우 정보에 AWS Glue 액세스하도록 허용할 수 있습니다. SSL 또는 SASL 인증 자격 증명을 저장하는 보안 암호를 지정합니다.
  - 사용자 이름 및 암호를 직접 제공합니다.
- SSL 클라이언트 인증 - 이 옵션을 선택하면 Amazon S3를 탐색하여 Kafka 클라이언트 키 스토어의 위치를 선택할 수 있습니다. 선택 사항으로 Kafka 클라이언트 키 스토어 암호와 Kafka 클라이언트 키 암호를 입력할 수 있습니다.



암호화 설정 구성(Configure encryption settings)

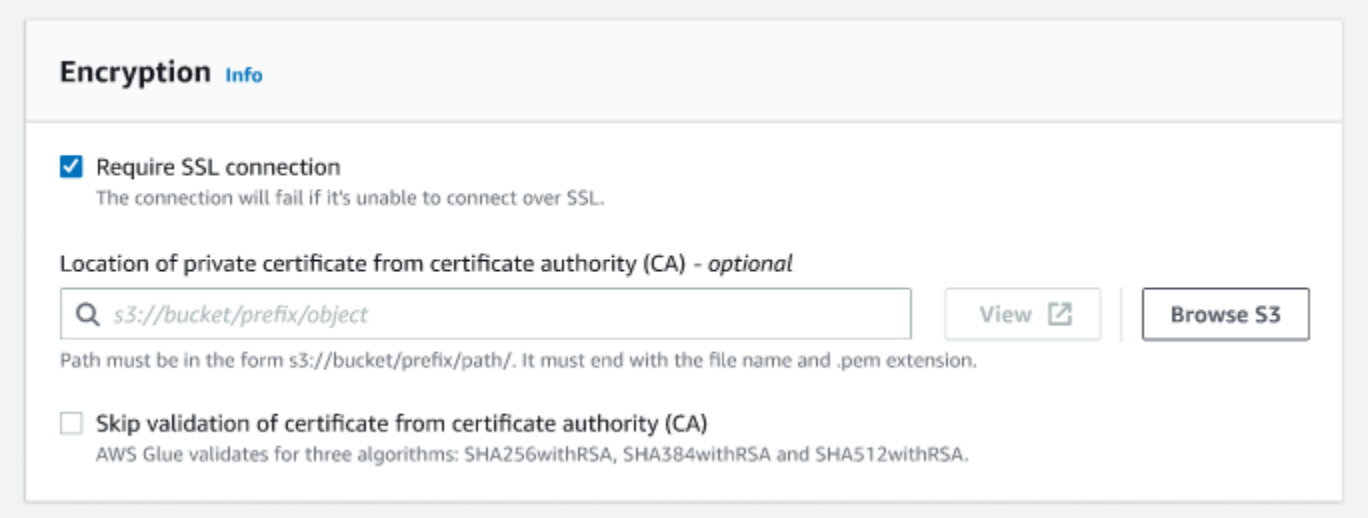
1. Kafka 연결에 SSL 연결이 필요한 경우 SSL 연결 필요 확인란을 선택합니다. 를 통해 연결할 수 없는 경우 연결이 실패합니다SSL. SSL 암호화용 는 모든 인증 방법(SASL/SCRAM-SHA-512, SASL/GSSAPI, SASL/PLAIN또는 SSL 클라이언트 인증)과 함께 사용할 수 있으며 선택 사항입니다.

   인증 방법이 SSL 클라이언트 인증 으로 설정된 경우 이 옵션은 자동으로 선택되고 변경되지 않도록 비활성화됩니다.

2. (선택 사항). 인증 기관(CA)의 사설 인증서 위치를 선택합니다. 인증 위치는 S3 위치에 있어야 합니다. 연결된 S3 버킷에서 파일을 선택하려면 찾아보기(Browse)를 선택합니다. 경로는 `s3://bucket/prefix/filename.pem` 형식이어야 합니다. 파일 이름과 .pem 확장자로 끝나야 합니다.

3. 인증 기관(CA)의 인증서 검증을 건너뛰도록 선택할 수 있습니다. 인증 기관(CA)의 인증서 검증 건너뛰기(Skip validation of certificate from certificate authority (CA)) 확인란을 선택합니다. 이 확인란을 선택하지 않으면 AWS Glue 는 다음 세 가지 알고리즘에 대해 인증서를 검증합니다.

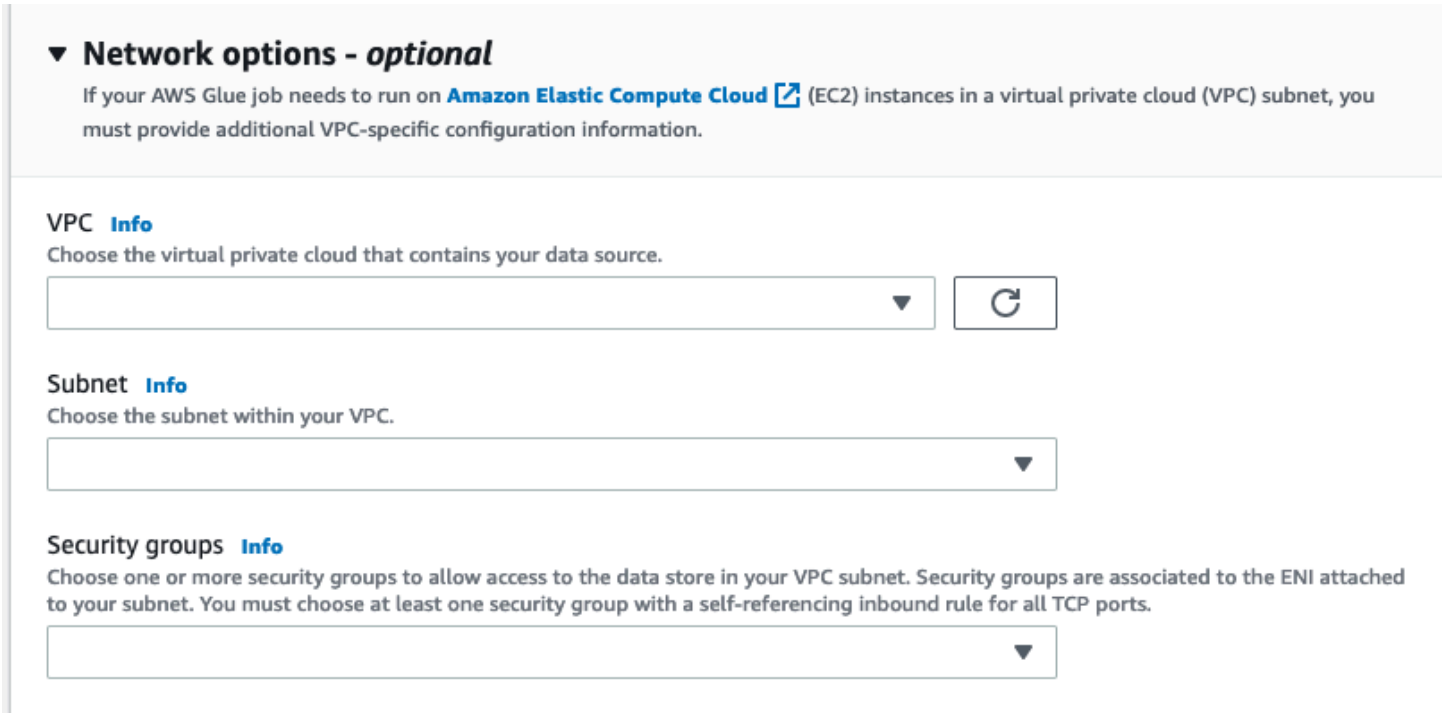
   - SHA256withRSA
   - SHA384withRSA
   - SHA512withRSA



(선택 사항) 네트워크 옵션((Optional) Network options)

다음은 VPC, 서브넷 및 보안 그룹을 구성하는 선택적 단계입니다. AWS Glue 작업이 가상 프라이빗 클라우드(VPC) 서브넷의 Amazon EC2 인스턴스에서 실행되어야 하는 경우 추가 VPC특정 구성 정보를 제공해야 합니다.

1. 데이터 소스가 포함된 VPC (가상 프라이빗 클라우드)를 선택합니다.
2. 에서 서브넷을 선택합니다VPC.
3. VPC 서브넷의 데이터 스토어에 대한 액세스를 허용하려면 하나 이상의 보안 그룹을 선택합니다. 보안 그룹은 서브넷에 ENI 연결된 에 연결됩니다. 모든 TCP 포트에 대해 자체 참조 인바운드 규칙이 있는 보안 그룹을 하나 이상 선택해야 합니다.



사용자 정의 커넥터로 작업 작성

AWS Glue Studio에서 데이터 원본 노드와 데이터 대상 노드 둘 다에 커넥터와 연결을 사용할 수 있습니다.

주제

- 데이터 원본에 커넥터를 사용하는 작업 생성
- 커넥터를 사용하는 노드의 소스 속성 구성
- 커넥터를 사용하는 노드의 대상 속성 구성

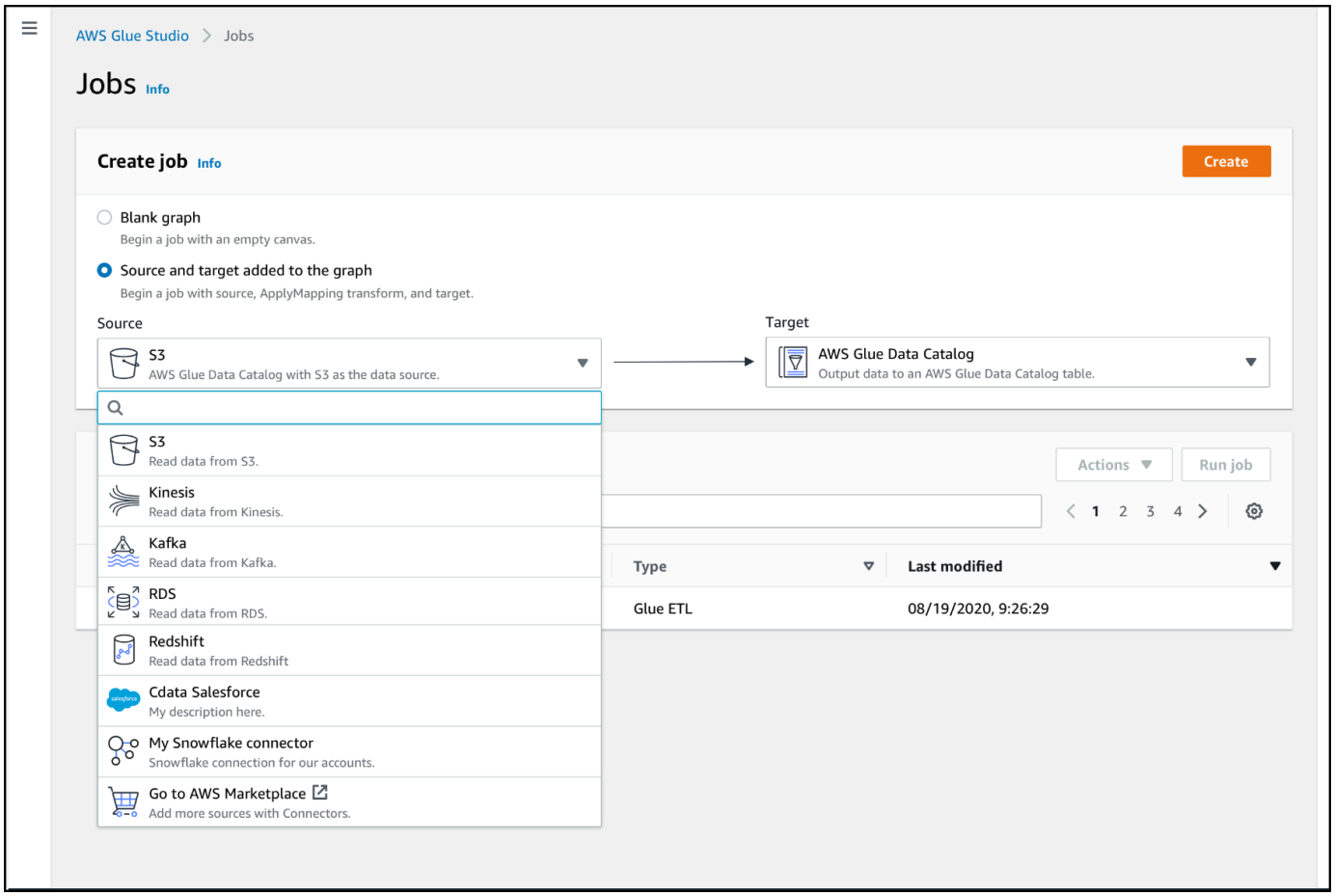
데이터 원본에 커넥터를 사용하는 작업 생성

새 작업을 생성할 때 데이터 원본 및 데이터 대상에 대한 커넥터를 선택할 수 있습니다.

데이터 원본 또는 데이터 대상에 대한 커넥터를 사용하는 작업을 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.
2. [커넥터(Connectors)] 페이지의 [연결(Your connections)] 리소스 목록에서 작업에 사용할 연결을 선택한 다음 [작업 생성(Create job)]을 선택합니다.

또는 AWS Glue Studio 작업(Jobs) 페이지의 작업 생성(Create job)에서 그래프에 추가된 소스 및 대상(Source and target added to the graph)을 선택합니다. [소스(Source)] 드롭다운 목록에서 작업에 사용하려는 사용자 정의 커넥터를 선택합니다. [대상(Target)]에 대한 커넥터를 선택할 수도 있습니다.



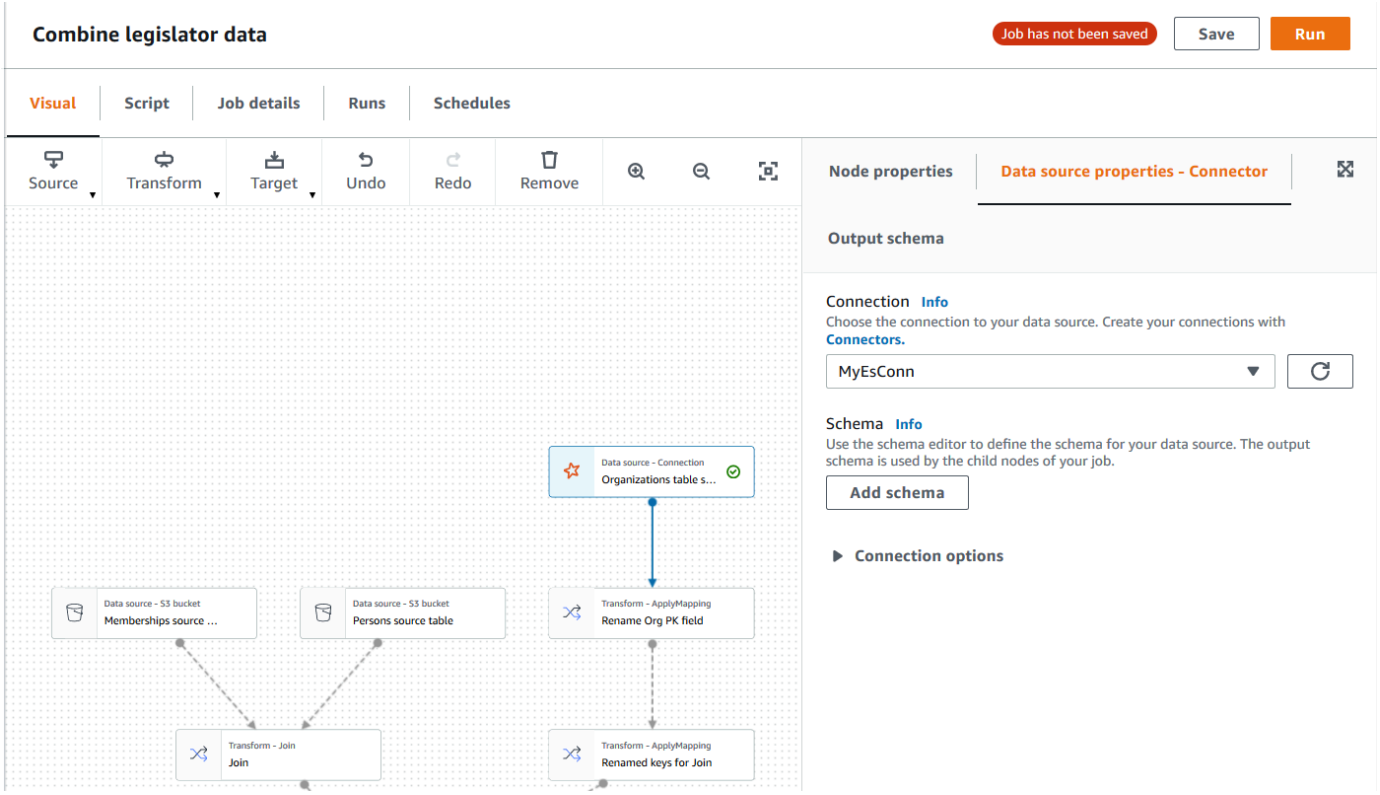
3. [생성(Create)]을 선택하여 시각적 작업 편집기를 엽니다.
4. 커넥터를 사용하는 노드의 소스 속성 구성에 설명된 대로 데이터 원본 노드를 구성합니다.
5. AWS Glue Studio에서 시각적 ETL 작업 시작에 설명된 대로 변환, 추가 데이터 스토어 및 데이터 대상을 추가하여 ETL 작업을 계속 생성합니다.
6. 작업 속성 수정에 설명된 대로 작업 속성을 구성하여 작업 실행 환경을 사용자 지정합니다.
7. 작업을 저장하고 실행합니다.

커넥터를 사용하는 노드의 소스 속성 구성

데이터 원본에 대해 커넥터를 사용하는 작업을 생성한 후 시각적 작업 편집기는 커넥터에 대해 구성된 데이터 원본 노드가 있는 작업 그래프를 표시합니다. 해당 노드에 대한 데이터 원본 속성을 구성해야 합니다.

커넥터를 사용하는 데이터 원본 노드의 속성을 구성하려면

1. 작업 그래프에서 커넥터 데이터 원본 노드를 선택하거나 새 노드를 추가하고 [노드 유형(Node type)]에 대한 커넥터를 선택하십시오. 그런 다음 오른쪽의 노드 세부 정보 패널에서 [데이터 원본 속성(Data source properties)] 탭을 선택합니다(아직 선택하지 않은 경우).



2. [데이터 원본 속성(Data source properties)] 탭에서 이 작업에 사용할 연결을 선택합니다.

   각 연결 유형에 필요한 추가 정보를 입력합니다.

   JDBC

   - [데이터 원본 입력 유형(Data source input type)]: 테이블 이름 또는 SQL 쿼리를 데이터 원본으로 제공하도록 선택합니다. 선택 항목에 따라 다음 추가 정보를 제공해야 합니다.

- [테이블 이름(Table name)]: 데이터 원본에 있는 테이블의 이름입니다. 데이터 원본에서 테이블이라는 용어를 사용하지 않는 경우 사용자 정의 커넥터 사용 정보(AWS Marketplace에서 사용 가능)에 표시된 대로 적절한 데이터 구조의 이름을 제공합니다.
- 필터 조건자(Filter predicate)[]: 데이터의 하위 집합을 검색하는 데 사용되는 WHERE 절과 유사하게 데이터 원본을 읽을 때 사용할 조건 절입니다.
- [쿼리 코드(Query code)]: 데이터 원본에서 특정 데이터 집합을 검색하는 데 사용할 SQL 쿼리를 입력합니다. 기본 SQL 쿼리의 예는 다음과 같습니다.

```
SELECT column_list FROM
                          table_name WHERE where_clause
```

- 스키마(Schema): AWS Glue Studio는 Data Catalog 테이블에서 메타데이터 정보를 검색하는 대신 연결에 저장된 정보를 사용하여 데이터 원본에 액세스하므로 데이터 원본에 대한 스키마 메타데이터를 제공해야 합니다. [스키마 추가(Add schema)]를 선택하여 스키마 편집기를 엽니다.

  스키마 편집기를 사용하는 방법에 대한 지침은 사용자 정의 변환 노드에서 스키마 편집 섹션을 참조하세요.

- [파티션 열(Partition column)]: (선택 사항) [파티션 열(Partition column)], [하한(Lower bound)], [상한(Upper bound)] 및 [파티션 수(Number of partitions)]에 대한 값을 제공하여 데이터 읽기를 분할하도록 선택할 수 있습니다.

  lowerBound 및 upperBound 값은 테이블의 행을 필터링하는 것이 아니라 파티션 스트라이드를 결정하는 데 사용됩니다. 테이블의 모든 행이 분할되어 반환됩니다.

  > **Note**
  >
  > 열 분할은 데이터를 읽는 데 사용되는 쿼리에 추가 분할 조건을 추가합니다. 테이블 이름 대신 쿼리를 사용하는 경우 쿼리가 지정된 분할 조건에서 작동하는지 확인해야 합니다. 예:
  >
  > - 쿼리 포맷이 "SELECT col1 FROM table1"이면 파티션 열을 사용하는 쿼리 끝에 WHERE 절을 추가하여 쿼리를 테스트합니다.
  > - 쿼리 포맷이 "SELECT col1 FROM table1 WHERE col2=val"이면 AND와 파티션 열을 사용하는 표현식으로 WHERE 절을 확장하여 쿼리를 테스트합니다.

- [데이터 유형 캐스팅(Data type casting)]: 데이터 원본에서 JDBC에서 사용할 수 없는 데이터 유형을 사용하는 경우 이 섹션을 사용하여 데이터 원본의 데이터 유형을 JDBC 데이터 유형으로 변환하는 방법을 지정합니다. 최대 50개의 데이터 유형 변환을 지정할 수 있습니다. 동일한 데이터 유형을 사용하는 데이터 원본의 모든 열을 동일한 방식으로 변환합니다.

  예를 들어 `Float` 데이터 유형을 사용하는 데이터 원본에 3개의 열이 있고 `Float` 데이터 유형을 JDBC `String` 데이터 유형으로 변환해야 한다고 지정하면 `Float` 데이터 유형을 사용하는 3개의 열이 모두 `String` 데이터 유형으로 변환됩니다.

- [작업 북마크 키(Job bookmark keys)]: AWS Glue는 작업 북마크로 상태 정보를 유지하고 이전 데이터의 재처리를 방지합니다. 하나 이상의 열을 북마크 키로 지정합니다. AWS Glue Studio는 북마크 키를 사용하여 이전에 ETL 작업을 실행할 때 이미 처리된 데이터를 추적합니다. 사용자 정의 북마크 키에 사용하는 모든 열은 엄격하게 단조롭게 증가하거나 감소해야 하지만 간격은 허용됩니다.

  여러 북마크 키를 입력하면 결합되어 단일 복합 키를 형성합니다. 복합 작업 북마크 키에 중복 열이 포함되면 안 됩니다. 북마크 키를 지정하지 않으면 AWS Glue Studio는 기본값으로 기본 키를 북마크 키로 사용합니다. 단, 기본 키가 간격 없이 순차적으로 증가하거나 감소해야 합니다. 테이블에 기본 키가 없지만 작업 북마크 속성이 활성화된 경우 사용자 정의 작업 북마크 키를 제공해야 합니다. 그렇지 않으면 기본값으로 사용할 기본 키 검색이 실패하고 작업 실행이 실패합니다.

- [작업 북마크 키 정렬 순서(Job bookmark keys sorting order)]: 키 값이 순차적으로 증가할지 또는 감소할지를 선택합니다.

Spark

- 스키마(Schema): AWS Glue Studio는 Data Catalog 테이블에서 메타데이터 정보를 검색하는 대신 연결에 저장된 정보를 사용하여 데이터 원본에 액세스하므로 데이터 원본에 대한 스키마 메타데이터를 제공해야 합니다. [스키마 추가(Add schema)]를 선택하여 스키마 편집기를 엽니다.

  스키마 편집기를 사용하는 방법에 대한 지침은 사용자 정의 변환 노드에서 스키마 편집 섹션을 참조하세요.

- [연결 옵션(Connection options)]: 추가 연결 정보 또는 옵션을 제공하기 위해 필요에 따라 추가 키-값 페어를 입력합니다. 예를 들어 데이터베이스 이름, 테이블 이름, 사용자 이름 및 암호를 입력할 수 있습니다.

예를 들어 OpenSearch의 경우 the section called “ 자습서: AWS Glue Connector for Elasticsearch 사용 ”의 설명에 따라 다음 키-값 페어를 입력합니다.

- `es.net.http.auth.user` : *`username`*
- `es.net.http.auth.pass` : *`password`*
- `es.nodes` : `https://`*`<Elasticsearch endpoint>`*
- `es.port` : `443`
- path: *`<Elasticsearch resource>`*
- `es.nodes.wan.only` : `true`

사용할 최소 연결 옵션의 예는 GitHub의 샘플 테스트 스크립트 MinimalSparkConnectorTest.scala를 참조하세요. 이 스크립트는 일반적으로 연결에서 제공하는 연결 옵션을 보여줍니다.

Athena

- [테이블 이름(Table name)]: 데이터 원본에 있는 테이블의 이름입니다. Athena-CloudWatch Logs에서 읽기 위해 커넥터를 사용하는 경우 테이블 이름 `all_log_streams`를 입력합니다.
- [Athena 스키마 이름(Athena schema name)]: Athena 데이터 원본에서 테이블을 포함하는 데이터베이스에 해당하는 스키마를 선택합니다. Athena-CloudWatch Logs에서 읽기 위해 커넥터를 사용하는 경우 `/aws/glue/`*`name`*과 유사한 스키마 이름을 입력합니다.
- 스키마(Schema): AWS Glue Studio는 Data Catalog 테이블에서 메타데이터 정보를 검색하는 대신 연결에 저장된 정보를 사용하여 데이터 원본에 액세스하므로 데이터 원본에 대한 스키마 메타데이터를 제공해야 합니다. [스키마 추가(Add schema)]를 선택하여 스키마 편집기를 엽니다.

  스키마 편집기를 사용하는 방법에 대한 지침은 사용자 정의 변환 노드에서 스키마 편집 섹션을 참조하세요.
- [추가 연결 옵션(Additional connection options)]: 추가 연결 정보 또는 옵션을 제공하기 위해 필요에 따라 추가 키-값 페어를 입력합니다.

예를 들어, https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/Athena의 `README.md` 파일을 참조하세요. 이 문서의 단계에서 샘플 코드는 최소한의 필수 연결 옵션인 `tableName`, `schemaName` 및 `className`을

보여줍니다. 코드 예제에서는 이러한 옵션을 `optionsMap` 변수의 일부로 지정하지만 연결에 대해 지정한 다음 연결을 사용할 수 있습니다.

3. (선택 사항) 필수 정보를 제공한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터 원본에 대한 결과 데이터 스키마를 볼 수 있습니다. 이 탭에 표시된 스키마는 작업 그래프에 추가하는 모든 하위 노드에서 사용됩니다.
4. (선택 사항) 노드 속성과 데이터 원본 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 데이터 원본에서 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

커넥터를 사용하는 노드의 대상 속성 구성

데이터 대상 유형에 커넥터를 사용하는 경우 데이터 대상 노드의 속성을 구성해야 합니다.

커넥터를 사용하는 데이터 대상 노드의 속성을 구성하려면

1. 작업 그래프에서 커넥터 데이터 대상 노드를 선택합니다. 그런 다음 오른쪽의 노드 세부 정보 패널에서 [데이터 대상 속성(Data target properties)] 탭을 선택합니다(아직 선택하지 않은 경우).
2. [데이터 대상 속성(Data target properties)] 탭에서 대상에 쓰는 데 사용할 연결을 선택합니다.

   각 연결 유형에 필요한 추가 정보를 입력합니다.

   JDBC

   - [연결(Connection)]: 커넥터에 사용할 연결을 선택합니다. 연결을 생성하는 방법에 대한 자세한 내용은 커넥터에 대한 연결 생성 섹션을 참조하세요.
   - [테이블 이름(Table name)]: 데이터 대상에 있는 테이블의 이름입니다. 데이터 대상에서 테이블이라는 용어를 사용하지 않는 경우 사용자 정의 커넥터 사용 정보(AWS Marketplace에서 사용 가능)에 표시된 대로 적절한 데이터 구조의 이름을 제공합니다.
   - [배치 크기(Batch size)](선택 사항): 단일 작업으로 대상 테이블에 삽입할 행 또는 레코드 수를 입력합니다. 기본값은 1000행입니다.

   Spark

   - [연결(Connection)]: 커넥터에 사용할 연결을 선택합니다. 이전에 연결을 생성하지 않은 경우 [연결 생성(Create connection)]을 선택하여 생성합니다. 연결을 생성하는 방법에 대한 자세한 내용은 커넥터에 대한 연결 생성 섹션을 참조하세요.

- [연결 옵션(Connection options)]: 추가 연결 정보 또는 옵션을 제공하기 위해 필요에 따라 추가 키-값 페어를 입력합니다. 데이터베이스 이름, 테이블 이름, 사용자 이름 및 암호를 입력할 수 있습니다.

  예를 들어 OpenSearch의 경우 the section called " 자습서: AWS Glue Connector for Elasticsearch 사용 "의 설명에 따라 다음 키-값 페어를 입력합니다.

  - `es.net.http.auth.user` : *`username`*
  - `es.net.http.auth.pass` : *`password`*
  - `es.nodes` : `https://`*`<Elasticsearch endpoint>`*
  - `es.port` : `443`
  - path: *`<Elasticsearch resource>`*
  - `es.nodes.wan.only` : `true`

  사용할 최소 연결 옵션의 예는 GitHub의 샘플 테스트 스크립트 MinimalSparkConnectorTest.scala를 참조하세요. 이 스크립트는 일반적으로 연결에서 제공하는 연결 옵션을 보여줍니다.

3. 필수 정보를 제공한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터 원본에 대한 결과 데이터 스키마를 볼 수 있습니다.

커넥터 및 연결 관리

AWS Glue의 커넥터 페이지를 사용하여 커넥터와 연결을 관리합니다.

주제

- 커넥터 및 연결 세부 정보 보기
- 커넥터 및 연결 편집
- 커넥터 및 연결 삭제
- 커넥터에 대한 구독 취소

커넥터 및 연결 세부 정보 보기

[커넥터(Connectors)] 페이지의 [커넥터(Your connectors)] 및 [연결(Your connections)] 리소스 테이블에서 커넥터 및 연결에 대한 요약 정보를 볼 수 있습니다. 세부 정보를 보려면 다음 단계를 수행합니다.

커넥터 또는 연결 세부 정보를 보려면

1. AWS Glue Studio 콘솔의 콘솔 탐색 창에서 커넥터(Connectors)를 선택합니다.
2. 자세한 정보를 보려는 커넥터 또는 연결을 선택합니다.
3. [작업(Actions)]을 선택하고 [세부 정보 보기(View details)]를 선택하여 해당 커넥터 또는 연결에 대한 세부 정보 페이지를 엽니다.
4. 세부 정보 페이지에서 커넥터 또는 연결을 Edit(편집)하거나 Delete(삭제)하도록 선택할 수 있습니다.

   - 커넥터의 경우 [연결 생성(Create connection)]을 선택하여 커넥터를 사용하는 새 연결을 생성할 수 있습니다.
   - 연결의 경우 [작업 생성(Create job)]을 클릭하여 연결을 사용하는 작업을 생성할 수 있습니다.

커넥터 및 연결 편집

[커넥터(Connectors)] 페이지를 사용하여 커넥터와 연결에 저장된 정보를 변경합니다.

커넥터 또는 연결을 수정하려면

1. AWS Glue Studio 콘솔의 콘솔 탐색 창에서 커넥터(Connectors)를 선택합니다.
2. 변경하려는 커넥터나 연결을 선택합니다.
3. 작업을 선택한 후 편집을 선택합니다.

   [세부 정보 보기(View details)]를 선택하고 커넥터 또는 연결 세부 정보 페이지에서 [편집(Edit)]을 선택할 수도 있습니다.
4. [커넥터 편집(Edit connector)] 또는 [연결 편집(Edit connection)] 페이지에서 정보를 업데이트한 다음 [저장(Save)]을 선택합니다.

커넥터 및 연결 삭제

[커넥터(Connectors)] 페이지를 사용하여 커넥터와 연결을 삭제합니다. 커넥터를 삭제하면 해당 커넥터에 대해 생성된 연결도 모두 삭제되어야 합니다.

AWS Glue Studio에서 커넥터를 제거하려면

1. AWS Glue Studio 콘솔의 콘솔 탐색 창에서 커넥터(Connectors)를 선택합니다.
2. 삭제하려는 커넥터나 연결을 선택합니다.

3. 작업을 선택한 후 삭제를 선택합니다.

   [세부 정보 보기(View details)]를 선택하고 커넥터 또는 연결 세부 정보 페이지에서 [삭제(Delete)]를 선택할 수도 있습니다.

4. **Delete**를 입력하여 커넥터 또는 연결 제거를 확인한 다음 [삭제(Delete)]를 선택합니다.

   커넥터를 삭제하면 해당 커넥터에 대해 생성된 연결도 모두 삭제됩니다.

삭제된 연결을 사용하는 모든 작업은 더 이상 작동하지 않습니다. 작업을 편집하여 다른 데이터 스토어를 사용하거나 작업을 제거할 수 있습니다. 작업을 삭제하는 방법에 대한 자세한 정보는 작업 삭제 섹션을 참조하세요.

커넥터를 삭제해도 AWS Marketplace에서 커넥터에 대한 구독이 취소되지는 않습니다. 삭제된 커넥터에 대한 구독을 제거하려면 커넥터에 대한 구독 취소의 지침을 따릅니다.

커넥터에 대한 구독 취소

AWS Glue Studio에서 연결과 커넥터를 삭제한 후 커넥터가 더 이상 필요하지 않은 경우 AWS Marketplace에서 구독을 취소할 수 있습니다.

> **Note**
>
> 커넥터에 대한 구독을 취소해도 계정에서 커넥터나 연결이 제거되지는 않습니다. 커넥터 및 관련 연결을 사용하는 모든 작업은 더 이상 커넥터를 사용할 수 없으며 실패합니다.
> AWS Marketplace에서 커넥터를 구독 취소하거나 다시 구독하기 전에 해당 AWS Marketplace 제품과 연결된 기존 연결과 커넥터를 삭제해야 합니다.

AWS Marketplace에서 커넥터 구독을 취소하려면

1. https://console.aws.amazon.com/marketplace에서 AWS Marketplace 콘솔에 로그인합니다.
2. [구독 관리(Manage subscriptions)]를 선택합니다.
3. [구독 관리(Manage subscriptions)] 페이지에서 취소하려는 커넥터 구독 옆에 있는 [관리(Manage)]를 선택합니다.
4. 그런 다음 [작업(Actions)]을 선택하고 [구독 취소(Cancel Subscription)]를 선택합니다.
5. 실행 중인 인스턴스가 계정에 청구됨을 확인하는 확인란을 선택한 다음 [예, 구독을 취소합니다(Yes, cancel subscription)]를 선택합니다.

사용자 정의 커넥터 개발

데이터 스토어에서 데이터를 읽거나 데이터를 쓰고 AWS Glue Studio 작업에 사용할 데이터를 포맷하는 코드를 작성할 수 있습니다. Spark, Athena 및 JDBC 데이터 스토어용 커넥터를 생성할 수 있습니다. GitHub에 게시된 샘플 코드는 구현해야 하는 기본 인터페이스에 대한 개요를 제공합니다.

커넥터 코드를 생성하려면 로컬 개발 환경이 필요합니다. 모든 IDE 또는 명령줄 편집기를 사용하여 커넥터를 작성할 수 있습니다. 개발 환경의 예는 다음과 같습니다.

- AWS Glue Developer Guide의 Developing Locally with Scala에 설명된 대로 로컬 AWS Glue ETL Maven 라이브러리가 있는 로컬 Scala 환경.
- IntelliJ IDE(https://www.jetbrains.com/idea/에서 다운로드).

주제

- Spark 커넥터 개발
- Athena 커넥터 개발
- JDBC 커넥터 개발
- AWS Glue Studio에서 사용자 지정 커넥터를 사용하는 예제
- AWS Marketplace용 AWS Glue 커넥터 개발

Spark 커넥터 개발

Spark DataSource API V2(Spark 2.4)로 Spark 커넥터를 생성하여 데이터를 읽을 수 있습니다.

사용자 정의 Spark 커넥터 생성

Spark 커넥터 개발을 위한 AWS Glue GitHub 샘플 라이브러리(https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/Spark/README.md에 위치)의 단계를 따르세요.

Athena 커넥터 개발

AWS Glue 및 AWS Glue Studio에서 사용자 지정 데이터 원본을 쿼리하는 데 사용할 Athena 커넥터를 생성할 수 있습니다.

사용자 정의 Athena 커넥터 생성

Athena 커넥터 개발을 위한 AWS Glue GitHub 샘플 라이브러리(https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/Athena에 위치)의 단계를 따르세요.

JDBC 커넥터 개발

JDBC를 사용하여 데이터 스토어에 액세스하는 커넥터를 생성할 수 있습니다.

사용자 정의 JDBC 커넥터를 생성하려면

1. 로컬 개발 환경에 AWS Glue Spark 런타임 라이브러리를 설치합니다. AWS Glue GitHub 샘플 라이브러리(https://github.com/aws-samples/aws-glue-samples/tree/master/GlueCustomConnectors/development/GlueSparkRuntime/README.md)의 지침을 참조하세요.
2. 데이터 원본에서 데이터 검색을 담당하는 JDBC 드라이버를 구현합니다. Java SE 8에 대한 Java 설명서를 참조하세요.

   AWS Glue Studio가 커넥터를 찾는 데 사용하는 코드 내의 진입점을 생성합니다. [클래스 이름(Class name)] 필드는 JDBC 드라이버의 전체 경로여야 합니다.
3. `GlueContext` API를 사용하여 커넥터로 데이터를 읽습니다. 사용자는 필요한 경우 AWS Glue Studio 콘솔에서 더 많은 입력 옵션을 추가하여 데이터 원본에 대한 연결을 구성할 수 있습니다. 사용자 정의 JDBC 커넥터를 사용하여 JDBC 데이터베이스에서 읽고 쓰는 방법을 보여주는 코드 예제는 사용자 정의 및 AWS Marketplace connectionType 값을 참조하세요.

AWS Glue Studio에서 사용자 지정 커넥터를 사용하는 예제

사용자 정의 커넥터 사용 예는 다음 블로그를 참조하세요.

- AWS Glue를 사용하여 데이터 저장소용 사용자 지정 커넥터 개발, 테스트 및 배포
- Apache Hudi: AWS Glue 사용자 정의 커넥터를 사용하여 Apache Hudi 테이블에 쓰기
- Google BigQuery: AWS Glue 사용자 정의 커넥터를 사용하여 Google BigQuery에서 Amazon S3로 데이터 마이그레이션
- Snowflake(JDBC): Snowflake 및 AWS Glue를 사용하여 데이터 변환 수행
- SingleStore: SingleStore 및 AWS Glue를 사용하여 빠른 ETL 구축
- Salesforce:AWS Glue와 함께 CData JDBC 사용자 정의 커넥터를 사용하여 Salesforce 데이터를 Amazon S3에 수집 -
- MongoDB: Amazon DocumentDB(MongoDB 호환성 포함) 및 MongoDB를 사용하여 AWS Glue Spark ETL 작업 구축

- Amazon Relational Database Service (Amazon RDS): Amazon RDS용 자체 JDBC 드라이버를 가져와 AWS Glue Spark ETL 작업 구축
- MySQL(JDBC): https://github.com/aws-samples/aws-glue-samples/blob/master/GlueCustomConnectors/development/Spark/SparkConnectorMySQL.scala

AWS Marketplace용 AWS Glue 커넥터 개발

AWS 파트너는 사용자 정의 커넥터를 생성하고 AWS Marketplace에 업로드하여 AWS Glue 고객에게 판매할 수 있습니다.

커넥터 코드를 개발하는 프로세스는 사용자 정의 커넥터와 동일하지만 커넥터 코드를 업로드하고 확인하는 프로세스가 더 세부적입니다. GitHub 웹 사이트의 AWS Marketplace용 커넥터 생성의 지침을 참조하세요.

AWS Glue Studio에서 커넥터와 연결을 사용하는 경우의 제한 사항

사용자 정의 커넥터 또는 AWS Marketplace의 커넥터를 사용하는 경우 다음 제한 사항에 유의하세요.

- testConnection API는 사용자 정의 커넥터에 대해 생성된 연결에서 지원되지 않습니다.
- Data Catalog 연결 암호 암호화는 사용자 정의 커넥터에서 지원되지 않습니다.
- JDBC 커넥터를 사용하는 데이터 원본 노드에 대한 필터 조건자를 지정하는 경우 작업 북마크를 사용할 수 없습니다.
- Marketplace 연결 생성은 AWS Glue Studio 사용자 인터페이스 외부에서 지원되지 않습니다.

AWS Glue 연결 테스트

가장 좋은 방법은 ETL 작업에서 AWS Glue 연결을 사용하기 전에 AWS Glue 콘솔을 사용하여 연결을 테스트하는 것입니다. AWS Glue에서는 연결할 때 파라미터를 사용하여 데이터 스토어에 액세스할 수 있는지 확인하고 오류를 보고합니다. AWS Glue 연결에 대한 자세한 내용은 데이터에 연결 단원을 참조하십시오.

AWS Glue 연결을 테스트하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 데이터 카탈로그에서 연결을 선택합니다. 탐색 창의 데이터 카탈로그 위의 데이터 연결을 선택할 수도 있습니다.

3. 연결에서 원하는 연결 옆의 확인란을 선택한 다음 작업을 선택합니다. 드롭다운 메뉴에서 연결 테스트를 선택합니다.
4. [연결 테스트(Test connection)] 대화 상자에서 역할을 선택하거나 [IAM 역할 생성]을 선택하여 AWS Identity and Access Management(IAM) 콘솔로 이동하여 새 역할을 생성합니다. 역할에는 데이터 스토어에 대한 권한이 있어야 합니다.
5. 확인을 선택합니다.

   테스트가 시작되고 완료하는 데 몇 분이 걸릴 수 있습니다. 테스트에 실패하는 경우 문제 해결을 선택하여 문제 해결 단계를 확인합니다.
6. 로그를 선택하여 CloudWatch에서 로그를 확인합니다. 로그를 보려면 필요한 IAM 권한이 있어야 합니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 AWS CloudWatch Logs용 관리형(미리 정의된) 정책을 참조하세요.

를 통과하도록 AWS 호출 구성 VPC

특수 작업 파라미터를 `disable-proxy-v2` 사용하면 를 AWS Glue 통해 Amazon S3 CloudWatch와 같은 서비스로 호출을 라우팅할 수 있습니다VPC. 기본적으로 로컬 프록시를 AWS Glue 사용하여 를 통해 AWS Glue VPC 트래픽을 전송하여 Amazon S3에서 스크립트 및 라이브러리를 다운로드하고, 로그 및 지표 게시 CloudWatch 를 위해 에 요청을 보내고, 데이터 카탈로그에 액세스하기 AWS Glue 위해 에 요청을 보냅니다. 이 프록시를 사용하면 가 Amazon S3와 같은 다른 AWS 서비스에 대한 적절한 라우팅을 구성하지 VPC 않더라도 작업이 정상적으로 작동할 수 있으며 CloudWatch, AWS Glue. AWS Glue now는 이 동작을 끌 수 있는 파라미터를 제공합니다. 자세한 내용은 에서 사용하는 작업 파라미터를 AWS Glue 참조하세요. AWS Glue 는 AWS Glue 작업의 CloudWatch 로그를 게시하는 데 로컬 프록시를 계속 사용합니다.

> **Note**
>
> - 이 기능은 AWS Glue 버전 2.0 이상의 AWS Glue 작업에서 지원됩니다. 이 기능을 사용할 때는 VPC가 NAT 또는 서비스 VPC 엔드포인트를 통해 Amazon S3로 라우팅을 구성했는지 확인해야 합니다.
> - 더 이상 사용되지 않는 작업 파라미터는 를 통해 스크립트 및 라이브러리를 다운로드하기 위해 Amazon S3로`disable-proxy`만 호출을 라우팅합니다VPC. 새 파라미터 `disable-proxy-v2`를 대신 사용하는 것이 좋습니다.

사용 예

를 사용하여 AWS Glue 작업 생성disable-proxy-v2:

```
aws glue create-job \
    --name no-proxy-job \
    --role GlueDefaultRole \
    --command "Name=glueetl,ScriptLocation=s3://my-bucket/glue-script.py" \
    --connections Connections="traffic-monitored-connection" \
    --default-arguments '{"--disable-proxy-v2" : "true"}'
```

의 JDBC 데이터 스토어에 연결 VPC

일반적으로 퍼블릭 인터넷을 통해 액세스할 수 없도록 Amazon Virtual Private Cloud(Amazon VPC) 내에 리소스를 생성합니다. 기본적으로 AWS Glue 는 내부의 리소스에 액세스할 수 없습니다VPC. 가 내 리소스 AWS Glue 에 액세스할 수 있도록 하려면 VPC 서브넷 IDs 및 보안 그룹 를 포함하는 추가 VPC 특정 구성 정보를 제공해야 VPC합니다IDs. 는 이 정보를 AWS Glue 사용하여 함수가 프라이빗 의 다른 리소스에 안전하게 연결할 수 있도록 하는 탄력적 네트워크 인터페이스를 설정합니다VPC.

VPC 엔드포인트를 사용하는 경우 라우팅 테이블에 추가합니다. 자세한 내용은 및 에 대한 인터페이스 VPC 엔드포인트 생성을 AWS Glue 참조하세요사전 조건.

Data Catalog에서 암호화를 사용하는 경우 KMS 인터페이스 엔드포인트를 생성하고 라우팅 테이블에 추가합니다. 자세한 내용은 섹션을 참조하세요.
에 대한 VPC 엔드포인트 생성 AWS KMS.

탄력적 네트워크 인터페이스를 사용하여 VPC 데이터 액세스

가 의 JDBC 데이터 스토어에 AWS Glue 연결되면 는 계정에 탄력적 네트워크 인터페이스(접두사 포함Glue_)를 VPC AWS Glue 생성하여 VPC 데이터에 액세스합니다. 이 네트워크 인터페이스가 에 연결되어 있는 한 삭제할 수 없습니다 AWS Glue. 탄력적 네트워크 인터페이스 생성의 일환으로 하나 AWS Glue 이상의 보안 그룹을 연결합니다. AWS Glue 가 네트워크 인터페이스를 생성하도록 활성화하려면 리소스와 연결된 보안 그룹이 소스 규칙을 사용하여 인바운드 액세스를 허용해야 합니다. 이 규칙은 리소스와 관련된 보안 그룹입니다. 따라서 탄력적 네트워크 인터페이스는 동일한 보안 그룹을 통해 데이터 스토어에 액세스할 수 있습니다.

가 구성 요소와 통신 AWS Glue 할 수 있도록 하려면 모든 TCP 포트에 대해 자체 참조 인바운드 규칙이 있는 보안 그룹을 지정합니다. 자체 참조 규칙을 생성하여 소스를 의 동일한 보안 그룹으로 제한할 수 VPC 있으며 모든 네트워크에 열려 있지는 않습니다. 의 기본 보안 그룹에는 에 대한 자체 참조 인바운드 규칙이 이미 있을 VPC 수 있습니다ALL Traffic.

Amazon VPC 콘솔에서 규칙을 생성할 수 있습니다. 를 통해 규칙 설정을 업데이트하려면 VPC 콘솔(https://console.aws.amazon.com/vpc/)로 AWS Management Console이동하여 적절한 보안 그룹을 선택합니다. ALL TCP의 인바운 규칙을 지정하여 원본이 동일한 보안 그룹 이름을 갖도록 합니다. 보안 그룹 규칙에 대한 자세한 내용은 의 보안 그룹을 참조하세요VPC.

지정한 서브넷의 IP 주소 범위에 속하는 프라이빗 IP 주소가 이 탄력적 네트워크 인터페이스에 할당됩니다. 네트워크 인터페이스에는 퍼블릭 IP 주소가 할당되지 않습니다. 에는 인터넷 액세스가 AWS Glue 필요합니다(예: VPC 엔드포인트가 없는 AWS 서비스에 액세스하기 위해). 내에서 네트워크 주소 변환(NAT) 인스턴스를 구성VPC하거나 Amazon VPC NAT 게이트웨이를 사용할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 NAT 게이트웨이를 참조하세요. 에 연결된 인터넷 게이트웨이를 서브넷 라우팅 테이블의 라우팅VPC으로 직접 사용할 수 없습니다. 네트워크 인터페이스에 퍼블릭 IP 주소가 있어야 하기 때문입니다.

VPC 네트워크 속성 enableDnsHostnames 및 를 true로 설정해야 enableDnsSupport 합니다. 자세한 내용은 DNS에서 사용을 참조하세요VPC.

> **Important**
>
> 인터넷 액세스가 없는 퍼블릭 서브넷 또는 프라이빗 서브넷에 데이터 스토어를 추가하지 마십시오. 대신 NAT 인스턴스 또는 Amazon VPC NAT 게이트웨이를 통해 인터넷에 액세스할 수 있는 프라이빗 서브넷에만 연결합니다.

탄력적 네트워크 인터페이스 속성

다음 속성을 제공하여 탄력적 네트워크 인터페이스를 생성합니다.

VPC

데이터 스토어VPC가 포함된 의 이름입니다.

서브넷

데이터 스토어VPC가 포함된 의 서브넷입니다.

보안 그룹

데이터 스토어와 연결된 보안 그룹입니다. 는 이러한 보안 그룹을 VPC 서브넷에 연결된 탄력적 네트워크 인터페이스와 AWS Glue 연결합니다. AWS Glue 구성 요소가 통신할 수 있도록 허용하고

다른 네트워크에서의 액세스를 방지하려면 선택한 보안 그룹이 모든 TCP 포트에 대해 자체 참조 인바운드 규칙을 지정해야 합니다.

Amazon RedshiftVPC로 를 관리하는 방법에 대한 자세한 내용은 Amazon Virtual Private Cloud(VPC)에서 클러스터 관리를 참조하세요.

Amazon Relational Database Service(Amazon RDS)를 VPC 사용하여 를 관리하는 방법에 대한 자세한 내용은 에서 Amazon RDS DB 인스턴스 작업을 참조하세요VPC.

MongoDB 또는 MongoDB Atlas 연결 사용

MongoDB 또는 MongoDB Atlas에 대한 연결을 생성한 후 ETL 작업에서 해당 연결을 사용할 수 있습니다. AWS Glue Data Catalog에 테이블을 생성하고 테이블의 `connection` 속성에 대해 MongoDB 또는 MongoDB Atlas 연결을 지정합니다.

AWS Glue는 연결 `url`과 자격 증명을 MongoDB 연결에 저장합니다. 연결 URI 형식은 다음과 같습니다.

- MongoDB의 경우: mongodb://host:port/database. 호스트는 호스트 이름, IP 주소 또는 UNIX 도메인 소켓일 수 있습니다. 연결 문자열이 포트를 지정하지 않는 경우 기본 MongoDB 포트인 27017을 사용합니다.
- MongoDB Atlas의 경우: mongodb+srv://server.example.com/database. 호스트는 DNS SRV 레코드에 해당하는 호스트 이름일 수 있습니다. SRV 형식에는 포트가 필요하지 않으며 기본 MongoDB 포트인 27017을 사용합니다.

또한 작업 스크립트에서 옵션을 지정할 수 있습니다. 자세한 내용은 the section called “MongoDB 연결” 단원을 참조하십시오.

VPC 엔드포인트를 사용하여 Amazon S3 데이터 스토어 크롤링

보안, 감사 또는 제어를 위해 Amazon Virtual Private Cloud 환경(Amazon)을 통해서만 Amazon S3 데이터 스토어 또는 Amazon S3 지원 데이터 카탈로그 테이블에 액세스할 수 있습니다VPC. Amazon Virtual Private Cloud 이 주제에서는 연결 유형을 사용하여 VPC 엔드포인트에서 Amazon S3 데이터 스토어 또는 Amazon S3 지원 데이터 카탈로그 테이블에 대한 `Network` 연결을 생성하고 테스트하는 방법을 설명합니다.

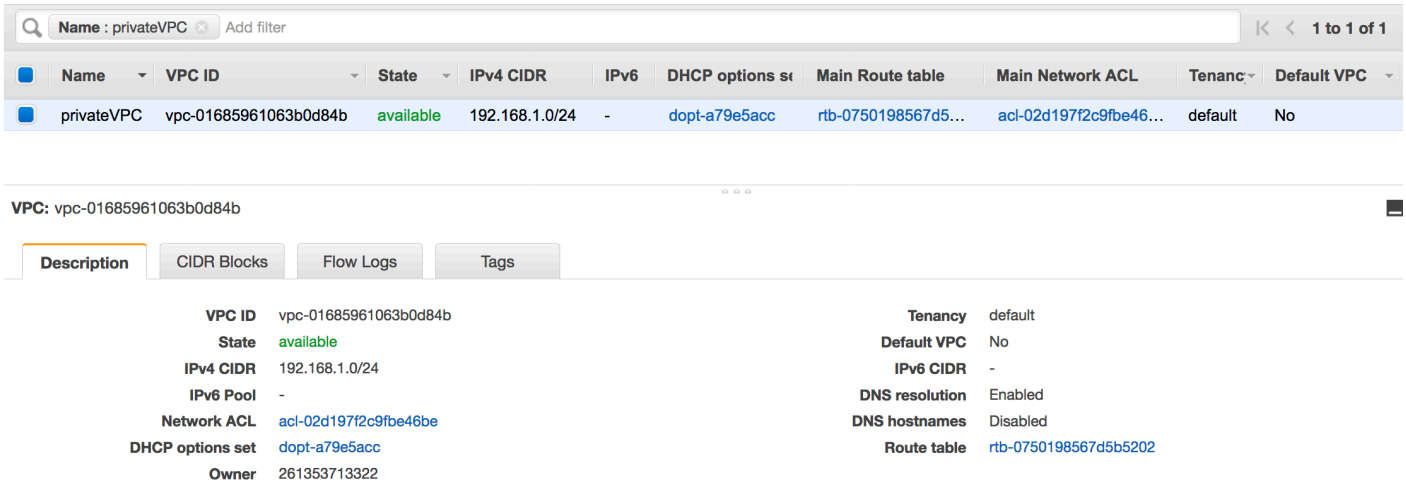
데이터 스토어에서 크롤러를 실행하려면 다음 태스크를 수행합니다.

- the section called “사전 조건”
- the section called “Amazon S3에 대한 연결 생성”
- the section called “Amazon S3에 대한 연결 테스트”
- the section called “Amazon S3 데이터 스토어의 크롤러 생성”
- the section called “크롤러 실행”

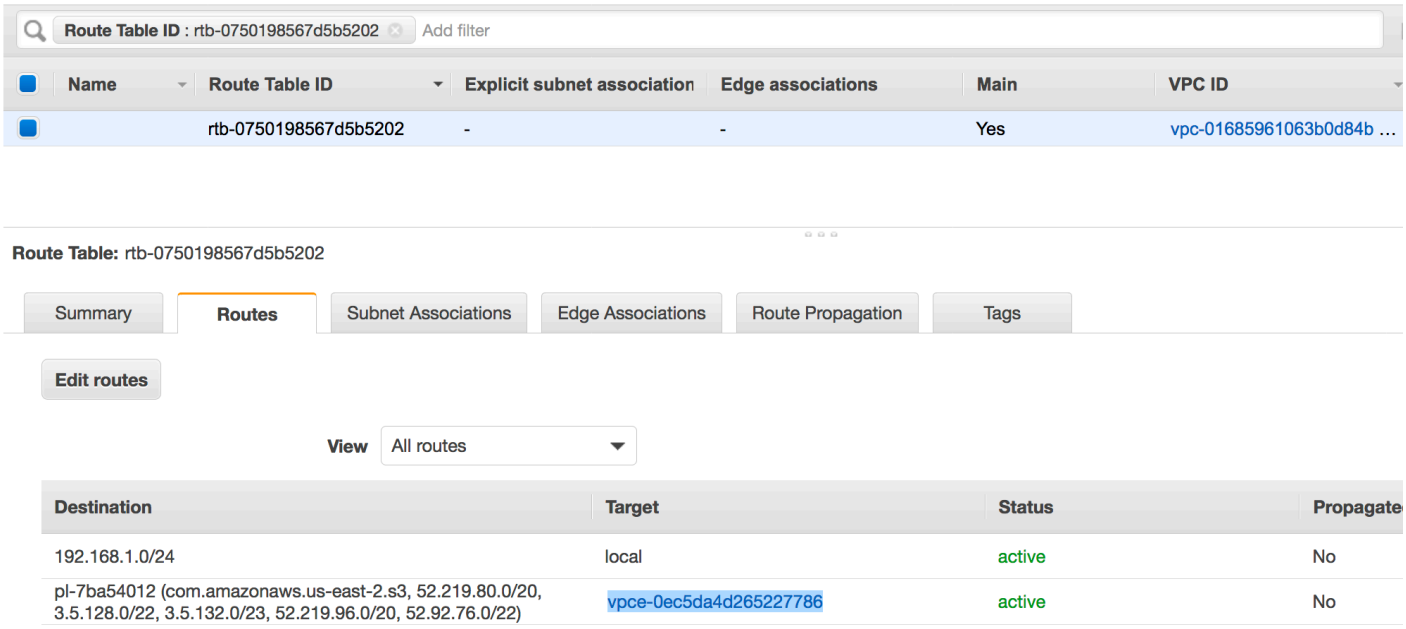
사전 조건

Amazon Virtual Private Cloud 환경(Amazon)을 통해 액세스할 Amazon S3 데이터 스토어 또는 Amazon S3 지원 데이터 카탈로그 테이블을 설정하기 위한 이러한 사전 요구 사항을 충족했는지 확인합니다VPC.

- 구성된 VPC. 예: vpc-01685961063b0d84b. 자세한 내용은 Amazon 사용 설명서의 Amazon 시작하기VPC를 참조하세요. VPC
- 에 연결된 Amazon S3 엔드포인트입니다VPC. 예: vpc-01685961063b0d84b. 자세한 내용은 Amazon 사용 설명서의 Amazon S3용 엔드포인트를 참조하세요. VPC



- VPC 엔드포인트를 가리키는 라우팅 항목입니다. 예: VPC 엔드포인트(vpce-0ec5da4d265227786)에서 사용하는 라우팅 테이블의 vpce-0ec5da4d265227786.



- 에 ACL 연결된 네트워크는 트래픽을 VPC 허용합니다.
- 에 연결된 보안 그룹은 트래픽을 VPC 허용합니다.

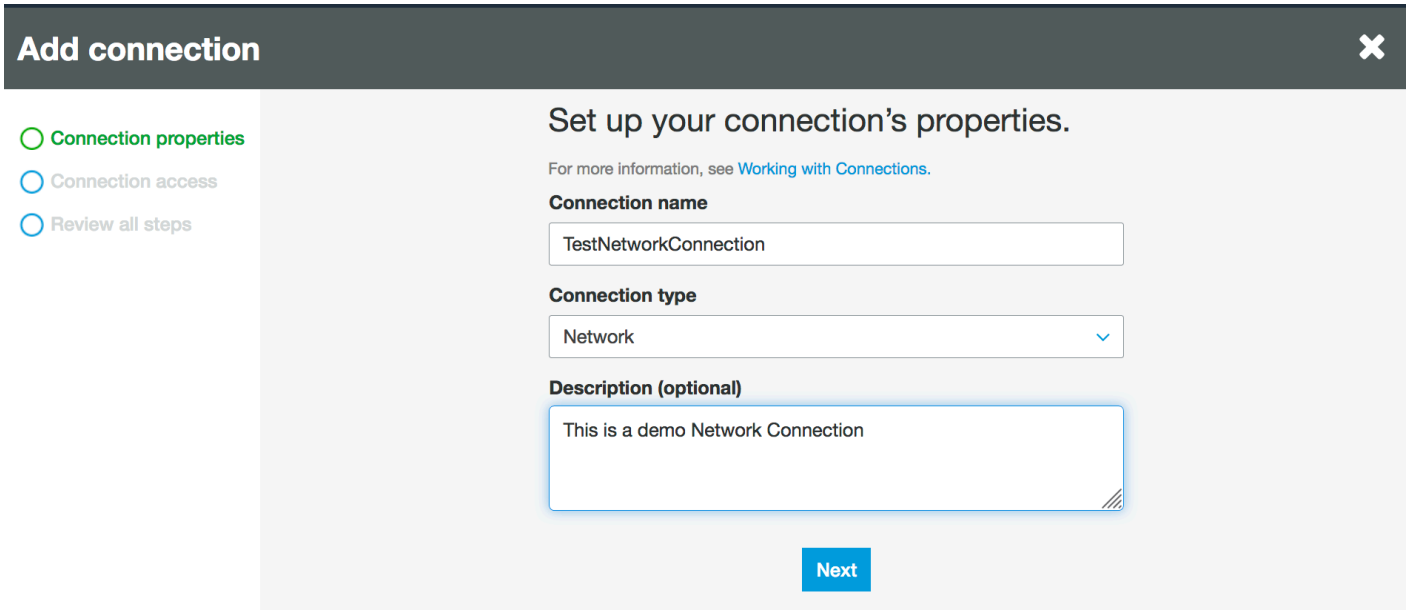
Amazon S3에 대한 연결 생성

일반적으로 퍼블릭 인터넷을 통해 액세스할 수 없도록 Amazon Virtual Private Cloud(AmazonVPC) 내에 리소스를 생성합니다. 기본적으로 AWS Glue 는 내부의 리소스에 액세스할 수 없습니다VPC. 가 내부의 리소스 AWS Glue 에 액세스할 수 있도록 하려면 VPC 서브넷 IDs 및 보안 그룹 을 포함하는 추가 VPC특정 구성 정보를 제공해야 VPC합니다IDs. `Network` 연결을 생성하려면 다음 정보를 지정해야 합니다.

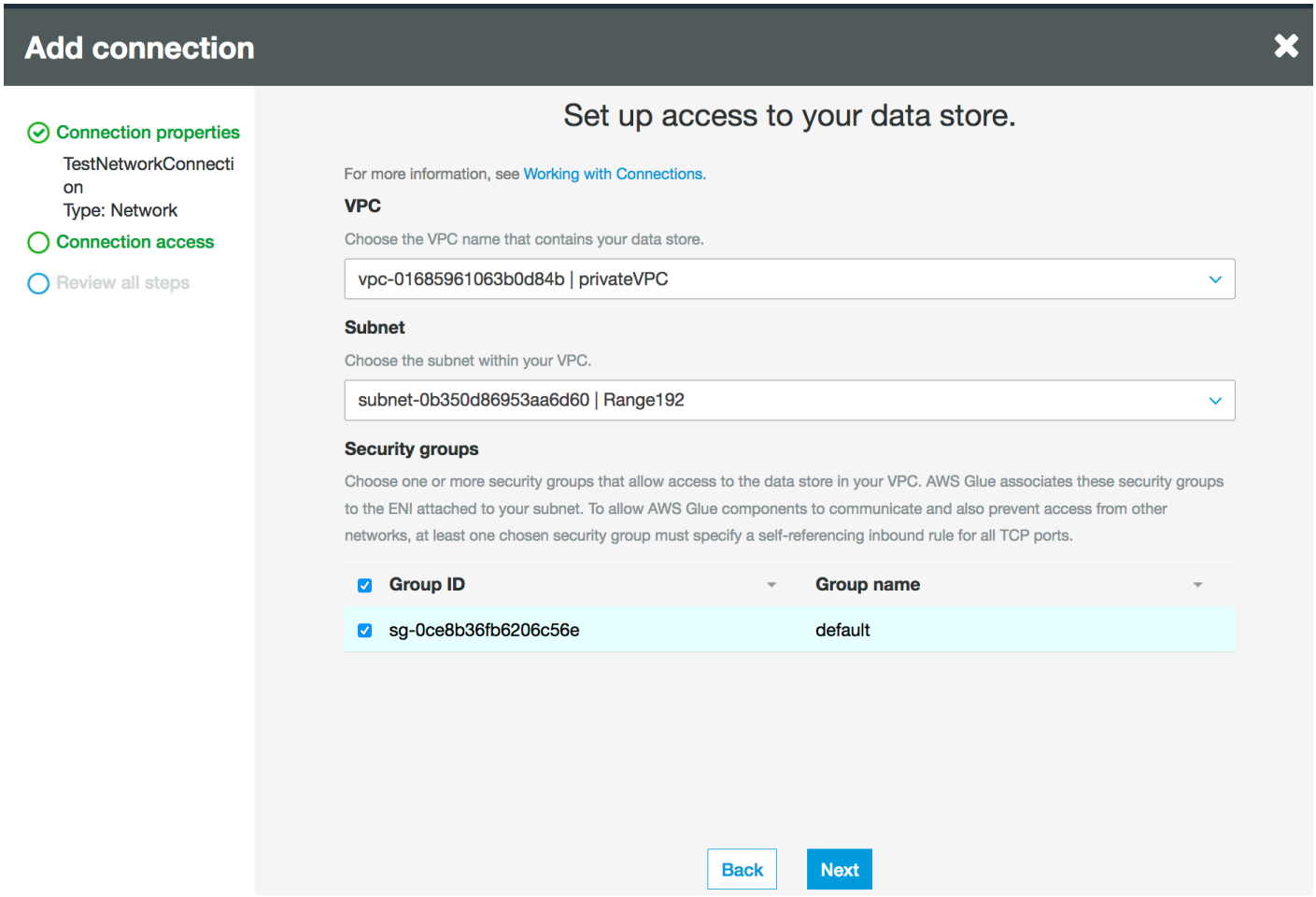
- VPC ID
- 내의 서브넷 VPC
- 보안 그룹

`Network` 연결을 설정하려면

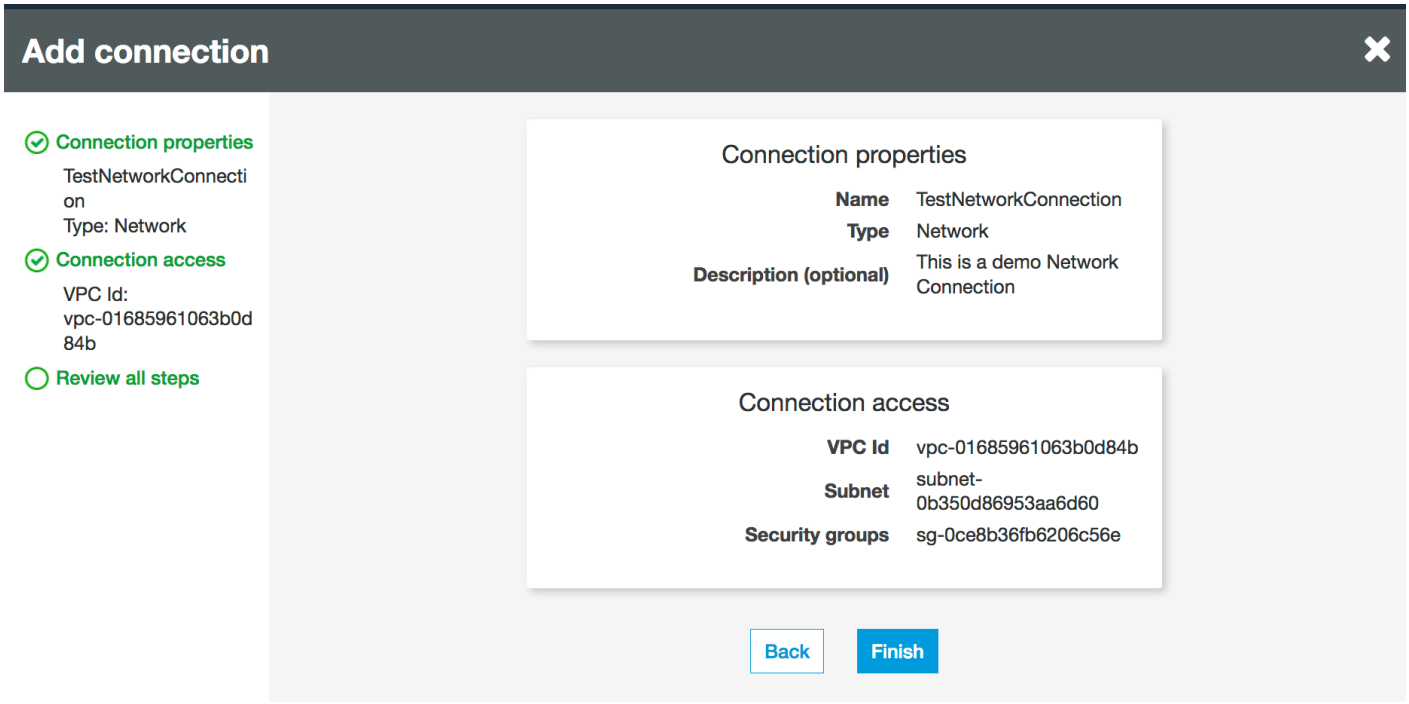
1. AWS Glue 콘솔의 탐색 창에서 [연결 추가(Add connection)]를 선택합니다.
2. 연결 이름을 입력하고 연결 유형으로 [네트워크(Network)]를 선택합니다. Next(다음)를 선택합니다.



3. VPC, 서브넷 및 보안 그룹 정보를 구성합니다.
   - VPC: 데이터 스토어가 포함된 VPC 이름을 선택합니다.
   - 서브넷: 내에서 서브넷을 선택합니다VPC.
   - 보안 그룹: 의 데이터 스토어에 대한 액세스를 허용하는 보안 그룹을 하나 이상 선택합니다VPC.



4. Next(다음)를 선택합니다.
5. 연결 정보를 확인하고 [마침(Finish)]을 선택합니다.



Amazon S3에 대한 연결 테스트

`Network` 연결을 생성한 후에는 VPC 엔드포인트에서 Amazon S3 데이터 스토어에 대한 연결을 테스트할 수 있습니다.

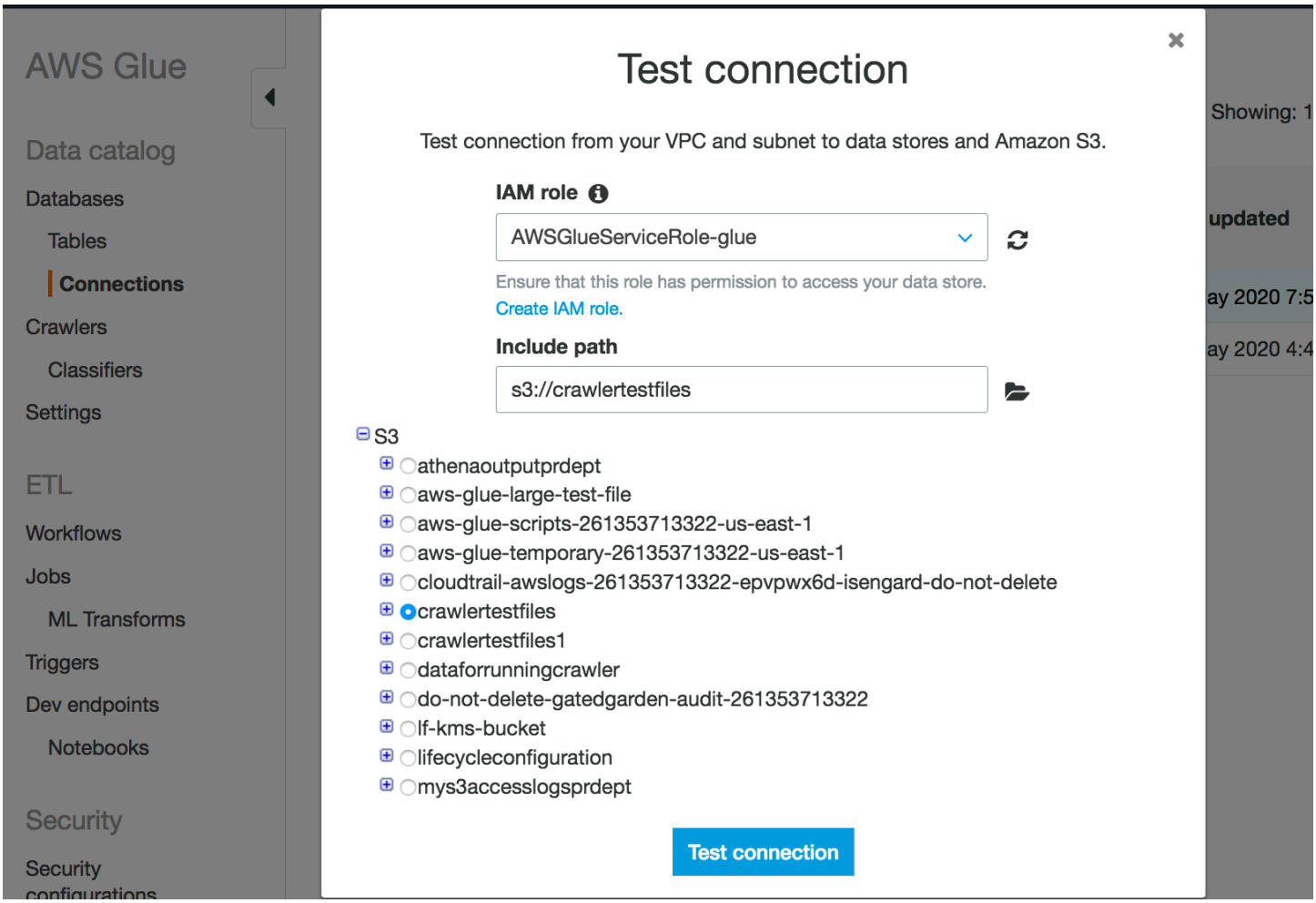
연결을 테스트할 때 다음 오류가 발생할 수 있습니다.

- INTERNET CONNECTION ERROR: 인터넷 연결 문제를 나타냅니다.
- INVALID BUCKET ERROR: Amazon S3 버킷의 문제를 나타냅니다.
- S3 CONNECTION ERROR: Amazon S3에 연결하지 못함을 나타냅니다.
- INVALID CONNECTION TYPE: 연결 유형에 예상 값이 없음을 나타냅니다. `NETWORK`
- INVALID CONNECTION TEST TYPE: 네트워크 연결 테스트 유형에 문제가 있음을 나타냅니다.
- INVALID TARGET: Amazon S3 버킷이 제대로 지정되지 않았음을 나타냅니다.

`Network` 연결을 테스트하려면

1. AWS Glue 콘솔에서 [네트워크(Network)] 연결을 선택합니다.
2. 연결 테스트를 선택합니다.
3. 이전 단계에서 생성한 IAM 역할을 선택하고 Amazon S3 버킷을 지정합니다.

4. [연결 테스트(Test connection)]를 선택하여 테스트를 시작합니다. 결과를 표시하는 데 몇 분 정도 걸릴 수 있습니다.



오류가 발생하면 다음을 확인합니다.

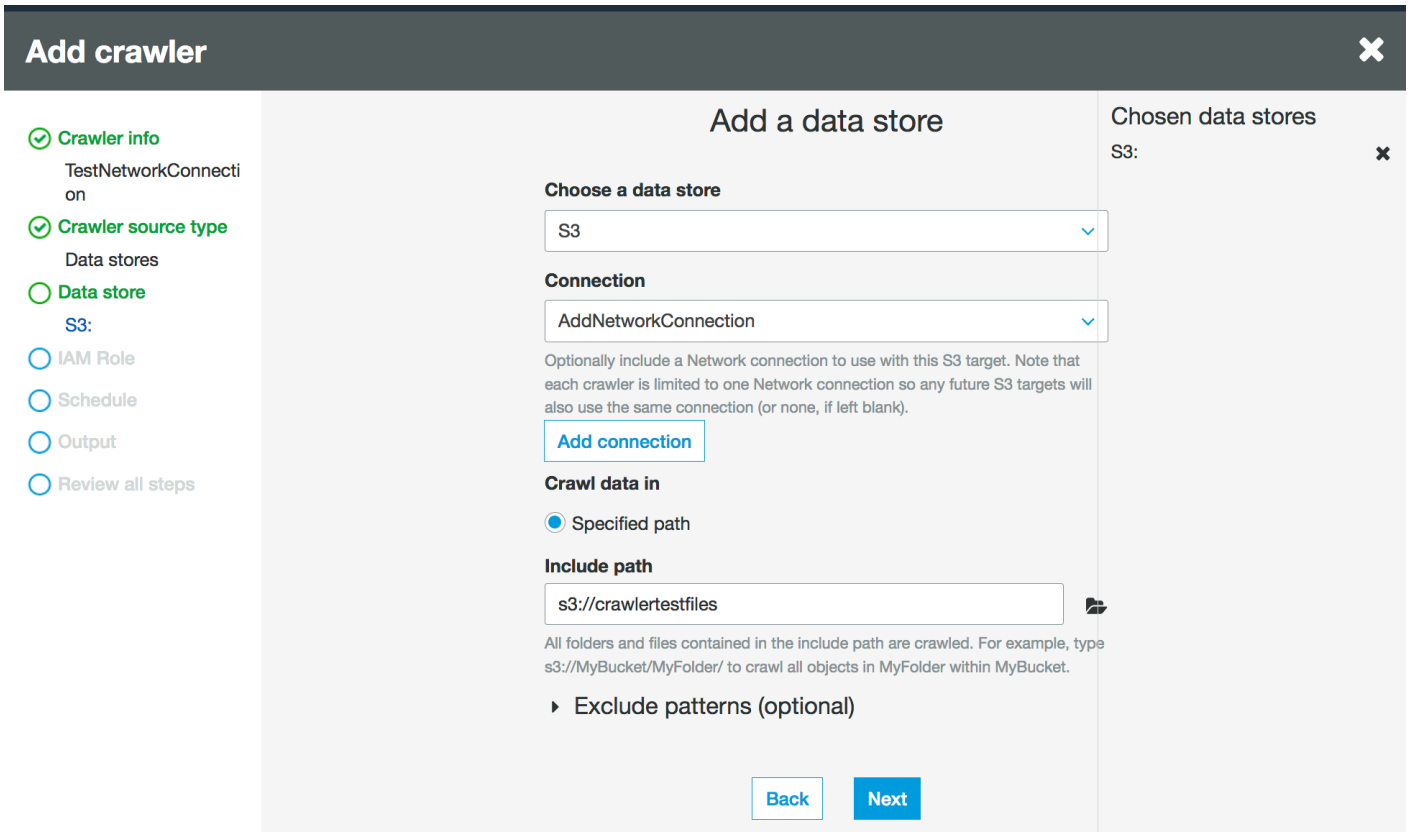
- 선택한 역할에 올바른 권한이 제공됩니다.
- 올바른 Amazon S3 버킷이 제공됩니다.
- 보안 그룹 및 네트워크는 필요한 수신 및 발신 트래픽을 ACL 허용합니다.
- VPC 지정한 가 Amazon S3 VPC 엔드포인트에 연결되어 있습니다.

연결 테스트에 성공하면 크롤러를 생성할 수 있습니다.

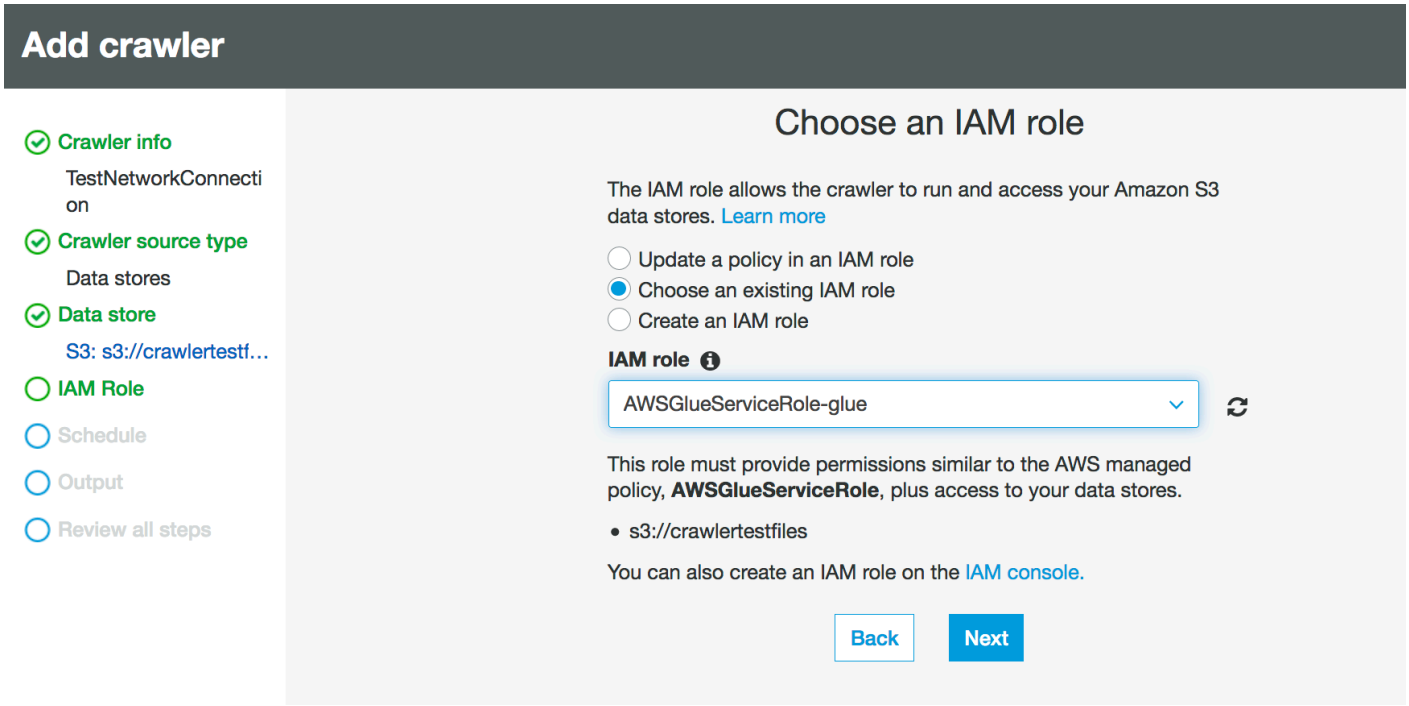
Amazon S3 데이터 스토어의 크롤러 생성

이제 생성한 Network 연결을 지정하는 크롤러를 생성할 수 있습니다. 크롤러 생성에 대한 자세한 내용은 크롤러 구성 섹션을 참조하세요.

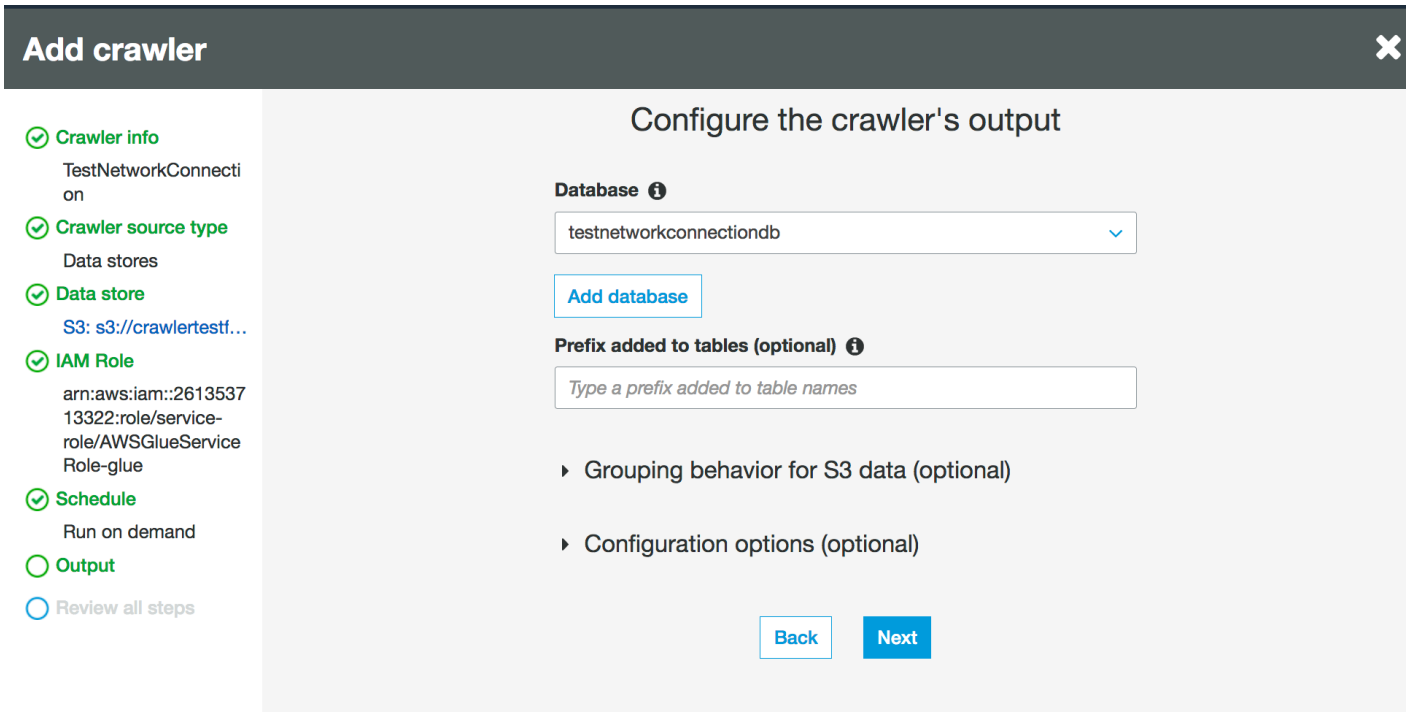
1. 먼저 AWS Glue 콘솔의 탐색 창에서 크롤러를 선택합니다.
2. 크롤러 추가를 선택합니다.
3. 크롤러 이름을 지정하고 [다음(Next)]을 선택합니다.
4. 데이터 원본을 묻는 메시지가 표시되면 [S3]를 선택하고 Amazon S3 버킷 접두사와 이전에 생성한 연결을 지정합니다.



5. 필요한 경우 동일한 네트워크 연결에 다른 데이터 스토어를 추가합니다.
6. IAM 역할을 선택합니다. IAM 역할은 AWS Glue 서비스 및 Amazon S3 버킷에 대한 액세스를 허용해야 합니다. 자세한 내용은 the section called “크롤러 구성” 단원을 참조하십시오.



7. 크롤러의 일정을 정의합니다.
8. 데이터 카탈로그에서 기존 데이터베이스를 선택하거나 새 데이터베이스 항목을 생성합니다.

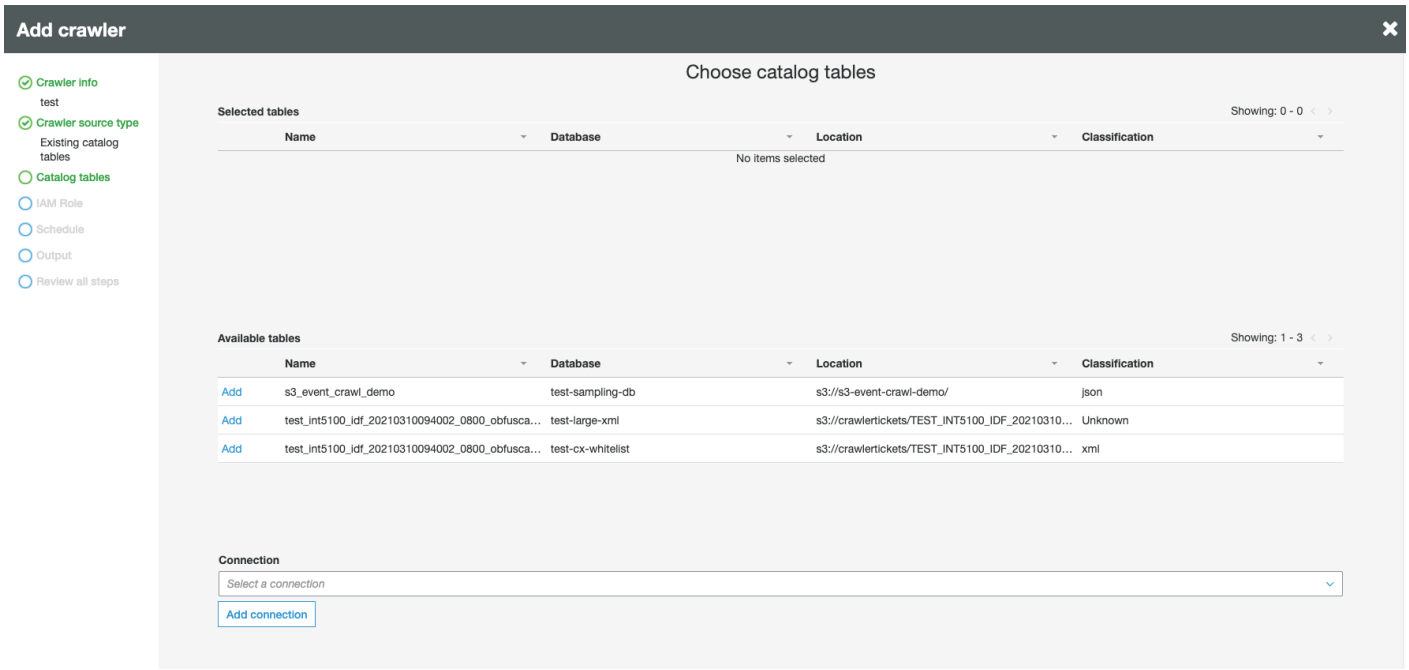


9. 나머지 설정을 완료합니다.

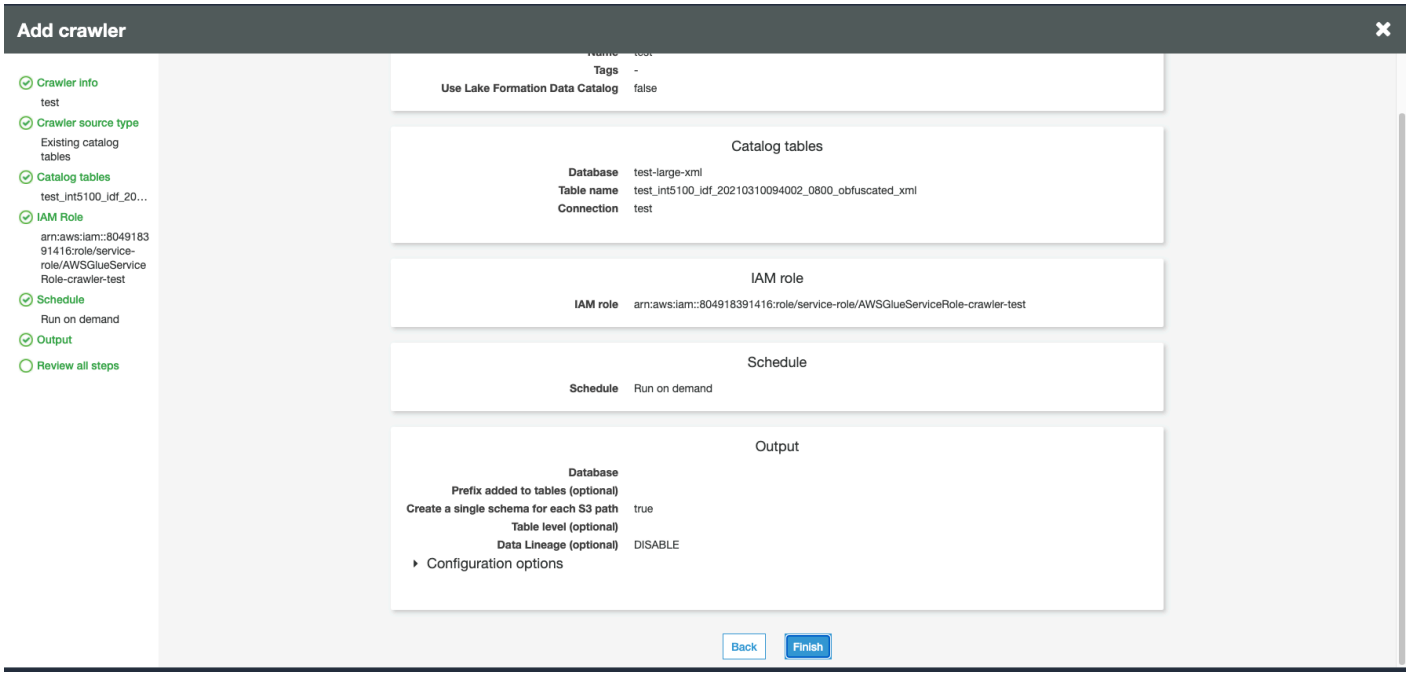
Amazon S3 기반 데이터 카탈로그 테이블의 크롤러 생성

이제 앞에서 생성한 Network 연결과 카탈로그 소스 유형을 지정하는 크롤러를 생성할 수 있습니다. 크롤러 생성에 대한 자세한 내용은 크롤러 구성 섹션을 참조하세요.

1. 먼저 AWS Glue 콘솔의 탐색 창에서 크롤러를 선택합니다.
2. 크롤러 추가를 선택합니다.
3. 크롤러 이름을 지정하고 [다음(Next)]을 선택합니다.
4. 크롤러 소스 유형을 묻는 메시지가 표시되면 Existing catalog tables(기존 카탈로그 테이블)를 선택한 후 Available Tables(사용 가능한 테이블) 목록에서 크롤링할 기존 카탈로그 테이블을 지정합니다.

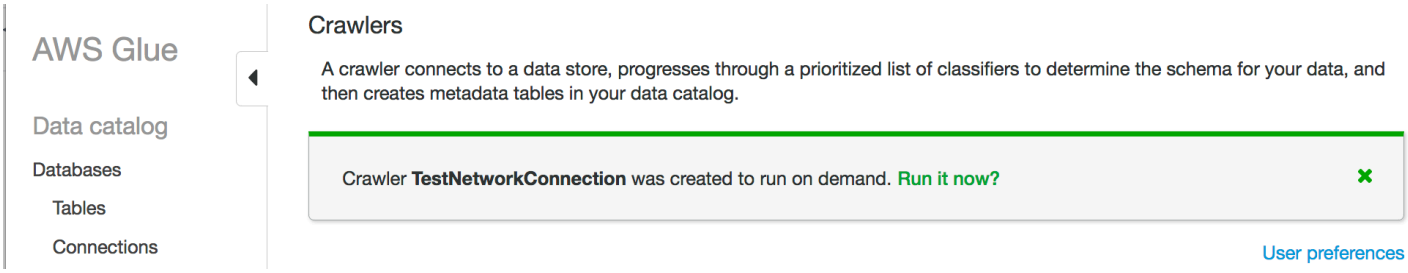


5. IAM 역할을 선택합니다. IAM 역할은 AWS Glue 서비스 및 Amazon S3 버킷에 대한 액세스를 허용해야 합니다. 자세한 내용은 the section called “크롤러 구성” 단원을 참조하십시오.
6. 크롤러의 일정을 정의합니다.
7. 데이터 카탈로그에서 기존 데이터베이스를 선택하거나 새 데이터베이스 항목을 생성합니다.
8. 나머지 설정을 완료하고 단계를 검토합니다.



크롤러 실행

크롤러를 실행합니다.



문제 해결

VPC 게이트웨이를 사용하는 Amazon S3 버킷과 관련된 문제를 해결하려면 게이트웨이 VPC 엔드포인트를 사용하여 S3 버킷에 연결할 수 없는 이유는 무엇입니까?를 참조하세요.

AWS Lake Formation 보호 데이터 사용

전체 테이블 액세스

AWS Glue는 AWS Lake Formation 보호된 테이블에 대해 전체 테이블 액세스 권한(FTA)을 지원합니다. 이렇게 하면 ETL 작업이 전체 테이블 권한으로 데이터를 읽고 쓸 수 있습니다.

사전 조건

- 적절한 IAM 역할 및 권한
- 데이터 카탈로그에 AWS Lake Formation 구성
- 호환되는 테이블 유형(Hive 또는 Iceberg)

주요 고려 사항

필수 권한

- `lakeformation:GetDataAccess` IAM 권한
- AWS Lake Formation 테이블 권한
- Amazon S3 버킷 액세스 권한

지원되는 테이블 유형

- Hive 테이블
- Iceberg 테이블

제한 사항

- Spark Streaming과 호환되지 않음
- 세분화된 액세스 제어와 동시에 사용할 수 없음
- Delta 또는 Hudi 테이블을 지원하지 않음

모범 사례

1. 작업 실행 전에 IAM 역할이 올바르게 구성되었는지 확인
2. 개발 환경에서 액세스 권한 테스트
3. 작업 실행 로그에서 권한 관련 문제 모니터링
4. 액세스 패턴에 대한 명확한 문서화 유지

문제 해결

일반적인 문제:

- IAM 권한 누락
- 잘못된 AWS Lake Formation 구성
- 테이블 유형 호환성 문제

전체 설정 지침 및 구성 세부 정보는 AWS Lake Formation 전체 테이블 액세스 권한 사용을 참조하세요.

AWS Glue 연결 문제 해결

AWS Glue 크롤러 혹은 작업을 데이터 스토어에 연결하기 위해 연결 속성을 사용할 경우, 연결을 시도할 때 오류가 발생할 수 있습니다. 탄력적 네트워크 인터페이스를 지정 Virtual Private Cloud(VPC)와 서브넷에 생성할 경우 AWS Glue는 서브넷의 프라이빗 IP 주소를 사용합니다. 연결에 지정된 보안 그룹은 탄력적 네트워크 인터페이스 각각에 적용됩니다. 보안 그룹이 발신 연결을 허용하는지 데이터베이스 클러스터 연결을 허용하는지 확인합니다.

또한 Apache Spark는 드라이버 및 관리자 노드의 양방향 연결을 요구합니다. 보안 그룹 중 하나는 모든 TCP 포트에서 수신을 허용해야 합니다. 자기 참조 보안 그룹을 통해 보안 그룹의 소스를 자체 내에서 제한하여 외부로 공개되는 것을 예방합니다.

다음은 연결 문제를 해결하기 위해 취해야 할 일반적인 행동입니다.

- 연결의 포트 주소를 확인합니다.
- 연결 또는 보안 암호에서 사용자 이름과 암호 문자열을 확인합니다.
- JDBC 데이터 스토어의 경우, 수신 연결을 허용하는지 확인합니다.
- VPC 내에서 스토어가 연결되는지 확인합니다.
- AWS Secrets Manager를 사용하여 연결 자격 증명을 저장하는 경우 AWS Glue에 대한 IAM 역할에 보안 암호에 액세스할 수 있는 권한이 있는지 확인합니다. 자세한 내용을 알아보려면 AWS Secrets Manager 사용 설명서의 예: 보안 암호 값을 검색할 수 있는 권한을 참조하세요. 네트워크 설정에 따라 VPC와 Secrets Manager 간의 프라이빗 연결을 설정하기 위해 VPC 엔드포인트를 생성해야 할 수도 있습니다. 자세한 내용을 알아보려면 AWS Secrets Manager VPC 엔드포인트 사용을 참조하세요.

자습서: AWS Glue Connector for Elasticsearch 사용

Elasticsearch는 로그 분석, 실시간 애플리케이션 모니터링, 클릭 스트림 분석 같은 사용 사례를 위한 인기 오픈 소스 검색 및 분석 엔진입니다. AWS Glue Studio에서 AWS Glue Connector for

Elasticsearch를 구성하여 OpenSearch를 추출, 변환 및 로드(ETL) 작업의 데이터 저장소로 사용할 수 있습니다. 이 커넥터는 AWS Marketplace에서 무료로 사용할 수 있습니다.

> **Note**
>
> AWS Marketplace Elasticsearch Spark 커넥터는 더 이상 사용되지 않습니다. 대신 AWS Glue Connector for Elasticsearch를 사용하세요.

이 자습서에서는 최소한의 단계로 Amazon OpenSearch Service 노드에 연결하는 방법을 보여줍니다.

주제

- 사전 조건
- 1단계: (선택 사항) OpenSearch 클러스터 정보에 대한 AWS 보안 암호 생성
- 2단계: 커넥터 구독
- 3단계: AWS Glue Studio에서 커넥터 활성화 및 연결 생성
- 4단계: ETL 작업에 대한 IAM 역할 구성
- 5단계: OpenSearch 연결을 사용하는 작업 생성
- 6단계: 작업 실행

사전 조건

이 튜토리얼을 사용하려면 다음이 필요합니다.

- AWS Glue Studio에 액세스
- AWS 클라우드의 OpenSearch 클러스터에 대한 액세스 권한
- (선택 사항) AWS Secrets Manager에 대한 액세스 권한.

1단계: (선택 사항) OpenSearch 클러스터 정보에 대한 AWS 보안 암호 생성

연결 자격 증명을 안전하게 보관하고 사용하려면 AWS Secrets Manager에 자격 증명을 저장합니다. 생성한 보안 암호는 튜토리얼의 뒷부분에서 연결에 사용됩니다. 자격 증명 키-값 페어는 일반 연결 옵션으로 AWS Glue Connector for Elasticsearch에 공급됩니다.

보안 암호 생성에 대한 자세한 내용은 AWS Secrets Manager User Guide의 Creating and Managing Secrets with AWS Secrets Manager를 참조하세요.

AWS 보안 암호를 생성하려면

1. AWS Secrets Manager 콘솔에 로그인합니다.
2. 서비스 소개 페이지 또는 비밀 목록 페이지에서 새 비밀 저장을 선택합니다.
3. 새 보안 암호 저장(Store a new secret) 페이지에서 다른 유형의 보안 암호(Other type of secret)를 선택합니다. 이 옵션은 보안 암호의 구조 및 세부 정보를 제공해야 함을 의미합니다.
4. OpenSearch 클러스터 사용자 이름에 대한 [키(Key)] 및 [값(Value)] 페어를 추가합니다. 예:

   `es.net.http.auth.user`*`username`*
5. [+ 행 추가(+ Add row)]를 선택하고 암호에 대해 다른 키-값 페어를 입력합니다. 예:

   `es.net.http.auth.pass`: *`password`*
6. Next(다음)를 선택합니다.
7. 보안 암호 이름을 입력합니다. 예: my-es-secret. 선택적으로 설명을 포함할 수 있습니다.

   이 튜토리얼의 뒷부분에서 사용되는 보안 암호를 기록하고 [다음(Next)]을 선택합니다.
8. [다음(Next)]을 다시 선택하고 [저장(Store)]을 선택하여 보안 암호를 생성합니다.

다음 단계

2단계: 커넥터 구독

2단계: 커넥터 구독

AWS Glue Connector for Elasticsearch는 AWS Marketplace에서 무료로 제공됩니다.

AWS Marketplace에서 AWS Glue Connector for Elasticsearch를 구독하려면

1. License Manager를 사용하도록 AWS 계정을 아직 구성하지 않은 경우 다음을 수행합니다.
   a. AWS License Manager 콘솔https://console.aws.amazon.com/license-manager을 엽니다.
   b. [고객 관리형 라이선스 생성(Create customer managed license)]을 선택합니다.
   c. IAM permissions(IAM 권한)(1회 설정) 창에서 I grant AWS License Manager the required permissions(필요한 권한 부여)를 선택한 다음 Grant permissions(권한 부여)를 선택합니다.

      이 창이 표시되지 않으면 필요한 권한을 이미 구성한 것입니다.
2. https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.

3. AWS Glue Studio 콘솔에서 메뉴 아이콘 (☰) 을 확장한 다음, 탐색 창에서 커넥터(Connectors)를 선택합니다.
4. Connectors(커넥터) 페이지에서 Go to AWS Marketplace(이동)을 선택합니다.
5. AWS Marketplace에서 Search AWS Glue Studio products(Glue Studio 제품 검색) 섹션의 검색 필드에 AWS Glue Connector for Elasticsearch를 입력하고 Enter 키를 누릅니다.
6. 커넥터 이름인 AWS Glue Connector for Elasticsearch를 선택합니다.
7. 커넥터의 제품 페이지에서 탭을 사용하여 커넥터에 대한 정보를 봅니다. 계속할 준비가 되면 [계속 구독(Continue to Subscribe)]을 선택합니다.
8. 소프트웨어의 사용 약관을 검토합니다. 약관 수락을 클릭하세요.
9. 구독 프로세스가 완료되면 다음과 같은 알림이 표시됩니다. "이 제품을 구독해 주셔서 감사합니다! 이제 소프트웨어를 구성할 수 있습니다." 배너 위에는 구성 계속 버튼이 있습니다. Continue to Configuration(구성 계속)을 선택합니다.
10. 소프트웨어 구성 페이지에서 이행(Fulfillment) 옵션을 선택합니다. AWS Glue 1.0/2.0 또는 AWS Glue 3.0 중에서 선택할 수 있습니다. 그런 뒤 [계속 시작(Continue to Launch)]을 선택합니다.

다음 단계

3단계: AWS Glue Studio에서 커넥터 활성화 및 연결 생성

3단계: AWS Glue Studio에서 커넥터 활성화 및 연결 생성

[계속 시작(Continue to Launch)]을 선택하면 AWS Marketplace에 [이 소프트웨어 시작(Launch this software)] 페이지가 표시됩니다. 링크를 사용하여 AWS Glue Studio에서 커넥터를 활성화한 후 연결을 생성합니다.

AWS Glue Studio에서 커넥터를 배포하고 연결을 생성하려면

1. AWS Marketplace 콘솔의 [이 소프트웨어 시작(Launch this software)] 페이지에서 [사용 지침(Usage Instructions)]을 선택한 다음 나타나는 창에서 링크를 선택합니다.

   브라우저가 AWS Glue Studio 콘솔의 Marketplace 연결 생성(Create Marketplace connection) 페이지로 리디렉션됩니다.
2. 연결의 이름을 입력합니다. 예: my-es-connection.
3. [연결 액세스(Connection access)] 섹션에서 [연결 자격 증명 유형(Connection credential type)]에 대해 [사용자 이름 및 암호(User name and password)]를 선택합니다.

4. AWS secret(보안 암호)에 대해 보안 암호 이름을 입력합니다. 예: my-es-secret.
5. [네트워크 옵션(Network options)] 섹션에서 OpenSearch 클러스터에 연결할 VPC 정보를 입력합니다.
6. [연결 생성 및 커넥터 활성화(Create connection and activate connector)]를 선택합니다.

다음 단계

4단계: ETL 작업에 대한 IAM 역할 구성

4단계: ETL 작업에 대한 IAM 역할 구성

AWS Glue ETL 작업을 생성할 때 사용할 작업에 대한 AWS Identity and Access Management(IAM) 역할을 지정합니다. 역할은 Amazon S3(모든 소스, 대상, 스크립트, 드라이버 파일 및 임시 디렉터리용)와 AWS Glue Data Catalog 객체를 포함하여 작업에서 사용하는 모든 리소스에 대한 액세스 권한을 부여해야 합니다.

AWS Glue ETL 작업에 대해 수임된 IAM 역할은 이전 섹션에서 생성된 보안 암호에도 액세스할 수 있어야 합니다. 기본적으로 AWS 관리형 역할 `AWSGlueServiceRole`은 보안 암호에 액세스할 수 없습니다. 보안 암호에 대한 액세스 제어를 설정하려면 AWS Secrets Manager에 대한 인증 및 액세스 제어와 특정 보안 암호에 대한 액세스 제한을 참조하세요.

ETL 작업에 대한 IAM 역할을 구성하려면

1. the section called “ETL 작업에 필요한 IAM 권한 검토”에 설명된 권한을 구성합니다.
2. the section called “커넥터 사용에 필요한 권한”에 설명된 것처럼, AWS Glue Studio에서 커넥터를 사용할 때 필요한 추가 권한을 구성합니다.

다음 단계

5단계: OpenSearch 연결을 사용하는 작업 생성

5단계: OpenSearch 연결을 사용하는 작업 생성

ETL 작업에 대한 역할을 생성한 후 AWS Glue Studio에서 Open Spark ElasticSearch용 커넥터와 연결을 사용하는 작업을 생성할 수 있습니다.

작업이 Amazon Virtual Private Cloud(Amazon VPC) 내에서 실행되는 경우 VPC가 올바르게 구성되었는지 확인합니다. 자세한 내용은 the section called “ETL 작업에 사용할 VPC 구성” 단원을 참조하십시오.

Elasticsearch Spark 커넥터를 사용하는 작업을 생성하려면

1. AWS Glue Studio에서 커넥터(Connectors)를 선택합니다.
2. [연결(Your connections)] 목록에서 방금 생성한 연결을 선택하고 [작업 생성(Create job)]을 선택합니다.
3. 시각적 작업 편집기에서 데이터 원본 노드를 선택합니다. 오른쪽의 [데이터 원본 속성 - 커넥터(Data source properties - Connector)] 탭에서 커넥터에 대한 추가 정보를 구성합니다.

   a. [스키마 추가(Add schema)]를 선택하고 데이터 원본에 있는 데이터 집합의 스키마를 입력합니다. 연결은 데이터 카탈로그에 저장된 테이블을 사용하지 않습니다. 즉, AWS Glue Studio는 데이터의 스키마를 알지 못합니다. 이 스키마 정보를 수동으로 제공해야 합니다. 스키마 편집기를 사용하는 방법에 대한 지침은 the section called “사용자 정의 변환 노드에서 스키마 편집” 섹션을 참조하세요.

   b. [연결 옵션(Connection options)]을 확장합니다.

   c. [새 옵션 추가(Add new option)]를 선택하고 AWS 보안 암호에 입력되지 않은 커넥터에 필요한 정보를 입력합니다.

      - es.nodes: https://<OpenSearch domain endpoint>
      - es.port: 443
      - path: 테스트
      - es.nodes.wan.only.: true

      이러한 연결 옵션에 대한 설명은 https://www.elastic.co/guide/en/elasticsearch/hadoop/current/configuration.html 섹션을 참조하세요.
4. 그래프에 대상 노드를 추가합니다.

   데이터 대상은 Amazon S3이거나 AWS Glue Data Catalog 또는 커넥터의 정보를 사용하여 다른 위치에 데이터를 쓸 수 있습니다. 예를 들어 데이터 카탈로그 테이블을 사용하여 Amazon RDS의 데이터베이스에 쓰거나 커넥터를 데이터 대상으로 사용하여 AWS Glue에서 기본적으로 지원되지 않는 데이터 스토어에 쓸 수 있습니다.

   데이터 대상에 대한 커넥터를 선택하는 경우 해당 커넥터에 대해 생성된 연결을 선택해야 합니다. 또한 커넥터 공급자가 요구하는 경우 커넥터에 추가 정보를 제공하는 옵션을 추가해야 합니다. AWS 보안 암호에 대한 정보가 포함된 연결을 사용하는 경우 연결 옵션에서 사용자 이름과 암호 인증을 제공할 필요가 없습니다.

5. 필요에 따라 the section called “AWS Glue 관리형 변환으로 데이터 변환”에 설명된 대로 추가 데이터 원본과 하나 이상의 변환 노드를 추가합니다.
6. the section called “작업 속성 수정”에 설명된 대로 3단계부터 작업 속성을 구성하고 작업을 저장합니다.

다음 단계

6단계: 작업 실행

6단계: 작업 실행

작업을 저장한 후 작업을 실행하여 ETL 작업을 수행할 수 있습니다.

AWS Glue Connector for Elasticsearch에 대해 생성한 작업을 실행하려면

1. AWS Glue Studio 콘솔을 사용해 시각적 편집기 페이지에서 실행(Run)을 선택합니다.
2. 성공 배너에서 [실행 세부 정보(Run Details)]를 선택하거나 시각적 편집기의 [실행(Runs)] 탭을 선택하여 작업 실행에 대한 정보를 볼 수 있습니다.

대화형 세션을 사용하여 AWS Glue 작업 구축

데이터 엔지니어는 AWS Glue의 대화형 세션을 사용하기 전보다 더 쉽고 빠르게 AWS Glue 작업을 작성할 수 있습니다.

주제

- AWS Glue 대화형 세션 개요
- AWS Glue 대화형 세션 시작하기
- Jupyter 및 AWS Glue Studio 노트북용 AWS Glue 대화형 세션 구성
- 스크립트 또는 노트북을 AWS Glue 작업으로 변환
- AWS Glue 대화형 세션에서 스트리밍 작업 수행
- 로컬로 AWS Glue 작업 스크립트 개발 및 테스트
- 개발 엔드포인트

AWS Glue 대화형 세션 개요

AWS Glue 대화형 세션을 통해 데이터 준비 및 분석 애플리케이션을 빠르게 구축, 테스트 및 실행할 수 있습니다. 대화형 세션은 데이터 준비를 위한 추출, 변환, 적재(ETL) 스크립트를 구축하고 테스트하기 위한 프로그래밍 방식의 시각적 인터페이스를 제공합니다. 대화형 세션은 Apache Spark 분석 애플리케이션을 실행하고 원격 Spark 런타임 환경에 대한 온디맨드 액세스를 제공합니다. AWS Glue는 이러한 대화형 세션에 대해 서버리스 Spark를 투명하게 관리합니다.

대화형 세션은 유연하기 때문에 선택한 환경에서 애플리케이션을 구축하고 테스트할 수 있습니다. AWS Command Line Interface 및 API를 통해 대화형 세션을 만들고 작업할 수 있습니다. Jupyter 호환 노트북을 사용하여 노트북 스크립트를 시각적으로 작성하고 테스트할 수 있습니다. 대화형 세션은 PyCharm, IntelliJ 및 VS Code와 같은 IDE와의 통합을 포함하여 Jupyter가 수행하는 거의 모든 곳에서 통합되는 오픈 소스 Jupyter 커널을 제공합니다. 이러한 특징 덕분에 로컬 환경에서 코드를 작성하고 대화형 세션 백엔드에서 원활하게 실행할 수 있습니다.

대화형 세션 API를 사용하여 고객은 Spark 인프라를 관리할 필요 없이 Apache Spark 분석을 사용하는 애플리케이션을 프로그래밍 방식으로 실행할 수 있습니다. 단일 대화형 세션 내에서 하나 이상의 Spark 문을 실행할 수 있습니다.

따라서 대화형 세션은 데이터 준비 및 분석 애플리케이션을 구축하고 실행하는 더 빠르고 경제적이며 유연한 방법을 제공합니다. 대화형 세션 사용 방법에 대한 자세한 내용은 이 섹션의 설명서를 참조하세요. AWS Glue에서 지원하는 매직

제한 사항

- 대화형 세션에서는 작업 북마크가 지원되지 않습니다.
- AWS Command Line Interface를 사용한 노트북 작업 생성은 지원되지 않습니다.
- AWS Glue Studio 노트북은 Scala를 지원하지 않습니다.

AWS Glue 대화형 세션 시작하기

이 섹션에서는 로컬에서 AWS Glue 대화형 세션을 실행하는 방법을 설명합니다.

대화형 세션을 로컬에서 설정하기 위한 사전 조건

다음은 대화형 세션을 설치하기 위한 사전 조건입니다.

- 지원되는 Python 버전은 3.6부터 3.10 이상입니다.
- MacOS/리눅스 및 Windows 지침은 아래 섹션을 참조하세요.

Jupyter 및 AWS Glue 대화형 세션 Jupyter 커널 설치

다음을 사용하여 커널을 로컬에 설치합니다.

`install-glue-kernels` 명령은 pyspark 커널과 spark 커널 모두에 대한 jupyter kernelspec을 설치하고 올바른 디렉터리에 로고도 설치합니다.

```
pip3 install --upgrade jupyter boto3 aws-glue-sessions
```

```
install-glue-kernels
```

Jupyter 실행

Jupyter Notebook을 실행하려면 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 Jupyter Notebook을 시작합니다.

   ```
   jupyter notebook
   ```

2. 새로 생성(New)을 선택한 다음 AWS Glue 커널 중 하나를 선택하여 AWS Glue에 대한 코딩을 시작합니다.

세션 자격 증명 및 리전 구성

MacOS/Linux 지침

AWS Glue 대화형 세션에는 AWS Glue 작업 및 개발 엔드포인트와 동일한 IAM 권한이 필요합니다. 다음 두 가지 방법 중 하나로 대화형 세션에 사용되는 역할을 지정합니다.

1. `%iam_role` 및 `%region` 매직 사용
2. `~/.aws/config`에 추가 줄 사용

매직을 사용하여 세션 역할 구성

첫 번째 셀에서, 실행된 첫 번째 셀에 `%iam_role <YourGlueServiceRole>`을 입력합니다.

`~/.aws/config`를 사용하여 세션 역할 구성

대화형 세션의 AWS Glue 서비스 역할은 노트북 자체에 지정되거나 AWS CLI 구성과 함께 저장될 수 있습니다. 일반적으로 AWS Glue 작업에서 사용하는 역할이 있다면 바로 그 역할일 것입니다. AWS Glue 작업에 사용하는 역할이 없는 경우 이 안내서의 AWS Glue에 대한 IAM 권한 구성에 따라 구성합니다.

이 역할을 대화형 세션의 기본 역할로 설정하려면 다음을 수행하세요.

1. 텍스트 편집기로 `~/.aws/config`를 엽니다.
2. AWS Glue에서 사용하는 프로파일을 찾습니다. 프로파일을 사용하지 않는 경우 `[Default]` 프로파일을 사용합니다.
3. 프로파일에서, 사용하려는 역할에 `glue_role_arn=<AWSGlueServiceRole>`과 같은 줄을 추가합니다.
4. [선택 사항]: 프로파일에 기본 리전 세트가 없는 경우, `region=us-east-1`을 추가하고 원하는 리전으로 `us-east-1`을 대체하는 것이 좋습니다.
5. 구성을 저장합니다.

자세한 내용은 IAM을 이용한 대화형 세션 단원을 참조하십시오.

Windows 지침

AWS Glue 대화형 세션에는 AWS Glue 작업 및 개발 엔드포인트와 동일한 IAM 권한이 필요합니다. 다음 두 가지 방법 중 하나로 대화형 세션에 사용되는 역할을 지정합니다.

1. %iam_role 및 %region 매직 사용
2. ~/.aws/config에 추가 줄 사용

매직을 사용하여 세션 역할 구성

첫 번째 셀에서, 실행된 첫 번째 셀에 %iam_role <YourGlueServiceRole>을 입력합니다.

~/.aws/config를 사용하여 세션 구성

대화형 세션의 AWS Glue 서비스 역할은 노트북 자체에 지정되거나 AWS CLI 구성과 함께 저장될 수 있습니다. 일반적으로 AWS Glue 작업에서 사용하는 역할이 있다면 바로 그 역할일 것입니다. AWS Glue 작업에 사용하는 역할이 없는 경우, AWS Glue에 대한 IAM 권한 설정 가이드를 따라 역할을 설정하세요.

이 역할을 대화형 세션의 기본 역할로 설정하려면 다음을 수행하세요.

1. 텍스트 편집기로 ~/.aws/config를 엽니다.
2. AWS Glue에서 사용하는 프로파일을 찾습니다. 프로파일을 사용하지 않는 경우 [Default] 프로파일을 사용합니다.
3. 프로파일에서, 사용하려는 역할에 glue_role_arn=<AWSGlueServiceRole>과 같은 줄을 추가합니다.
4. [선택 사항]: 프로파일에 기본 리전 세트가 없는 경우, region=us-east-1을 추가하고 원하는 리전으로 us-east-1을 대체하는 것이 좋습니다.
5. 구성을 저장합니다.

자세한 내용은 IAM을 이용한 대화형 세션 단원을 참조하십시오.

대화형 세션 평가판 업그레이드

커널은 버전 0.27과 함께 릴리스되었을 때 새 이름으로 업그레이드되었습니다. 커널의 평가판 버전을 정리하려면 터미널 또는 PowerShell에서 다음을 실행합니다.

> **Note**
>
> 사용자 지정 서비스 모델이 필요한 다른 AWS Glue 평가판의 일부인 경우 커널을 제거하면 사용자 지정 서비스 모델이 제거됩니다.

```
# Remove Old Glue Kernels
jupyter kernelspec remove glue_python_kernel
jupyter kernelspec remove glue_scala_kernel

# Remove Custom Model
cd ~/.aws/models
rm -rf glue/
```

SageMaker AI Studio에서 대화형 세션 사용

AWS Glue 대화형 세션은 데이터 과학자와 엔지니어가 데이터 준비 및 분석 애플리케이션을 빠르게 구축, 테스트 및 실행하는 데 사용할 수 있는 온디맨드 서버리스 Apache Spark 런타임 환경입니다. Amazon SageMaker AI Studio Classic 노트북을 시작하여 AWS Glue 대화형 세션을 시작할 수 있습니다.

자세한 내용은 AWS Glue 대화형 세션을 사용하여 데이터 준비를 참조하세요.

Microsoft Visual Studio Code에서 대화형 세션 사용

사전 조건

- AWS Glue 대화형 세션을 설치하고 Jupyter Notebook에서 작동하는지 확인합니다.
- Visual Studio Code with Jupyter를 다운로드하고 설치합니다. 자세한 내용은 VS Code의 Jupyter Notebook을 참조하세요.

VSCode를 사용하여 대화형 세션을 시작하려면

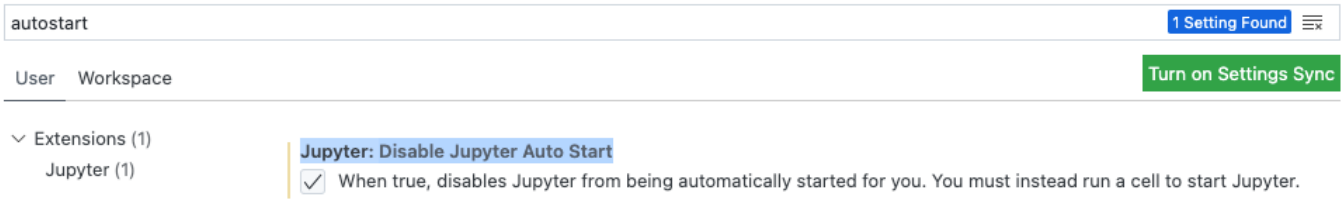
1. VSCode에서 Jupyter 자동 시작을 비활성화합니다.

   Visual Studio Code에서 Jupyter 커널이 자동으로 시작되어 세션이 이미 시작될 때 매직이 적용되지 않습니다. Windows에서 자동 시작을 비활성화하려면 파일 > 기본 설정 > 확장 프로그램 > Jupyter로 이동하고 Jupyter에서 마우스 오른쪽 버튼을 클릭한 다음 확장 프로그램 설정을 선택합니다.

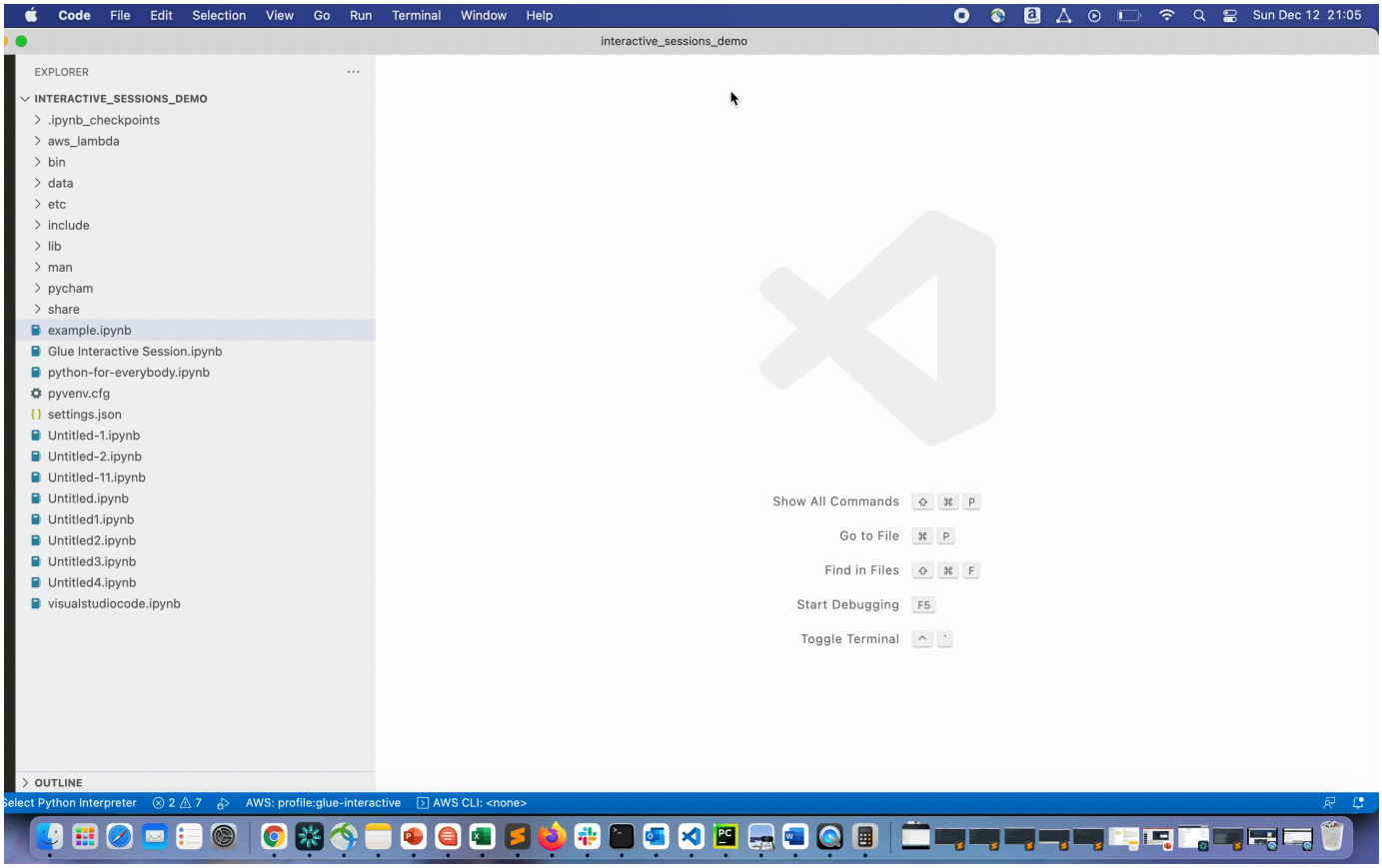
   MacOS의 경우 코드 > 설정 > 확장 프로그램 > Jupyter로 이동하고 Jupyter에서 마우스 오른쪽 버튼을 클릭한 다음 확장 프로그램 설정을 선택합니다.

   Jupyter: Jupyter 자동 시작 비활성화가 보일 때까지 아래로 스크롤합니다. 'True이면 Jupyter가 자동으로 시작되지 않도록 설정합니다. 대신 Jupyter를 시작하려면 셀을 실행해야 합니다."("When

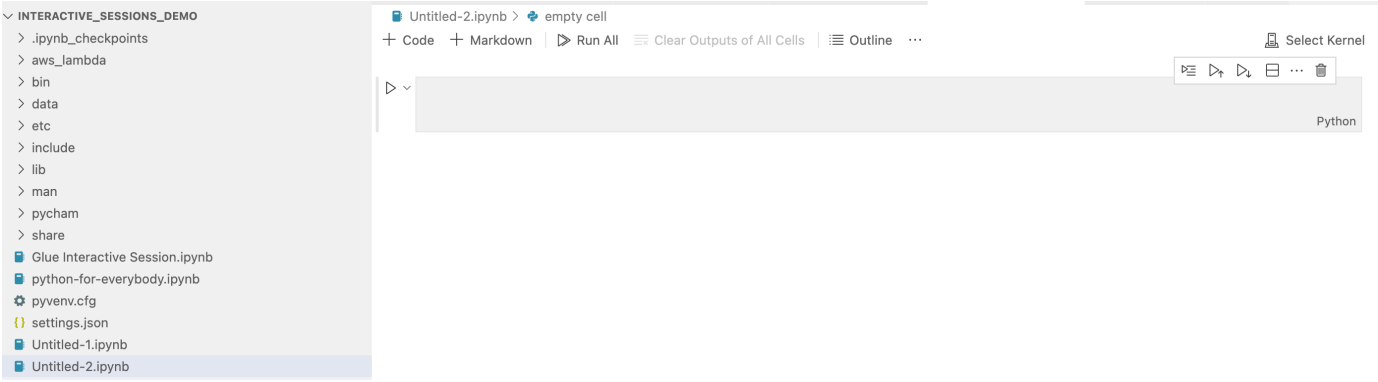
true, disables Jupyter from being automatically started for you. You must instead run a cell to start Jupyter.")라는 레이블이 지정된 확인란을 선택합니다.



2. 파일(File) > 새 파일(New File) > 저장(Save)으로 이동하여 원하는 이름의 `.ipynb` 확장자로 저장하거나 언어 선택(Select a language)에서 Jupyter를 선택하여 파일을 저장합니다.



3. 파일을 두 번 클릭합니다. Jupyter 셀이 표시되고 노트북이 열립니다.



4. Windows에서 파일을 처음 만들 때 기본적으로 선택되어 있는 커널은 없습니다. 커널 선택(Select Kernel)을 클릭하면 사용 가능한 커널 목록이 표시됩니다. Glue PySpark를 선택합니다.

   MacOS에서 Glue PySpark 커널이 보이지 않으면 다음 단계를 시도하세요.

   1. 로컬 Jupyter 세션을 실행하여 URL을 확보합니다.

      예를 들어 다음 명령을 실행하여 Jupyter Notebook을 시작합니다.

      ```
      jupyter notebook
      ```

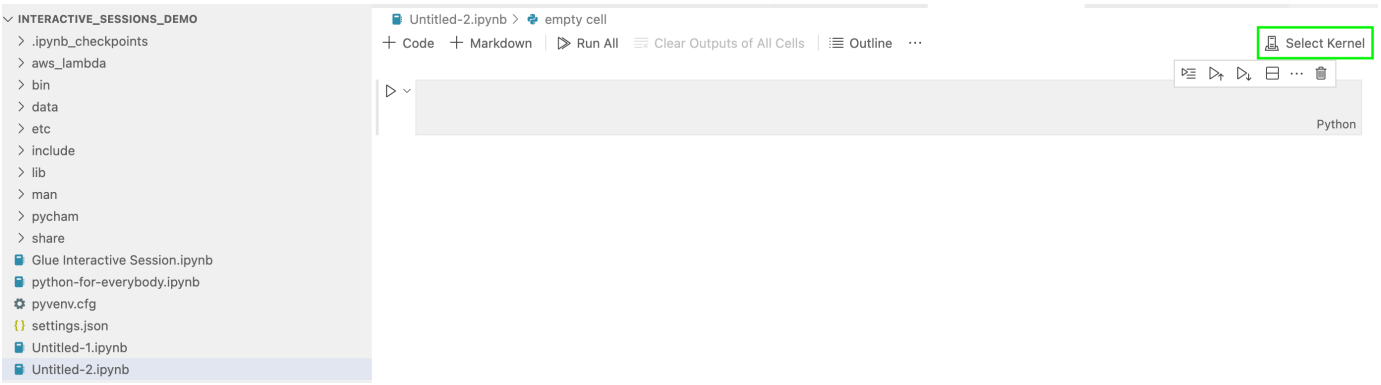
      노트북을 처음 실행하면 `http://localhost:8888/?token=3398XXXXXXXXXXXXXXX`와 같은 URL이 표시됩니다.

      URL을 복사합니다.

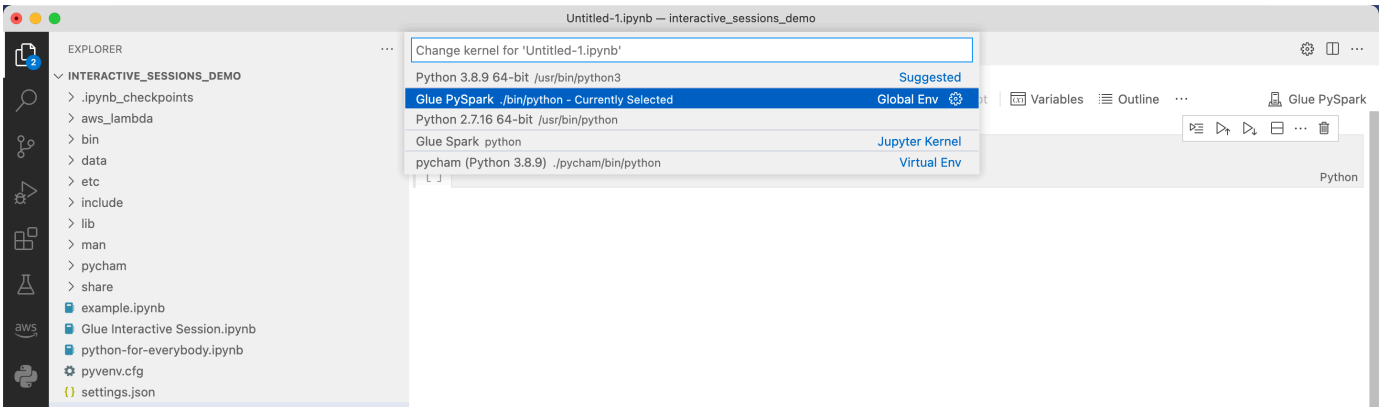
   2. VS Code에서 현재 커널을 클릭한 다음 다른 커널 선택..., 기존 Jupyter 서버...를 차례로 선택합니다. 위 단계에서 복사한 URL을 붙여넣습니다.

      오류 메시지가 표시된다면, VS Code Jupyter wiki를 참조하세요.

   3. 성공하면 커널이 Glue PySpark로 설정됩니다.



Glue PySpark 또는 Glue Spark 커널(각각 Python 및 Scala용)을 선택합니다.



드롭다운 목록에 AWS Glue PySpark 및 AWS Glue Spark 커널이 없다면 전 단계에서 AWS Glue 커널을 설치했는지 또는 Visual Studio Code에서 `python.defaultInterpreterPath` 설정이 올바른지 확인하세요. 자세한 내용은 python.defaultInterpreterPath 설정 설명을 참조하세요.

5. AWS Glue 대화형 세션을 생성합니다. Jupyter Notebook에서와 동일한 방식으로 세션을 생성합니다. 첫 번째 셀 위에 매직을 지정하고 코드 문을 실행합니다.

IAM을 이용한 대화형 세션

이 섹션에서는 AWS Glue 대화형 세션의 보안 고려 사항에 대해 설명합니다.

주제

- 대화형 세션에 사용되는 IAM 보안 주체
- 클라이언트 보안 주체 설정
- 런타임 역할 설정
- TagOnCreate로 세션 비공개 설정

- IAM 정책 고려 사항

대화형 세션에 사용되는 IAM 보안 주체

AWS Glue 대화형 세션에 사용되는 두 가지 IAM 보안 주체를 사용합니다.

- 클라이언트 보안 주체: 클라이언트 보안 주체(사용자 또는 역할)는 보안 주체의 ID 기반 자격 증명으로 구성된 AWS Glue 클라이언트의 대화형 세션에 대한 API 작업에 권한을 부여합니다. 예를 들어. 이는 일반적으로 AWS Glue Console에 액세스하는 데 사용하는 IAM 역할일 수도 있습니다. 보안 인증 정보가 AWS Command Line Interface에 사용되는 IAM의 사용자 또는 대화형 세션 Jupyter 커널에 사용되는 AWS Glue 클라이언트에 할당되는 역할일 수도 있습니다.
- 런타임 역할: 런타임 역할은 클라이언트 보안 주체가 대화형 세션 API 작업에 전달하는 IAM 역할입니다. AWS Glue는 이 역할을 사용하여 세션에서 문을 실행합니다. 예를 들어, 이 역할은 AWS Glue ETL 작업 실행에 사용되는 역할일 수 있습니다.

  자세한 내용은 런타임 역할 설정 단원을 참조하십시오.

클라이언트 보안 주체 설정

대화형 세션 API를 호출할 수 있도록 하려면 클라이언트 보안 주체에 자격 증명 정책을 연결해야 합니다. 이 역할에는 `CreateSession`과 같은 대화형 세션 API로 전달할 실행 역할에 대한 `iam:PassRole` 액세스 권한이 있어야 합니다. 예를 들어, 정책이 연결된 계정의 사용자가 계정에 생성된 모든 세션(예: 런타임 문 또는 취소 문)에 액세스할 수 있도록 허용하는 AWSGlueConsoleFullAccess 관리형 정책을 IAM 역할에 연결할 수 있습니다.

세션을 보호하고 세션을 생성한 사용자와 연결된 특정 IAM 역할에게만 공개하도록 설정하려면 AWS Glue 대화형 세션의 태그 기반 권한 부여 제어인 TagOnCreate를 사용할 수 있습니다. 소유자 태그 기반 범위 축소 관리형 정책이 TagonCreate를 사용하여 세션을 비공개로 만드는 방법에 대한 자세한 내용은 TagOnCreate로 세션 비공개 설정 섹션을 참조하세요. ID 기반 정책에 대한 자세한 내용은 AWS Glue에 대한 ID 기반 정책을 참조하세요.

런타임 역할 설정

AWS Glue가 대화형 세션에서 문을 가정하고 실행할 수 있도록 하려면 CreateSession API 작업에 IAM 역할을 전달해야 합니다. 역할에는 일반적인 AWS Glue 작업을 실행하는 데 필요한 IAM 권한과 동일한 IAM 권한이 있어야 합니다. 예를 들어 AWS Glue가 사용자를 대신하여 AWS 서비스를 호출하도록 허용하는 AWSGlueServiceRole 정책을 사용하여 서비스 역할을 생성할 수 있습니다. AWS Glue 콘솔

을 사용하는 경우 자동으로 사용자를 대신하여 서비스 역할을 생성하거나 기존 역할을 사용합니다. 고유한 IAM 역할을 생성하고 고유한 IAM 정책을 연결하여 유사한 권한을 허용할 수도 있습니다.

세션을 보호하고 세션을 생성한 사용자에게만 공개하도록 설정하려면 AWS Glue 대화형 세션의 태그 기반 권한 부여 제어인 TagOnCreate를 사용할 수 있습니다. 소유자 태그 기반 범위 축소 관리형 정책이 TagonCreate를 사용하여 세션을 비공개로 만드는 방법에 대한 자세한 내용은 TagOnCreate로 세션 비공개 설정 섹션을 참조하세요. 자격 증명 기반 정책에 대한 자세한 내용은 AWS Glue에 대한 자격 증명 기반 정책 섹션을 참조하세요. IAM 콘솔에서 직접 실행 역할을 생성하고 TagOnCreate 기능을 사용하여 서비스를 비공개로 설정하려면 아래 단계를 따르세요.

1. 역할 유형이 `Glue`로 설정된 IAM 역할을 생성합니다.
2. AwsGlueSessionUserRestrictedServiceRole이라는 AWS Glue 관리형 정책을 연결합니다.
3. 역할 이름 앞에 정책 이름 AwsGlueSessionUserRestrictedServiceRole을 붙입니다. 예를 들어, AwsGlueSessionUserRestrictedServiceRole-myrole이라는 이름을 가진 역할을 생성해서 AWS Glue 관리형 정책 AwsGlueSessionUserRestrictedServiceRole을 연결할 수 있습니다.
4. AWS Glue이(가) 역할을 수임할 수 있도록 다음과 같이 신뢰 정책을 연결합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
      ]
    }
  ]
}
```

대화형 세션 Jupyter 커널의 경우 AWS Command Line Interface 프로파일에 `iam_role` 키를 지정할 수 있습니다. 자세한 내용은 ~/.aws/config를 사용하여 세션 구성을 참조하세요. AWS Glue 노트북을 사용하여 대화형 세션과 상호 작용하는 경우 실행하는 첫 번째 셀에서 `%iam_role` 매직의 실행 역할을 전달할 수 있습니다.

TagOnCreate로 세션 비공개 설정

AWS Glue 대화형 세션은 대화형 세션에 대한 태그 지정 및 TBAC(태그 기반 권한 부여)를 명명된 리소스로 지원합니다. TagResource 및 UntagResource API를 사용하는 TBAC 외에도, AWS Glue 대화형 세션은 CreateSession 작업을 통해 세션을 생성하는 동안에만 지정된 태그가 있는 세션을 '태그 지정'하는 TagOnCreate 기능을 지원합니다. 이는 또한 이러한 태그가 DeleteSession에서 제거됨을 의미합니다(일명 UntagOnDelete).

TagOnCreate는 세션 생성자가 세션을 비공개로 만들 수 있는 강력한 보안 메커니즘을 제공합니다. 예를 들어, 호출자의 userId와 일치하는 값을 가진 'owner' 태그가 CreateSession 요청에서 userId 태그로서 제공되는 경우에만 세션 생성을 허용하려면 'owner' RequestTag 및 ${aws:userId} 값이 있는 IAM 정책을 클라이언트 보안 주체(예: 사용자)에 연결할 수 있습니다. 이 정책은 AWS Glue 대화형 세션이 세션 리소스를 생성하도록 하고 세션 생성 시간 동안에만 userId 태그가 있는 세션에 태그를 지정하도록 허용합니다. 또한 CreateSession 중에 전달한 실행 역할에 'owner' ResourceTag가 있는 IAM 정책을 연결하여 세션 생성자(${aws:userId} 값을 가진 소유자 태그)들 대상으로만 세션에 대한 액세스 범위를 좁힐 수 있습니다.

TagOnCreate 기능을 사용하여 세션을 더 쉽게 비공개로 만들기 위해, AWS Glue에서는 특화된 관리형 정책 및 서비스 역할을 제공합니다.

IAM AssumeRole 보안 주체(즉, IAM 역할을 수임하여 발급받은 자격 증명 사용)를 사용하여 AWS Glue 대화형 세션을 생성하고 이 세션을 생성자 외에는 비공개로 설정하려면 AWSGlueSessionUserRestrictedNotebookPolicy 및 AWSGlueSessionUserRestrictedNotebookServiceRole과 유사한 정책을 각각 사용해야 합니다. 이 정책은 AWS Glue가 ${aws:PrincipalTag}를 사용하여 소유자 태그 값을 추출하도록 허용합니다. 이렇게 하려면 수임 역할 자격 증명에서 ${aws:userId} 값을 가진 userId 태그를 SessionTag로 전달해야 합니다. ID 세션 태그를 참조하세요. 자격 증명을 발급하는 인스턴스 프로파일이 있는 Amazon EC2 인스턴스를 사용하고 있고 Amazon EC2 인스턴스 내에서 세션을 생성하거나 세션과 상호 작용하려면, 수임 역할 자격 증명에서 ${aws:userId} 값을 가진 userId 태그를 SessionTag로 전달해야 합니다.

예를 들어, IAM AssumeRole 보안 주체 자격 증명을 사용하여 세션을 생성하고, TagOnCreate 기능을 사용하여 서비스를 비공개로 설정하려는 경우 아래 단계를 따르세요.

1. IAM 콘솔에서 직접 런타임 역할을 생성합니다. 이 AWS Glue 관리형 정책 AwsGlueSessionUserRestrictedNotebookServiceRole을 연결하고 역할 이름 앞에 정책 이름 AwsGlueSessionUserRestrictedNotebookServiceRole을 붙이세요. 예를 들어 AwsGlueSessionUserRestrictedNotebookServiceRole-myrole이라는 이름을 가진 역할을 생성해서 AWS Glue 관리형 정책 AwsGlueSessionUserRestrictedNotebookServiceRole을 연결할 수 있습니다.

2. AWS Glue가 위의 역할을 수임할 수 있도록 허용하려면 아래와 같이 신뢰 정책을 연결하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
       ]
    }
   ]
}
```

3. 이름에 접두사 AwsGlueSessionUserRestrictedNotebookPolicy가 붙은 다른 역할을 생성하고 AWS Glue 관리형 정책 AwsGlueSessionUserRestrictedNotebookPolicy를 연결하여 세션을 비공개로 만듭니다. 관리형 정책 외에도 다음 인라인 정책을 연결하여 iam:PassRole을 1단계에서 생성한 역할에 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/
AwsGlueSessionUserRestrictedNotebookServiceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
```

```
            }
        }
    ]
}
```

4. 다음과 같이 위의 IAM AWS Glue에 신뢰 정책을 연결하여 역할을 수임합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": [
                "glue.amazonaws.com"
            ]
        },
        "Action": [
            "sts:AssumeRole",
            "sts:TagSession"
        ]
    }]
}
```

> **Note**
>
> 선택적으로 단일 역할(예: 노트북 역할)을 사용하고 위의 관리형 정책 AwsGlueSessionUserRestrictedNotebookServiceRole 및 AwsGlueSessionUserRestrictedNotebookPolicy를 모두 연결할 수 있습니다. 또한 추가 인라인 정책을 연결하여 `iam:passrole`을 AWS Glue에 허용하세요. 마지막으로 위의 신뢰 정책을 연결하여 `sts:AssumeRole` 및 `sts:TagSession`을 허용하세요.

AWSGlueSessionUserRestrictedNotebookPolicy

AWSGlueSessionUserRestrictedNotebookPolicy는 태그 키 'owner' 및 보안 주체(사용자 또는 역할)의 AWS 사용자 ID와 일치하는 값이 있는 경우에만 노트북에서 AWS Glue 대화형 세션을 생성할 수 있는 액세스 권한을 제공합니다. 자세한 내용은 정책 변수를 사용할 수 있는 경우를 참조하세요. 이 정책은 AWS Glue Studio에서 AWS Glue 대화형 세션 노트북을 생성하는 보안 주체(사용자 또는 역할)에 연결됩니다. 또한 이 정책은 보안 주체의 AWS 사용자 ID와 일치하는 'owner' 태그 값으로 생성된 AWS

Glue Studio 대화형 세션 리소스와 상호 작용할 수 있는 충분한 액세스 권한을 AWS Glue Studio 노트북에 허용합니다. 이 정책은 세션이 생성된 후 AWS Glue 세션 리소스에서 'owner' 태그를 변경하거나 제거할 수 있는 권한을 거부합니다.

AWSGlueSessionUserRestrictedNotebookServiceRole

AWSGlueSessionUserRestrictedNotebookServiceRole은 노트북을 생성하는 보안 주체(사용자 또는 역할)의 AWS 사용자 ID와 일치하는 'owner' 태그 값으로 생성된 AWS Glue 대화형 세션 리소스와 상호 작용할 수 있는 충분한 액세스 권한을 AWS Glue Studio 노트북에 제공합니다. 자세한 내용은 정책 변수를 사용할 수 있는 경우를 참조하세요. 이 서비스-역할 정책은 노트북에 매직으로 전달되었거나 CreateSession API에 실행 역할로 전달된 역할에 연결됩니다. 또한 이 정책은 태그 키 'owner' 및 보안 주체의 AWS 사용자 ID와 일치하는 값이 제공된 경우에만 노트북에서 AWS Glue 대화형 세션을 생성할 수 있도록 허용합니다. 이 정책은 세션이 생성된 후 AWS Glue 세션 리소스에서 'owner' 태그를 변경하거나 제거할 수 있는 권한을 거부합니다. 이 정책에는 Amazon S3 버킷에서 읽고 쓰기, CloudWatch 로그 쓰기, AWS Glue에서 사용되는 Amazon EC2 리소스에 대한 태그 생성 및 삭제 권한도 포함됩니다.

사용자 정책에서 세션을 프라이빗으로 설정

AWSGlueSessionUserRestrictedPolicy를 계정의 각 사용자에 연결된 IAM 역할에 연결하여 자신의 ${aws:userId}와 일치하는 값을 가진 소유자 태그로만 세션을 생성하도록 제한할 수 있습니다. AWSGlueSessionUserRestrictedNotebookPolicy 및 AWSGlueSessionUserRestrictedNotebookServiceRole을 사용하는 대신 각각 AWSGlueSessionUserRestrictedPolicy 및 AWSGlueSessionUserRestrictedServiceRole과 유사한 정책을 사용해야 합니다. 자세한 내용은 자격 증명 기반 정책 사용을 참조하세요. 이 정책은 자체 ${aws:userId}를 포함한 소유자 태그로 세션을 생성한 사용자의 ${aws:userId}인 생성자에게만 세션에 대한 액세스 범위를 좁힙니다. 런타임 역할 설정의 단계에 따라 IAM 콘솔을 사용하여 직접 실행 역할을 생성한 경우, AwsGlueSessionUserRestrictedPolicy 관리형 정책 연결 외에도 계정의 각 IAM 사용자에게 다음과 같은 인라인 정책을 연결하여 이전에 생성한 실행 역할에 대해 `iam:PassRole`을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": [
```

```
            "arn:aws:iam::*:role/AwsGlueSessionUserRestrictedServiceRole*"
        ],
        "Condition": {
            "StringLike": {
                "iam:PassedToService": [
                    "glue.amazonaws.com"
                ]
            }
        }
    }]
}
```

AWSGlueSessionUserRestrictedPolicy

AWSGlueSessionUserRestrictedPolicy는 태그 키 'owner' 및 AWS 사용자 ID와 일치하는 값이 제공된 경우에만 CreateSession API를 사용하여 AWS Glue 대화형 세션을 생성할 수 있는 액세스를 제공합니다. 이 자격 증명 정책은 CreateSession API를 호출하는 사용자에게 연결됩니다. 또한 이 정책은 'owner' 태그 및 AWS 사용자 ID와 일치하는 값으로 생성된 AWS Glue 대화형 세션 리소스와 상호 작용하도록 허용합니다. 이 정책은 세션이 생성된 후 AWS Glue 세션 리소스에서 'owner' 태그를 변경하거나 제거할 수 있는 권한을 거부합니다.

AWSGlueSessionUserRestrictedServiceRole

AWSGlueSessionUserRestrictedServiceRole은 세션을 제외한 모든 AWS Glue 리소스에 대한 전체 액세스 권한을 제공하고 사용자가 자신과 연결된 대화형 세션만 생성하고 사용할 수 있도록 허용합니다. 이 정책에는 AWS Glue에서 다른 AWS 서비스의 Glue 리소스를 관리하는 데 필요한 기타 권한도 포함됩니다. 또한 이 정책은 다른 AWS 서비스의 AWS Glue 리소스에 태그를 추가할 수 있도록 허용합니다.

IAM 정책 고려 사항

대화형 세션은 AWS Glue의 IAM 리소스입니다. IAM 리소스이기 때문에 세션에 대한 액세스 및 상호 작용은 IAM 정책에 의해 관리됩니다. 클라이언트 보안 주체 또는 관리자가 구성한 실행 역할에 연결된 IAM 정책을 기반으로 클라이언트 보안 주체(사용자 또는 역할)는 새 세션을 생성하고 자신의 세션 및 다른 세션과 상호 작용할 수 있습니다.

관리자가 해당 계정의 모든 AWS Glue 리소스에 대한 액세스를 허용하는 AWSGlueConsoleFullAccess 또는 AWSGlueServiceRole과 같은 IAM 정책을 연결한 경우 클라이언트 보안 주체는 서로 협업할 수 있습니다. 예를 들어 정책에서 허용하는 경우 한 사용자는 다른 사용자가 생성한 세션과 상호 작용할 수 있습니다.

특정 요구 사항에 맞게 정책을 구성하려면 정책의 리소스 구성에 대한 IAM 설명서를 참조하세요. 예를 들어, 사용자에게 속한 세션을 분리하려면 AWS Glue 대화형 세션에서 지원하는 TagonCreate 기능을 사용할 수 있습니다. TagOnCreate로 세션 비공개 설정 섹션을 참조하세요.

대화형 세션에서는 특정 VPC 조건에 따라 세션 생성을 제한할 수 있습니다. 조건 키를 사용하여 설정을 제어하는 정책 제어를 참조하세요.

Jupyter 및 AWS Glue Studio 노트북용 AWS Glue 대화형 세션 구성

Jupyter 매직 소개

Jupyter 매직은 셀의 시작 부분이나 셀 전체에 실행할 수 있는 명령입니다. %로 시작하는 매직은 라인 매직이고 %%로 시작하면 셀 매직입니다. 라인 매직(예:%region 및 %connections)은 하나의 셀 안에서 여러 매직으로 실행될 수 있으며, 또는 다음 예제와 같이 셀 본문에 포함된 코드로 실행할 수 있습니다.

```
%region us-east-2
%connections my_rds_connection
dy_f = glue_context.create_dynamic_frame.from_catalog(database='rds_tables',
 table_name='sales_table')
```

셀 매직은 전체 셀을 사용해야 하며 명령이 여러 줄에 걸쳐 포함될 수 있습니다. %%sql의 예는 아래와 같습니다.

```
%%sql
select * from rds_tables.sales_table
```

Jupyter용 AWS Glue 대화형 세션에서 지원되는 매직

다음은 Jupyter Notebook용 AWS Glue 대화형 세션에서 사용할 수 있는 매직입니다.

세션 매직

명칭	유형	설명
%help	해당 사항 없음	모든 매직 명령에 대한 설명 및 입력 유형 목록을 반환합니다.

명칭	유형	설명
`%profile`	String	자격 증명 공급자로 사용할 프로파일을 AWS 구성에서 지정합니다.
`%region`	String	세션을 초기화할 AWS 리전을 지정합니다. 기본값은 `~/.aws/configure.` 에 있습니다. 예시: `%region us-west-1`
`%idle_timeout`	정수	셀이 실행된 후 세션이 시간 초과되기까지의 비활성 시간(분)입니다. Spark ETL 세션의 기본 유휴 제한 시간 값은 기본 제한 시간 값인 2,880분(48시간)입니다. 다른 세션 유형에 대해서는 해당 세션 유형에 대한 설명서를 참조하세요. 예시: `%idle_timeout 3000`
`%session_id`	해당 사항 없음	실행 중인 세션의 세션 ID를 반환합니다.
`%session_id_prefix`	String	형식이 [session_id_prefix]-[session_id]인 모든 세션 ID 앞에 오는 문자열을 정의합니다. 세션 ID가 제공되지 않으면 임의의 UUID가 생성됩니다. AWS Glue Studio에서 Jupyter Notebook을 실행할 때는 이 매직이 지원되지 않습니다. 예시: `%session_id_prefix 001`
`%status`		기간, 구성, 실행 사용자/역할을 포함한 현재 AWS Glue 세션의 상태를 반환합니다.
`%stop_session`		현재 세션을 중지합니다.
`%list_sessions`		현재 실행 중인 모든 세션을 이름과 ID로 나열합니다.

명칭	유형	설명
`%session_type`	String	세션 유형을 스트리밍, ETL 또는 Ray 중 하나로 설정합니다. 예시: `%session_type Streaming`
`%glue_version`	String	이 세션에서 사용할 AWS Glue 버전입니다. 예시: `%glue_version 3.0`

작업 유형 선택을 위한 매직

명칭	유형	설명
`%streaming`	String	세션 유형을 AWS Glue 스트리밍으로 변경합니다.
`%etl`	String	세션 유형을 AWS Glue ETL로 변경합니다.
%glue_ray	String	세션 유형을 AWS Glue for Ray로 변경합니다. AWS Glue Ray 대화형 세션에서 지원되는 매직을 참조하세요.

AWS Glue for Spart 구성 매직

`%%configure` 매직은 세션에 대한 모든 구성 파라미터로 구성된 JSON 형식의 사전입니다. 각 파라미터는 여기서 지정하거나 개별 매직을 통해 지정할 수 있습니다.

명칭	유형	설명
`%%configure`	딕셔너리	세션에 대한 모든 구성 파라미터로 구성된 JSON 포맷 딕셔너리를 지정합니다. 각 파라미터는 여기서 지정하거나 개별 매직을 통해 지정할 수 있습니다.

명칭	유형	설명
		`%%configure` 를 사용하는 방법에 대한 예제와 파라미터 목록은 %%configure 셀 매직 인수 섹션을 참조하세요.
`%iam_role`	String	세션을 실행하는 데 사용할 IAM 역할 ARN을 지정합니다. 기본값은 ~/.aws/configure에 있습니다. 예시: `%iam_role AWSGlueServiceRole`
`%number_of_workers`	정수	작업이 실행될 때 할당되는 정의된 worker_type의 작업자 수입니다. `worker_type` 도 설정해야 합니다. `number_of_workers` 의 기본값은 5입니다. 예시: `%number_of_workers 2`
`%additional_python_modules`	나열	클러스터에 포함할 추가 Python 모듈의 쉼표로 구분된 목록입니다(PyPI 또는 S3에서 가져올 수 있음). 예: `%additional_python_modules pandas, numpy`.

명칭	유형	설명

%%tags	String	세션에 태그를 추가합니다. 태그를 중괄호 {}로 묶어 지정합니다. 각 태그 이름 페어는 괄호(" ")로 묶고 쉼표(,)로 구분합니다.
		
		`%%tags`
		`{"billing":"Data-Platform",`
		`"team":"analytics"}`
		
		%status 매직을 사용하여 세션과 관련된 태그를 확인합니다.
		
		`%status`
		
		`Session ID: <sessionId>`
		`Status: READY`
		`Role: <example-role>`
		`CreatedOn: 2023-05-26 11:12:17.`
		`056000-07:00`
		`GlueVersion: 3.0`
		`Job Type: glueetl`
		`Tags: {'owner':'example-owner',`
		`'team':'analytics', 'billing'`
		`:'Data-Platform'}`
		`Worker Type: G.4X`
		`Number of Workers: 5`
		`Region: us-west-2`
		`Applying the following default`
		`arguments:`
		`--glue_kernel_version 0.38.0`
		`--enable-glue-datacatalog true`
		`Arguments Passed: ['--glue_`
		`kernel_version: 0.38.0', '--`
		`enable-glue-datacatalog: true']`

명칭	유형	설명

%%assume_role	사전	json 형식의 사전 또는 IAM 역할 ARN 문자열을 지정하여 크로스 계정 액세스를 위한 세션을 생성합니다.
		
		ARN 관련 예제:
		
		`%%assume_role`
		`{`
		`  'arn:aws:iam::XXXXXXXXXXXX:`
		`role/AWSGlueServiceRole'`
		`}`
		
		보안 인증 관련 예제:
		
		`%%assume_role`
		`{{`
		`  "aws_access_key_id" =`
		`"XXXXXXXXXXXX",`
		`  "aws_secret_access_key" =`
		`"XXXXXXXXXXXX",`
		`  "aws_session_token" =`
		`"XXXXXXXXXXXX"`
		`}}`

%%configure 셀 매직 인수

%%configure 매직은 세션에 대한 모든 구성 파라미터로 구성된 JSON 형식의 사전입니다. 각 파라미터는 여기서 지정하거나 개별 매직을 통해 지정할 수 있습니다. %%configure 셀 매직이 지원하는 인수 예제는 아래를 참조하세요. 작업에 지정된 실행 인수에 -- 접두사를 사용합니다. 예제:

```
%%configure
{
   "--user-jars-first": "true",
   "--enable-glue-datacatalog": "false"
}
```

작업 파라미터에 대한 자세한 내용은 작업 파라미터 섹션을 참조하세요.

세션 구성

파라미터	유형	설명

`max_retries`	정수	실패한 경우 이 작업을 다시 시도할 수 있는 최대 횟수입니다.
		`%%configure`
		`{`
		`  "max_retries": "0"`
		`}`

`max_concurrent_runs`	정수	작업에 허용된 최대 동시 실행 수입니다.
		예제:
		`%%configure`
		`{`
		`  "max_concurrent_runs": "3"`
		`}`

세션 파라미터

파라미터	유형	설명

`--enable-spark-ui`	불	Spark UI를 활성화하여 AWS Glue ETL 작업을 모니터링하고 디버깅합니다.
		`%%configure`
		`{`
		`  "--enable-spark-ui": "true"`
		`}`

`--spark-event-logs-path`	String	Amazon S3 경로를 지정합니다. Spark UI 모니터링 기능을 사용하는 경우.

파라미터	유형	설명

		예제:
		
		`%%configure`
		`{`
		`  "--spark-event-logs-path":`
		` "s3://path/to/event/logs/"`
		`}`

`--script_location`	String	작업을 실행하는 스크립트의 S3 경로를 지정합니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "script_location": "s3://new-folder-here"`
		`}`

`--SECURITY_CONFIGURATION`	String	AWS Glue 보안 구성의 이름입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--security_configuration":  { "encryption_type": "kms", "kms_key_id": "YOUR_KMS_KEY_ARN" }`
		`}`

파라미터	유형	설명

`--job-language`	String	스크립트 프로그래밍 언어. 'scala' 또는 'python'의 값이 허용됩니다. 기본값은 'python'입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--job-language": "scala"`
		`}`

`--class`	String	Scala 스크립트 진입점으로써 Scala 클래스. 기본값은 null입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--class": "className"`
		`}`

`--user-jars-first`	불	클래스 경로에서 고객의 추가 JAR 파일의 우선순위를 지정합니다. 기본값은 null입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--user-jars-first": "true"`
		`}`

파라미터	유형	설명

`--use-postgres-driver`	불	Amazon Redshift JDBC 드라이버와의 충돌을 피하기 위해 클래스 경로에서 Postgres JDBC 드라이버의 우선순위를 지정합니다. 기본값은 null입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--use-postgres-driver": "true"`
		`}`

`--extra-files`	List(string)	스크립트를 실행하기 전에 AWS Glue에서 스크립트의 작업 디렉터리에 복사하는 구성 파일과 같은 추가 파일에 대한 Amazon S3 경로입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--extra-files": "s3://path/to/additional/files/"`
		`}`

파라미터	유형	설명

`--job-bookmark-option`	String	작업 북마크 동작을 제어합니다. 'job-bookmark-enable', 'job-bookmark-disable' 또는 'job-bookmark-pause'의 값이 허용됩니다. 기본값은 'job-bookmark-disable'입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--job-bookmark-option": "job-bookmark-enable"`
		`}`

`--TempDir`	String	작업의 임시 디렉터리로 사용될 수 있는 버킷에 대한 Amazon S3 경로를 지정합니다. 기본값은 null입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--TempDir": "s3://path/to/temp/dir"`
		`}`

파라미터	유형	설명

`--enable-s3-parquet-optimized-committer`	불	Amazon S3에 Parquet 데이터를 쓸 수 있도록 EMRFS Amazon S3 최적화 커미터를 활성화합니다. 기본값은 'true'입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--enable-s3-parquet-optimized-committer": "false"`
		`}`

`--enable-rename-algorithm-v2`	불	EMRFS 이름 바꾸기 알고리즘 버전을 버전 2로 설정합니다. 기본값은 'true'입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--enable-rename-algorithm-v2": "true"`
		`}`

`--enable-glue-datacatalog`	불	AWS Glue 데이터 카탈로그를 Apache Spark Hive 메타스토어로 사용할 수 있습니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--enable-glue-datacatalog": "true"`
		`}`

파라미터	유형	설명

--enable-metrics	불	작업 실행을 위해 작업 프로파일링용 지표 수집을 활성화합니다. 기본값은 'false'입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--enable-metrics": "true"`
		`}`

--enable-continuous-cloudwatch-log	불	AWS Glue 작업에 대한 실시간 연속 로깅을 활성화합니다. 기본값은 'false'입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--enable-continuous-cloudwatch-log": "true"`
		`}`

--enable-continuous-log-filter	불	연속 로깅을 위해 활성화된 작업을 편집하거나 해당 작업을 생성할 때 표준 필터 또는 필터 없음을 지정합니다. 기본값은 'true'입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--enable-continuous-log-filter": "true"`
		`}`

파라미터	유형	설명

`--continuous-log-stream-prefix`	String	연속 로깅에 대해 활성화된 작업의 사용자 지정 Amazon CloudWatch 로그 스트림 접두사를 지정합니다. 기본값은 null입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--continuous-log-stream-prefix": "prefix"`
		`}`

`--continuous-log-conversionPattern`	String	연속 로깅에 대해 활성화된 작업의 사용자 지정 변환 로그 패턴을 지정합니다. 기본값은 null입니다.
		
		예제:
		
		`%%configure`
		`{`
		`  "--continuous-log-conversionPattern": "pattern"`
		`}`

파라미터	유형	설명

--conf	String	Spark 구성 파라미터를 제어합니다. 고급 사용 사례에 해당됩니다. 각 파라미터 앞에 --conf를 사용하세요. 예제:
		
		%%configure
		{
		"--conf": "spark.hadoop.hive
		.metastore.glue.catalogid=1
		23456789012 --conf hive.meta
		store.client.factory.class=
		com.amazonaws.glue.catalog.
		metastore.AWSGlueDataCatalo
		gHiveClientFactory --conf
		hive.metastore.schema.verif
		ication=false"
		}

제한 시간	정수	Spark 세션이 종료되기 전에 문이 완료될 때까지 기다려야 하는 최대 시간을 결정합니다.
		
		%%configure
		{
		"timeout": "30"
		}

Auto Scaling	불	오토 스케일링 사용 여부를 결정합니다.
		
		%%configure
		{
		"--enable-auto-scaling": "true"
		}

Spark 작업(ETL 및 스트리밍) 매직

명칭	유형	설명
`%worker_type`	String	표준, G.1X 또는 G.2X가 있습니다. `number_of_workers` 도 설정해야 합니다. 기본 worker_type은 G.1X입니다.
`%connections`	나열	세션에서 사용할 연결을 쉼표로 구분된 목록으로 지정합니다. 예제: `%connections my_rds_connection dy_f = glue_context.create_dynamic _frame.from_catalog(databas e='rds_tables', table_nam e='sales_table')`
`%extra_py_files`	나열	Amazon S3에 있는 추가 Python 파일의 쉼표로 구분된 목록입니다.
`%extra_jars`	나열	클러스터에 포함할 추가 jar의 쉼표로 구분된 목록입니다.
`%spark_conf`	String	세션의 사용자 지정 Spartk 구성을 지정합니다. 예: `%spark_conf spark.ser ializer=org.apache.spark.se rializer.KryoSerializer` .

Ray 작업을 위한 매직

명칭	유형	설명
`%min_workers`	정수	Ray 작업에 할당되는 작업자의 최소 수입니다. 기본값: 1.

명칭	유형	설명
		예시: `%min_workers 2`
`%object_memory_head`	정수	웜 스타트 이후 인스턴스 헤드 노드에서 사용 가능한 메모리의 백분율입니다. 최소값: 0 최대값: 100 예시: `%object_memory_head 100`
%object_memory_worker	정수	웜 스타트 이후 인스턴스 워커 노드에서 사용 가능한 메모리의 백분율입니다. 최소값: 0 최대값: 100 예시: `%object_memory_worker 100`

작업 매직

명칭	유형	설명
`%%sql`	String	SQL 코드를 실행합니다. 처음 `%%sql` 매직 이후의 모든 줄이 SQL 코드의 일부로 전달됩니다. 예시: `%%sql select * from rds_tables.sales_table`

`%matplot`	Matplotlib 수치	matplotlib 라이브러리를 사용하여 데이터를 시각화합니다.
		
		예제:
		
		`import matplotlib.pyplot as plt`
		
		`# Set X-axis and Y-axis values`
		`x = [5, 2, 8, 4, 9]`
		`y = [10, 4, 8, 5, 2]`
		
		`# Create a bar chart`
		`plt.bar(x, y)`

명칭	유형	설명
		 ``` # Show the plot %matplot plt ```

%plotly	Plotly 수치	plotly 라이브러리를 사용하여 데이터를 시각화합니다.
		
		예제:
		
		``
		import plotly.express as px
		
		#Create a graphical figure
		fig = px.line(x=["a","b","c"],
		 y=[1,3,2], title="sample
		 figure")
		
		#Show the figure
		%plotly fig
		``

세션 이름 지정

AWS Glue 대화형 세션은 AWS 리소스이며 이름이 필요합니다. 이름은 각 세션마다 고유해야 하며 IAM 관리자가 제한할 수 있습니다. 자세한 내용은 IAM을 이용한 대화형 세션 단원을 참조하십시오. Jupyter 커널은 자동으로 고유한 세션 이름을 생성합니다. 그러나 세션의 이름은 두 가지 방법으로 직접 지정할 수 있습니다.

1. ~.aws/config에 있는 AWS Command Line Interface 구성 파일을 사용합니다. AWS Command Line Interface를 사용하여 AWS Config 설정을 참조하세요.
2. %session_id_prefix 매직을 사용합니다. Jupyter용 AWS Glue 대화형 세션에서 지원되는 매직 섹션을 참조하세요.

세션 이름은 다음과 같이 생성됩니다.

- 접두사와 session_id가 제공된 경우 세션 이름은 {접두사}-{UUID}가 됩니다.
- 제공된 항목이 없을 경우 세션 이름은 {UUID}가 됩니다.

세션 이름에 접두사를 사용하면 세션을 AWS CLI 또는 콘솔에 나열할 때 식별할 수 있습니다.

대화형 세션에 대한 IAM 역할 지정

대화형 세션에서 실행하는 AWS ETL 코드에 사용할 AWS Glue Identity and Access Management(IAM) 역할을 지정해야 합니다.

역할에는 AWS Glue 작업을 실행하는 데 필요한 IAM 권한과 동일한 IAM 권한이 필요합니다. AWS Glue 작업 및 대화형 세션을 위한 역할 생성에 대한 자세한 내용은 AWS Glue에 대한 IAM 역할 생성을 참조하세요.

IAM 역할은 두 가지 방법으로 지정할 수 있습니다.

- `~.aws/config`에 있는 AWS Command Line Interface 구성 파일을 사용합니다(권장). 자세한 내용은 ~/.aws/config를 사용하여 세션 구성을 참조하세요.

  > **Note**
  >
  > `%profile` 매직이 사용될 때 해당 프로파일의 `glue_iam_role`에 대한 구성이 적용됩니다.

- %iam_role 매직을 사용합니다. 자세한 내용은 Jupyter용 AWS Glue 대화형 세션에서 지원되는 매직 단원을 참조하십시오.

명명된 프로파일을 사용하여 세션 구성

AWS Glue 대화형 세션은 AWS Command Line Interface 또는 boto3와 동일한 보안 인증을 사용하며, 대화형 세션은 `~/.aws/config`(Linux 및 MacOS) 또는 `%USERPROFILE%\.aws\config`(Windows)에 있는 AWS CLI와(과) 같은 명명된 프로파일을 인식하고 작동합니다. 자세한 내용은 명명된 프로파일 사용을 참조하세요.

대화형 세션은 AWS Glue 서비스 역할 및 세션 ID 접두사가 프로파일에서 지정될 수 있도록 하여 명명된 프로파일을 활용합니다. 프로파일 역할을 구성하려면 아래와 같이 `iam_role` 키 및/또는 `session_id_prefix` 에 대한 줄을 명명된 프로파일에 추가합니다. `session_id_prefix`에는 따옴표가 필요하지 않습니다. 예를 들어 `session_id_prefix`를 추가하려면 `session_id_prefix=myprefix`의 값을 입력합니다.

```
[default]
region=us-east-1
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
```

```
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
glue_iam_role=arn:aws:iam::<AccountID>:role/<GlueServiceRole>
session_id_prefix=<prefix_for_session_names>

[user1]
region=eu-west-1
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
glue_iam_role=arn:aws:iam::<AccountID>:role/<GlueServiceRoleUser1>
session_id_prefix=<prefix_for_session_names_for_user1>
```

자격 증명을 생성하는 사용자 지정 방법이 있는 경우, `~/.aws/config` 파일에서 `credential_process` 파라미터를 사용하도록 프로파일을 구성할 수도 있습니다 예:

```
[profile developer]
region=us-east-1
credential_process = "/Users/Dave/generate_my_credentials.sh" --username helen
```

`credential_process` 파라미터를 통한 자격 증명 소싱에 대한 자세한 내용은 외부 프로세스를 통해 자격 증명 소싱을 참조하세요.

사용 중인 프로파일에 리전 또는 `iam_role`이 설정되어 있지 않으면 실행하는 첫 번째 셀에서 `%region` 및 `%iam_role` 매직을 사용하여 지정해야 합니다.

스크립트 또는 노트북을 AWS Glue 작업으로 변환

스크립트나 노트북을 AWS Glue 작업으로 변환할 수 있는 방법은 다음과 같이 두 가지가 있습니다.

- nbconvert를 사용하여 Jupyter `.ipynb` Notebook 문서 파일을 `.py` 파일로 변환합니다. 자세한 내용은 nbconvert를 이용하여 노트북을 다른 형식으로 변환을 참조하세요.
- AWS Glue Studio 노트북에 파일을 업로드합니다.
  - AWS Glue Studio 콘솔의 탐색 메뉴에서 작업(Jobs)을 선택합니다.
  - 작업 생성(Create job) 섹션에서 Jupyter Notebook을 선택합니다.
  - 옵션(Options) 섹션에서 기존 노트북 업로드 및 편집(Upload and edit an existing notebook)을 선택합니다.
  - 파일 선택(Choose file)을 선택하여 `.ipynb` 파일을 업로드합니다.

AWS Glue 대화형 세션에서 스트리밍 작업 수행

스트리밍 세션 유형 전환

AWS Glue 대화형 세션 구성 매직(`%streaming`)을 사용하여 실행 중인 작업을 정의하고 스트리밍 대화형 세션을 초기화합니다.

대화형 개발을 위한 샘플링 입력 스트림

AWS Glue 대화형 세션의 대화형 경험을 개선하기 위해 도출한 한 가지 방법은 정적 DynamicFrame에서 스트림의 스냅샷을 얻을 수 있도록 `GlueContext`에 새로운 방법을 추가하는 것입니다. `GlueContext`는 워크플로를 검사, 상호 작용, 구현할 수 있도록 합니다.

`GlueContext` 클래스 인스턴스를 사용하면 `getSampleStreamingDynamicFrame` 메서드를 찾을 수 있습니다. 이 메서드의 필수 인수는 다음과 같습니다.

- `dataFrame`: Spark Streaming Dataframe
- `options`: 아래 사용 가능한 옵션 참조

사용 가능한 옵션은 다음과 같습니다.

- windowSize: 마이크로 배치 기간이라고도 합니다. 이 파라미터는 이전 배치가 트리거된 후 스트리밍 쿼리가 대기하는 시간을 결정합니다. 이 파라미터 값은 `pollingTimeInMs`보다 작아야 합니다.
- pollingTimeInMs: 메서드가 실행될 총 시간입니다. 입력 스트림에서 샘플 레코드를 얻기 위해 적어도 하나의 마이크로 배치를 실행합니다.
- recordPollingLimit: 이 파라미터를 사용하면 스트림에서 폴링할 총 레코드 수를 제한할 수 있습니다.
- (선택 사항) `writeStreamFunction`을 사용하여 모든 레코드 샘플링 함수에 이 사용자 지정 함수를 적용할 수도 있습니다. Scala 및 Python 예제는 아래를 참조하세요.

Scala

```
val sampleBatchFunction = (batchDF: DataFrame, batchId: Long) => {//Optional but
 you can replace your own forEachBatch function here}
val jsonString: String = s"""{"pollingTimeInMs": "10000", "windowSize": "5
 seconds"}"""
```

```
val dynFrame = glueContext.getSampleStreamingDynamicFrame(YOUR_STREAMING_DF,
 JsonOptions(jsonString), sampleBatchFunction)
dynFrame.show()
```

Python

```
def sample_batch_function(batch_df, batch_id):
       //Optional but you can replace your own forEachBatch function here
options = {
            "pollingTimeInMs": "10000",
            "windowSize": "5 seconds",
        }
glue_context.getSampleStreamingDynamicFrame(YOUR_STREAMING_DF, options,
 sample_batch_function)
```

> **Note**
>
> 몇 가지 이유로 인해 샘플링된 DynFrame이 비어 있는 경우가 발생할 수 있습니다.
>
> - 스트리밍 소스가 '최신'으로 설정되어 있으며 샘플링 기간 동안 새 데이터가 수집되지 않았습니다.
> - 폴링 시간이 충분하지 않아 수집된 레코드를 처리할 수 없습니다. 전체 배치가 처리되지 않으면 데이터가 표시되지 않습니다.

대화형 세션에서 스트리밍 애플리케이션 실행

AWS Glue 대화형 세션에서는 AWS Glue 콘솔에서 스트리밍 애플리케이션을 생성하는 것처럼 AWS Glue 스트리밍 애플리케이션을 실행할 수 있습니다. 대화형 세션은 세션 기반이므로 런타임에 예외가 발생해도 세션이 중지되지 않습니다. 이제 배치 함수를 반복적으로 개발할 수 있다는 추가 이점이 있습니다. 예:

```
def batch_function(data_frame, batch_id):
    log.info(data_frame.count())
    invalid_method_call()
glueContext.forEachBatch(frame=streaming_df, batch_function = batch_function, options =
 {**})
```

위의 예에서는 잘못된 메서드 사용을 포함했고, 전체 애플리케이션을 종료하는 일반 AWS Glue 작업과는 달리 사용자의 코딩 컨텍스트 및 정의가 완전히 보존되며 세션이 여전히 작동 중입니다. 새 클러스터를 부트스트랩하고 모든 이전 변환을 다시 실행할 필요가 없습니다. 이를 통해 배치 함수 구현을 신속하게 반복하여 바람직한 결과를 얻을 수 있습니다.

대화형 세션은 세션이 한 번에 하나의 문만 실행하도록 각 문을 차단 방식으로 평가한다는 점에 유의해야 합니다. 스트리밍 쿼리는 지속적이고 끝나지 않으므로 활성 스트리밍 쿼리가 포함된 세션은 중단되지 않는 한 어떤 후속 문도 처리할 수 없습니다. Jupyter Notebook에서 직접 중단 명령을 실행할 수 있으며 커널이 취소를 처리할 것입니다.

실행 대기 중인 다음 일련의 문을 예로 들어 보겠습니다.

```
Statement 1:
      val number = df.count()
      #Spark Action with deterministic result
      Result: 5

Statement 2:
      streamingQuery.start().awaitTermination()
      #Spark Streaming Query that will be executing continously
      Result: Constantly updated with each microbatch

Statement 3:
      val number2 = df.count()
      #This will not be executed as previous statement will be running indefinitely
```

로컬로 AWS Glue 작업 스크립트 개발 및 테스트

Spark용 AWS Glue 작업 스크립트를 개발하고 테스트할 때 사용할 수 있는 옵션은 여러 가지가 있습니다.

- AWS Glue Studio 콘솔
  - Visual editor(시각적 편집기)
  - 스크립트 에디터
  - AWS Glue Studio 노트북

- 대화형 세션
  - Jupyter Notebook
- 도커 이미지
  - 로컬 개발
  - 원격 개발

요구 사항에 따라 위의 옵션 중 하나를 선택할 수 있습니다.

코드가 없거나 코드 경험이 적은 경우 AWS Glue Studio 시각적 편집기를 선택하는 것이 좋습니다.

대화형 노트북 환경을 선호하는 경우 AWS Glue Studio 노트북을 선택하는 것이 좋습니다. 자세한 내용은 AWS Glue Studio 및 AWS Glue를 사용하여 노트북 사용을 참조하세요. 로컬 환경을 직접 사용하려는 경우 대화형 세션을 선택하는 것이 좋습니다. 자세한 내용은 AWS Glue를 사용하여 대화형 세션 사용을 참조하세요.

로컬/원격 개발 환경을 선호하는 경우 Docker 이미지를 사용하는 것이 좋습니다. 이를 통해 AWS Glue 비용을 들이지 않고도 원하는 곳에서 AWS Glue for Spark 작업 스크립트를 개발하고 테스트할 수 있습니다.

Docker를 사용하지 않고 로컬 개발을 선호하는 경우 AWS Glue ETL 라이브러리 디렉터리를 사용하는 것이 좋습니다.

AWS Glue Studio를 사용한 개발

AWS Glue Studio 시각적 편집기는 AWS Glue에서 추출, 전환, 적재(ETL) 작업을 쉽게 생성, 실행, 모니터링할 수 있게 해주는 그래픽 인터페이스입니다. 데이터 변환 워크플로를 시각적으로 구성하고 AWS Glu의 Apache Spark 기반 서버리스 ETL 엔진에서 원활하게 실행할 수 있습니다. 작업의 각 단계에서 스키마 및 데이터 결과를 검사할 수 있습니다. 자세한 내용은 AWS Glue Studio User Guide를 참조하세요.

대화형 세션을 사용하여 개발

대화형 세션을 사용하면 선택한 환경에서 애플리케이션을 구축하고 테스트할 수 있습니다. 자세한 내용은 AWS Glue를 사용하여 대화형 세션 사용을 참조하세요.

Docker 이미지를 사용하여 로컬에서 AWS Glue 작업 개발 및 테스트

프로덕션 준비 데이터 플랫폼의 경우 AWS Glue에 대한 개발 프로세스 및 CI/CD 파이프라인 작업이 핵심 주제입니다. 도커 컨테이너에서 AWS Glue 작업을 유연하게 개발하고 테스트할 수 있습니다. AWS

Glue는 Docker Hub에서 도커 이미지를 호스팅하여 추가 유틸리티로 개발 환경을 설정합니다. AWS Glue ETL 라이브러리를 사용하여 선호하는 IDE, 노트북 또는 REPL을 사용할 수 있습니다. 이 주제에서는 Docker 이미지를 사용하여 Docker 컨테이너에서 AWS Glue 버전 5.0 작업을 개발하고 테스트하는 방법을 설명합니다.

사용 가능한 Docker 이미지

Amazon ECR의 AWS Glue에 대해 다음 Docker 이미지를 사용할 수 있습니다.

- AWS Glue 버전 5.0: `public.ecr.aws/glue/aws-glue-libs:5`
- AWS Glue 버전 4.0의 경우: `public.ecr.aws/glue/aws-glue-libs:glue_libs_4.0.0_image_01`
- AWS Glue 버전 3.0의 경우: `public.ecr.aws/glue/aws-glue-libs:glue_libs_3.0.0_image_01`
- AWS Glue 버전 2.0의 경우: `public.ecr.aws/glue/aws-glue-libs:glue_libs_2.0.0_image_01`

> **Note**
>
> AWS Glue Docker 이미지는 x86_64 및 arm64와 호환됩니다.

이 예제에서는 `public.ecr.aws/glue/aws-glue-libs:5`를 사용하고 로컬 시스템(Mac, Windows, Linux)에서 컨테이너를 실행합니다. 이 컨테이너 이미지는 AWS Glue 버전 5.0 Spark 작업에 대해 테스트되었습니다. 이미지에는 다음이 포함되어 있습니다.

- Amazon Linux 2023
- AWS Glue ETL 라이브러리
- Apache Spark 3.5.4
- 오픈 테이블 형식 라이브러리; Apache Iceberg 1.7.1, Apache Hudi 0.15.0, Delta Lake 3.3.0
- AWS Glue Data Catalog 클라이언트
- Apache Spark용 Amazon Redshift 커넥터
- Apache Hadoop용 Amazon DynamoDB 커넥터

컨테이너를 설정하려면 ECR 퍼블릭 갤러리에서 이미지를 가져온 다음 컨테이너를 실행합니다. 이 주제에서는 요구 사항에 따라 다음 방법으로 컨테이너를 실행하는 방법을 보여줍니다.

- `spark-submit`
- REPL 쉘 (`pyspark`)
- `pytest`
- Visual Studio Code

사전 조건

시작하기 전에 Docker가 설치되어 있고 Docker 데몬이 실행 중인지 확인하세요. 설치 지침은 Mac 또는 Linux용 도커 설명서를 참조하세요. Docker를 실행하는 시스템은 AWS Glue 컨테이너를 호스트합니다. 또한 Docker를 실행하는 호스트의 이미지를 위해 7GB 이상의 디스크 공간이 있는지 확인하세요.

로컬에서 AWS Glue 코드 개발 시 제한에 대한 자세한 정보는 로컬 개발 제한 사항을 참조하세요.

AWS 구성

컨테이너에서 AWS API 호출을 활성화하려면 다음 단계에 따라 AWS 자격 증명을 설정합니다. 다음 섹션에서는 AWS(이)라는 이름의 프로필을 사용합니다.

1. AWS 명명된 프로필을 생성합니다.
2. Windows 또는 Mac/Linux의 터미널에서 cmd를 열고 터미널에서 다음 명령을 실행합니다.

```
PROFILE_NAME="<your_profile_name>"
```

다음 섹션에서는 AWS 명명된 프로필을 사용합니다.

Windows에서 Docker를 실행하는 경우 이미지를 가져오기 전에 Docker 아이콘(마우스 오른쪽 버튼 클릭)을 선택하고 Linux 컨테이너로 전환을 선택합니다.

ECR 퍼블릭으로부터 이미지를 로컬 시스템으로 가져오려면 다음 명령을 실행합니다.

```
docker pull public.ecr.aws/glue/aws-glue-libs:5
```

컨테이너 실행

이제 이 이미지를 사용하여 컨테이너를 실행할 수 있습니다. 요구 사항에 따라 다음 중 하나를 선택할 수 있습니다.

spark-submit

컨테이너에서 `spark-submit` 명령을 실행하여 AWS Glue 작업 스크립트를 실행할 수 있습니다.

1. 아래 예제와 같이 스크립트를 작성하고 `sample.py`로 저장한 이후 다음 명령을 사용하여 `/local_path_to_workspace/src/` 디렉터리에 저장합니다.

```
$ WORKSPACE_LOCATION=/local_path_to_workspace
$ SCRIPT_FILE_NAME=sample.py
$ mkdir -p ${WORKSPACE_LOCATION}/src
$ vim ${WORKSPACE_LOCATION}/src/${SCRIPT_FILE_NAME}
```

2. 이러한 변수는 아래의 docker run 명령에 사용됩니다. 아래의 spark-submit 명령에 사용되는 샘플 코드(sample.py://)는 이 주제의 끝에 있는 부록에 포함되어 있습니다.

   다음 명령을 실행하여 컨테이너에서 `spark-submit` 명령을 실행하여 새 Spark 애플리케이션을 제출합니다.

```
$ docker run -it --rm \
    -v ~/.aws:/home
    /hadoop/.aws \
    -v $WORKSPACE_LOCATION:/home/hadoop/workspace/ \
    -e AWS_PROFILE=$PROFILE_NAME \
    --name glue5_spark_submit \
    public.ecr.aws/glue/aws-glue-libs:5 \
    spark-submit /home/hadoop/workspace/src/$SCRIPT_FILE_NAME
```

3. (선택 사항) 환경에 맞게 `spark-submit`을 구성합니다. 예를 들어 `--jars` 구성을 사용하여 종속성을 전달할 수 있습니다. 자세한 내용은 Spark 설명서에서 Spark Properties 동적 로딩을 참조하세요.

REPL 셸(Pyspark)

대화형 개발을 위해 REPL(`read-eval-print loops`) 쉘을 실행할 수 있습니다. 다음 명령을 실행하여 컨테이너에서 PySpark 명령을 실행하여 REPL 셸을 시작합니다.

```
$ docker run -it --rm \
    -v ~/.aws:/home/hadoop/.aws \
    -e AWS_PROFILE=$PROFILE_NAME \
    --name glue5_pyspark \
    public.ecr.aws/glue/aws-glue-libs:5 \
    pyspark
```

출력은 다음과 같습니다.

```
Python 3.11.6 (main, Jan  9 2025, 00:00:00) [GCC 11.4.1 20230605 (Red Hat 11.4.1-2)] on
 linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
 setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.4-amzn-0
      /_/

Using Python version 3.11.6 (main, Jan  9 2025 00:00:00)
Spark context Web UI available at None
Spark context available as 'sc' (master = local[*], app id = local-1740643079929).
SparkSession available as 'spark'.
>>>
```

이 REPL 쉘을 사용하면 대화형으로 코딩 및 테스트할 수 있습니다.

Pytest

유닛 테스트의 경우 AWS Glue Spark 작업 스크립트에 pytest를 사용할 수 있습니다. 준비를 위해 다음 명령을 실행합니다.

```
$ WORKSPACE_LOCATION=/local_path_to_workspace
$ SCRIPT_FILE_NAME=sample.py
$ UNIT_TEST_FILE_NAME=test_sample.py
$ mkdir -p ${WORKSPACE_LOCATION}/tests
$ vim ${WORKSPACE_LOCATION}/tests/${UNIT_TEST_FILE_NAME}
```

다음 명령을 실행하고 docker run을 사용하여 pytest를 실행합니다.

```
$ docker run -i --rm \
    -v ~/.aws:/home/hadoop/.aws \
    -v $WORKSPACE_LOCATION:/home/hadoop/workspace/ \
    --workdir /home/hadoop/workspace \
    -e AWS_PROFILE=$PROFILE_NAME \
    --name glue5_pytest \
    public.ecr.aws/glue/aws-glue-libs:5 \
```

```
    -c "python3 -m pytest --disable-warnings"
```

pytest가 단위 테스트 실행을 완료하면 출력은 다음과 같습니다.

```
============================= test session starts ==============================
platform linux -- Python 3.11.6, pytest-8.3.4, pluggy-1.5.0
rootdir: /home/hadoop/workspace
plugins: integration-mark-0.2.0
collected 1 item

tests/test_sample.py .                                                   [100%]

======================== 1 passed, 1 warning in 34.28s =========================
```

Visual Studio Code를 사용하도록 컨테이너 설정

Visual Studio Code로 컨테이너를 설정하려면 다음 단계를 완료합니다.

1. Visual Studio Code를 설치합니다.
2. Python을 설치합니다.
3. Visual Studio Code Remote - Containers를 설치합니다.
4. Visual Studio Code에서 작업 영역 폴더를 엽니다.
5. Ctrl+Shift+P(Windows/Linux) 또는 Cmd+Shift+P(Mac)를 누릅니다.
6. 유형 `Preferences: Open Workspace Settings (JSON)`
7. Enter를 누릅니다.
8. 다음 JSON을 붙여 넣고 저장합니다.

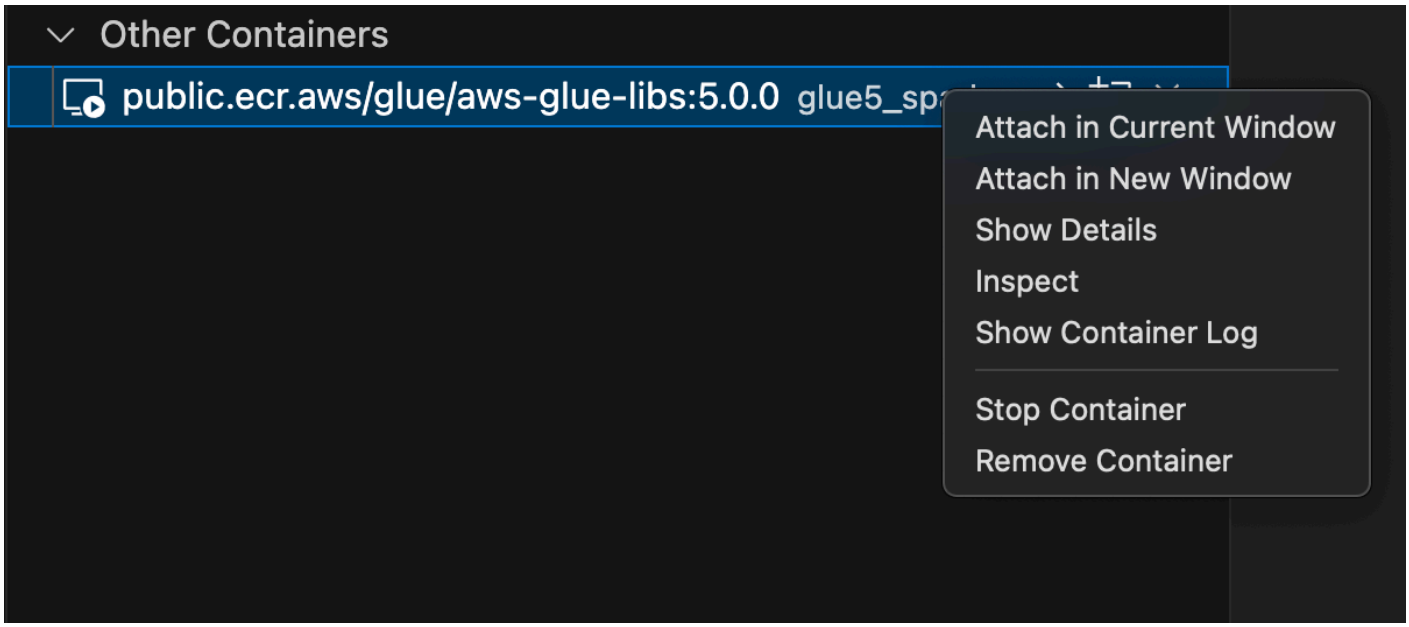
```
{
    "python.defaultInterpreterPath": "/usr/bin/python3.11",
    "python.analysis.extraPaths": [
        "/usr/lib/spark/python/lib/py4j-0.10.9.7-src.zip:/usr/lib/spark/python/:/usr/
lib/spark/python/lib/",
    ]
}
```

컨테이너 설정:

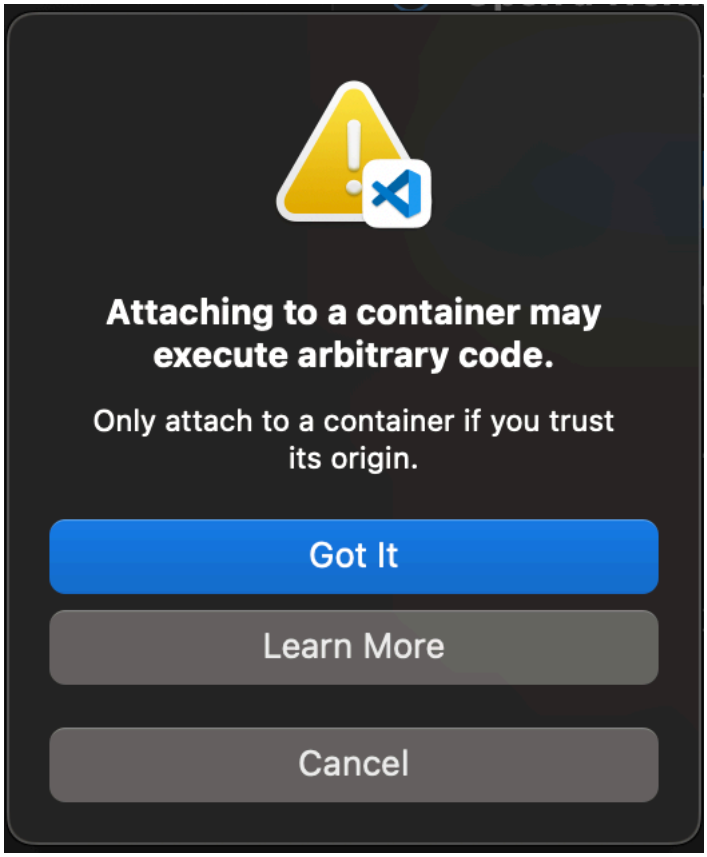
1. Docker 컨테이너를 실행합니다.

```
$ docker run -it --rm \
    -v ~/.aws:/home/hadoop/.aws \
    -v $WORKSPACE_LOCATION:/home/hadoop/workspace/ \
    -e AWS_PROFILE=$PROFILE_NAME \
    --name glue5_pyspark \
    public.ecr.aws/glue/aws-glue-libs:5 \
    pyspark
```

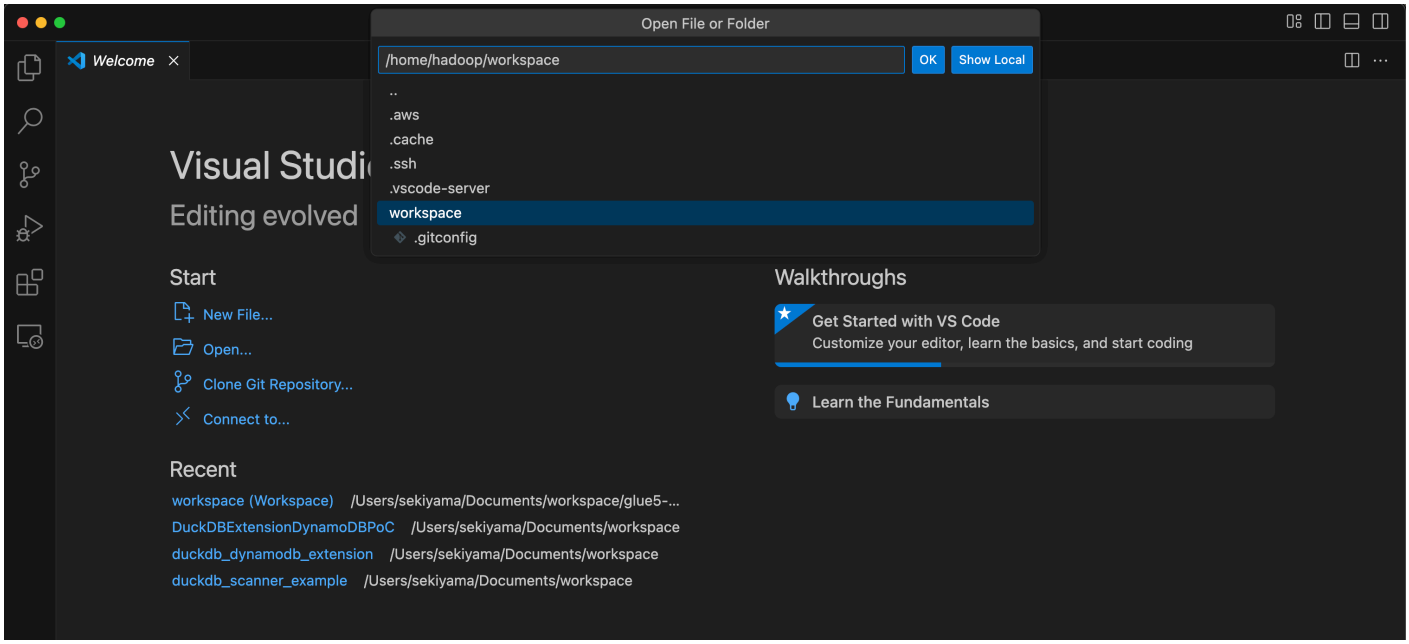
2. Visual Studio Code를 시작합니다.
3. 왼쪽 메뉴에서 Remote Explorer를 선택하고 amazon/aws-glue-libs:glue_libs_4.0.0_image_01을(를) 선택합니다.
4. 마우스 오른쪽 버튼을 클릭하고 현재 창에서 연결을 선택합니다.



5. 다음 대화 상자가 나타나면 알겠습니다를 선택합니다.

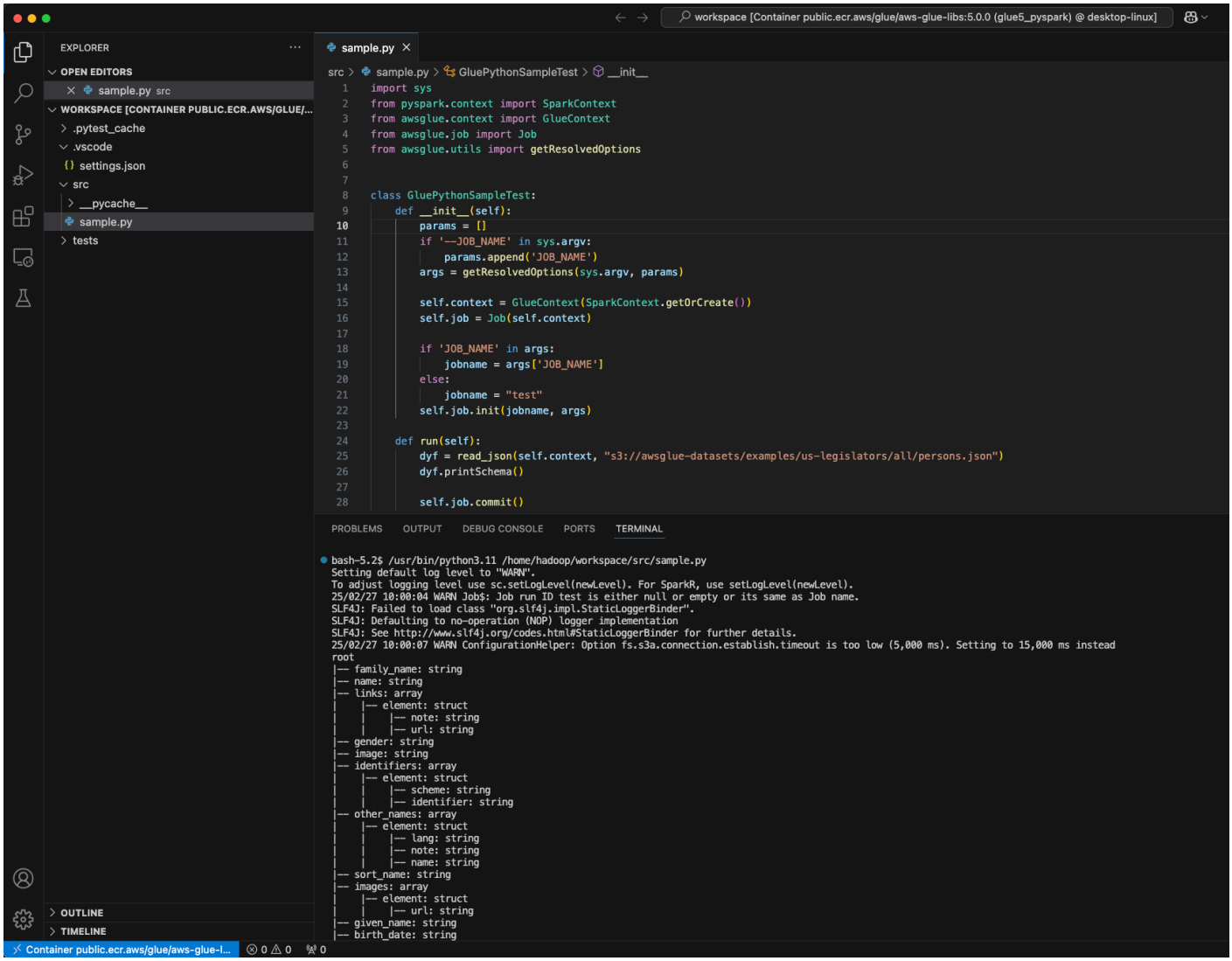


6. /home/handoop/workspace/를 엽니다.



7. AWS Glue PySpark 스크립트를 만들고 실행을 선택합니다.

   스크립트가 성공적으로 실행됨을 확인할 수 있습니다.



AWS Glue 4.0과 AWS Glue 5.0 Docker 이미지 간의 변경 사항

AWS Glue 4.0과 AWS Glue 5.0 Docker 이미지 간의 주요 변경 사항:

- AWS Glue 5.0에는 배치 작업과 스트리밍 작업 모두에 사용하는 단일 컨테이너 이미지가 있습니다. 이와 달리 Glue 4.0은 배치 작업용 이미지와 스트리밍 작업용 이미지가 하나씩 있었습니다.
- AWS Glue 5.0에서 컨테이너의 기본 사용자 이름은 hadoop입니다. AWS Glue 4.0에서 기본 사용자 이름은 `glue_user`였습니다.
- AWS Glue 5.0에서는 JupyterLab과 Livy를 포함한 여러 추가 라이브러리가 이미지에서 제거되었습니다. 이를 수동으로 설치할 수 있습니다.

- AWS Glue 5.0에서는 모든 Iceberg, Hudi 및 Delta 라이브러리가 기본적으로 사전 로드되며 환경 변수 DATALAKE_FORMATS는 더 이상 필요하지 않습니다. AWS Glue 4.0 이하의 경우 로드해야 하는 특정 테이블 형식을 지정하는 데 환경 변수 DATALAKE_FORMATS가 사용되었습니다.

위 목록은 Docker 이미지에 해당됩니다. AWS Glue 5.0 업데이트를 자세히 알아보려면 AWS Glue 5.0 for Apache Spark 소개 및 AWS Glue for Spark 작업을 AWS Glue 버전 5.0으로 마이그레이션을 참조하세요.

고려 사항

AWS Glue 컨테이너 이미지를 사용하여 로컬에서 작업 스크립트를 개발할 때는 다음 기능이 지원되지 않습니다.

- 작업 북마크
- AWS Glue Parquet 라이터(AWS Glue에서 Parquet 형식 사용)
- FillMissingValues 변환
- FindMatches 변환
- 벡터화된 SIMD CSV 리더
- Amazon S3 경로에서 JDBC 드라이버를 로드하는 데 사용되는 customJdbcDriverS3Path 속성
- AWS Glue 데이터 품질
- 민감한 데이터 감지
- AWS Lake Formation 권한 기반 자격 증명 벤딩

부록: JDBC 드라이버 및 Java 라이브러리 추가

컨테이너에서 현재 사용할 수 없는 JDBC 드라이버를 추가하려면 필요한 JAR 파일이 포함된 새 디렉터리를 워크스페이스에 생성하고 디렉터리를 docker run 명령의 /opt/spark/jars/에 마운트할 수 있습니다. 컨테이너 내의 /opt/spark/jars/에 있는 JAR 파일은 Spark Classpath에 자동으로 추가되며 작업 실행 중에 사용할 수 있습니다.

예를 들어 다음 docker run 명령을 사용하여 JDBC 드라이버 jar을 PySpark REPL 쉘에 추가합니다.

```
docker run -it --rm \
    -v ~/.aws:/home/hadoop/.aws \
    -v $WORKSPACE_LOCATION:/home/hadoop/workspace/ \
    -v $WORKSPACE_LOCATION/jars/:/opt/spark/jars/ \
    --workdir /home/hadoop/workspace \
```

```
    -e AWS_PROFILE=$PROFILE_NAME \
    --name glue5_jdbc \
    public.ecr.aws/glue/aws-glue-libs:5 \
    pyspark
```

고려 사항에서 강조하는 것처럼 `customJdbcDriverS3Path` 연결 옵션을 사용하여 AWS Glue 컨테이너 이미지의 Amazon S3에서 사용자 지정 JDBC 드라이버를 가져올 수 없습니다.

개발 엔드포인트

> **Note**
>
> 개발 엔드포인트의 콘솔 경험은 2023년 3월 31일을 기점으로 제거됩니다. 개발 엔드포인트의 생성, 업데이트 및 모니터링은 개발 엔드포인트 API 및 AWS Glue CLI를 통해 계속 사용 가능합니다.

아래 나열된 이유로 개발 엔드포인트에서 대화형 세션으로 마이그레이션하는 것이 좋습니다. 개발 엔드포인트에서 대화형 세션으로 마이그레이션하는 방법에 대한 필수 작업은 개발 엔드포인트에서 대화형 세션으로 마이그레이션을 참조하세요.

설명	dev 엔드포인트	대화형 세션
Glue 버전 지원	AWS Glue 버전 0.9 및 1.0을 지원합니다	AWS Glue 버전 2.0 이상을 지원합니다.
개발 엔드포인트는 향후 아시아 태평양(자카르타)(`ap-southeast-3`), 중동(UAE)(`me-central-1`), 유럽(스페인)(`eu-south-2`), 유럽(취리히)(`eu-central-2`) 또는 기타 신규 리전에서 사용할 수 없습니다.	대화형 세션은 현재 중동(UAE)(`me-central-1`) 리전에서는 사용할 수 없지만 나중에 사용할 수 있습니다.	
Spark 클러스터에 대한 액세스 방법	SSH, REPL 쉘, Jupyter Notebook, IDE(예: PyCharm) 지원	AWS Glue Studio 노트북, Jupyter Notebook, 다양한 IDE(예: Visual Studio Code,

설명	dev 엔드포인트	대화형 세션
		PyCharm) 및 SageMaker AI 노트북 지원
첫 번째 쿼리 시간	Spark 클러스터를 설정하는 데 10~15분 소요	임시 Spark 클러스터를 설정하는 데 최대 1분이 소요될 수 있음
가격 모델	AWS에서는 엔드포인트가 프로비저닝된 시간과 DPU 수를 기준으로 개발 엔드포인트 요금이 청구됩니다. 개발 엔드포인트는 제한 시간이 없습니다. 프로비저닝된 각 개발 엔드포인트에는 최소 10분의 청구 기간이 있습니다. 또한 AWS에서는 Amazon EC2 인스턴스의 Jupyter Notebook과 개발 엔드포인트로 구성한 SageMaker AI 노트북에 대한 요금을 부과합니다.	AWS는 세션이 활성화된 시간과 DPU 수를 기준으로 대화형 세션 요금을 청구합니다. 대화형 세션에는 유휴 시간 제한을 구성할 수 있습니다. AWS Glue Studio 노트북에는 대화형 세션을 위한 내장 인터페이스가 추가 비용 없이 제공됩니다. 각 대화형 세션에는 최소 1분의 청구 기간이 있습니다. AWS Glue Studio 노트북은 대화형 세션을 위한 기본 제공 인터페이스를 추가 비용 없이 제공합니다.
콘솔 경험	CLI 및 API를 통해서만 사용 가능	AWS Glue 콘솔, CLI 및 API를 통해 사용 가능

개발 엔드포인트에서 대화형 세션으로 마이그레이션

다음 체크리스트를 사용하여 개발 엔드포인트에서 대화형 세션으로 마이그레이션하는 적절한 방법을 결정합니다.

스크립트가 AWS Glue 0.9 또는 1.0 특정 기능(예: HDFS, YARN 등)에 종속됩니까?

대답이 '예'인 경우 Glue 0.9 또는 1.0에서 Glue 3.0 이상으로 마이그레이션하는 방법을 알아보려면 AWS Glue 버전 3.0으로 AWS Glue 작업 마이그레이션을 참조하세요.

개발 엔드포인트에 액세스하는 데 사용하는 방법은 무엇입니까?

사용되는 방법	수행할 작업
SageMaker AI 노트북, Jupyter Notebook 또는 JupyterLab	Jupyter에서 `.ipynb` 파일을 다운로드하여 AWS Glue Studio 노트북으로 마이그레이션하고 `.ipynb` 파일을 업로드하여 새 AWS Glue Studio 노트북 작업을 생성합니다. 또는 SageMaker AI Studio를 사용하고 AWS Glue 커널을 선택할 수도 있습니다.
Zeppelin 노트북	코드를 복사하여 붙여 넣거나 ze2nb와 같은 타사 변환기를 자동으로 사용하여 노트북을 Jupyter Notebook으로 수동으로 변환할 수 있습니다. 그런 다음 AWS Glue Studio 노트북 또는 SageMaker AI Studio에서 노트북을 사용합니다.
IDE	PyCharm에서 AWS Glue 대화형 세션을 사용하여 AWS Glue 작업 작성 또는 Microsoft Visual Studio 코드로 대화형 세션 사용을 참조하세요.
REPL	`aws-glue-session package` 를 로컬로 설치한 후 다음 명령을 실행합니다. • Python의 경우: `jupyter console --kernal glue_pyspark` • Scala의 경우: `jupyter console --kernal glue_spark`
SSH	대화형 세션에는 해당 옵션이 없습니다. 또는 도커 이미지를 사용할 수 있습니다. 자세한 내용은 도커 이미지를 사용하여 개발을 참조하세요.

다음 단원에서는 dev 엔드포인트를 사용하여 AWS Glue 버전 1.0에서 작업을 개발하는 방법에 대한 정보를 제공합니다.

주제

- 개발 엔드포인트를 사용하여 스크립트 개발

- 노트북 관리

개발 엔드포인트를 사용하여 스크립트 개발

> **Note**
>
> 개발 엔드포인트는 AWS Glue의 2.0 이전 버전에서만 지원됩니다. ETL 스크립트를 작성하고 테스트할 수 있는 대화형 환경에서는 AWS Glue Studio에서 노트북을 사용합니다.

AWS Glue는 추출, 변환 및 로드(ETL) 스크립트를 반복적으로 개발하고 테스트할 수 있는 개발 엔드포인트라는 환경을 생성할 수 있습니다. AWS Glue 콘솔 또는 API를 사용하여 개발 엔드포인트를 만들고, 편집하고, 삭제할 수 있습니다.

개발 환경 관리

개발 엔드포인트를 만들 때 구성 값을 제공하여 개발 환경을 제공합니다. 이 값은 AWS Glue가 어떻게 네트워크를 설치하는지 알려줘 안전하게 엔드포인트로 액세스할 수 있고 엔드포인트는 데이터 스토어에 액세스할 수 있습니다.

그런 다음, 엔드포인트로 연결되는 노트북을 만들고 이를 사용하여 ETL 스크립트를 작성 및 테스트할 수 있습니다. 개발 절차에 따른 결과를 만족하면 스크립트를 실행하는 ETL 작업을 생성합니다. 이 과정을 통해 함수를 추가하고 상호적으로 스크립트를 디버깅할 수 있습니다.

이 섹션의 자습서에 따라 노트북에서 개발 엔드포인트를 사용하는 방법을 배우십시오.

주제

- 개발 엔드포인트 워크플로
- AWS Glue 개발 엔드포인트가 SageMaker 노트북과 함께 작동하는 방식
- 개발 엔드포인트 추가
- 개발 엔드포인트 액세스
- 자습서: JupyterLab에 Jupyter Notebook을 설정하여 ETL 스크립트 테스트 및 디버깅
- 자습서: 개발 엔드포인트와 SageMaker AI 노트북 함께 사용
- 자습서: 개발 엔드포인트를 통해 REPL 셸 사용하기
- 자습서: 개발 엔드포인트로 PyCharm Professional 설치하기
- 고급 구성: 여러 사용자 간에 개발 엔드포인트 공유

개발 엔드포인트 워크플로

다음 절차에 따라 AWS Glue 개발 엔드포인트를 사용합니다.

1. API를 사용하여 개발 엔드포인트를 생성합니다. 이 엔드포인트를 정의한 보안 그룹으로 Virtual Private Cloud(VPC)에서 시작합니다.
2. 개발 엔드포인트가 프로비저닝되어 작업을 시작할 준비가 된 상태가 될 때까지 API를 사용하여 개발 엔드포인트를 폴링합니다. 준비가 되었다면, 다음 방법 중 하나를 사용해 개발 엔드포인트로 연결하여 AWS Glue 스크립트를 작성 및 테스트할 수 있습니다.
   - 계정에서 SageMaker AI 노트북을 생성합니다. 노트북을 생성하는 방법에 대한 자세한 내용은 the section called “AWS Glue Studio 노트북을 사용한 코드 작성” 단원을 참조하십시오.
   - 터미널 창을 열어 바로 개발 엔드포인트로 연결합니다.
   - JetBrains PyCharm Python IDE의 전문가 버전이 있다면 개발 엔드포인트로 연결하고 이것을 사용하여 상호적으로 개발합니다. 스크립트에 `pydevd` 문을 입력하면 PyCharm은 원격 중단점을 지원할 수 있습니다.
3. 개발 엔드포인트의 디버깅과 테스트가 완료된다면 이것을 삭제할 수 있습니다.

AWS Glue 개발 엔드포인트가 SageMaker 노트북과 함께 작동하는 방식

개발 엔드포인트에 액세스하는 보편적인 방법 중 하나로 SageMaker에서 Jupyter를 사용하는 방법이 있습니다. Jupyter Notebook은 오픈 소스 웹 애플리케이션의 일종으로 시각화, 분석, 기계 학습 등에 폭넓게 사용됩니다. AWS Glue SageMaker 노트북이 AWS Glue 개발 엔드포인트를 포함한 Jupyter Notebook 환경을 제공합니다. AWS Glue SageMaker 노트북에서 Jupyter Notebook 환경은 SparkMagic으로 사전 구성되어 있습니다. 이것은 오픈 소스 Jupyter 플러그인으로 Spark 작업을 원격 Spark 클러스터에 제출하는 역할을 합니다. Apache Livy는 REST API를 통해 원격 Spark 클러스터와 상호작용을 주고받게 해주는 서비스입니다. AWS Glue SageMaker 노트북에 SparkMagic이 구성되어 있어 AWS Glue 개발 엔드포인트에서 실행되는 Livy 서버에 대하여 REST API를 호출합니다.

다음 텍스트 흐름은 각각의 구성 요소의 작동 방식을 설명한 것입니다.

AWS Glue SageMaker 노트북: (Jupyter → SparkMagic) → (네트워크) → AWS Glue 개발 엔드포인트: (Apache Livy → Apache Spark)

Jupyter Notebook의 각 단락에 쓰인 Spark 스크립트를 실행하면 해당 Spark 코드를 SparkMagic을 통해 Livy 서버로 제출하고, 그런 다음 "livy-session-N"이라는 이름의 Spark 작업이 Spark 클러스터에서 실행됩니다. 이 작업을 Livy 세션이라고 합니다. 이 Spark 작업은 노트북 세션이 실행 중인 동안 실행됩

니다. 노트북에서 Jupyter 커널을 종료하거나 세션이 시간 초과되면 해당 Spark 작업이 종료됩니다. 노트북(.ipnyb) 파일당 Spark 작업이 하나씩 실행됩니다.

여러 개의 SageMaker 노트북 인스턴스와 단 한 개의 AWS Glue 개발 엔드포인트만 사용할 수 있습니다. 각각의 SageMaker 노트북 인스턴스에 노트북 파일을 여러 개 생성할 수 있습니다. 각각의 노트북 파일을 열고 단락을 실행하면 SparkMagic을 통해 Spark 클러스터의 노트북 파일당 Livy 세션이 하나씩 실행됩니다. Livy 세션은 각각 Spark 작업 한 개씩에 상응합니다.

AWS Glue 개발 엔드포인트 및 SageMaker 노트북의 기본 동작

Spark 작업은 Spark 구성에 따라 실행됩니다. Spark 구성은 여러 가지 방식으로 설정할 수 있습니다(예: Spark 클러스터 구성, SparkMagic 구성 등).

기본적으로 Spark는 Spark 클러스터 구성에 따라 클러스터 리소스를 Livy 세션에 할당합니다. AWS Glue 개발 엔드포인트의 경우 클러스터 구성은 작업자 유형에 좌우됩니다. 다음은 작업자 유형에 따른 보편적인 구성을 설명한 테이블입니다.

	표준	G.1X	G.2X
`spark.driver.memory`	5G	10G	20G
`spark.executor.memory`	5G	10G	20G
`spark.executor.cores`	4	8	16
`spark.dynamicAllocation.enabled`	TRUE	TRUE	TRUE

Spark 실행기 최대 수는 DPU(또는 `NumberOfWorkers`)와 작업자 유형을 조합하여 자동으로 계산됩니다.

	표준	G.1X	G.2X
Spark 실행기 최대 수	`(DPU - 1) * 2 - 1`	`(NumberOfWorkers - 1)`	`(NumberOfWorkers - 1)`

예를 들어 개발 엔드포인트에 작업자가 10개이고 작업자 유형이 `G.1X`인 경우, Spark 실행기는 9개이고 클러스터 전체에는 실행기 메모리가 90G가 됩니다. 실행기마다 메모리가 10G씩이 때문입니다.

지정된 작업자 유형과 관계없이 Spark 동적 리소스 할당이 활성화됩니다. 데이터 집합이 충분히 크면 Spark가 모든 실행기를 하나의 Livy 세션에 할당할 가능성이 있습니다. `spark.dynamicAllocation.maxExecutors`가 기본적으로 설정되어 있지 않기 때문입니다. 이는 같은 개발 엔드포인트상의 다른 Livy 세션은 새 실행기 실행을 대기하게 된다는 뜻입니다. 데이터 집합 크기가 작은 경우 Spark가 실행기를 동시에 여러 Livy 세션에 할당할 수 있습니다.

> **Note**
>
> 다양한 사용 사례에서 리소스 할당 방식을 자세히 알아보고 동작을 수정하기 위해 구성을 변경하는 법을 알아보려면 고급 구성: 여러 사용자 간에 개발 엔드포인트 공유 섹션을 참조하세요.

개발 엔드포인트 추가

AWS Glue에서 개발 엔드포인트를 사용하여 추출, 변환 및 로드(ETL) 스크립트를 반복적으로 개발하고 테스트합니다. 개발 엔드포인트 작업은 AWS Command Line Interface에서만 가능합니다.

1. 명령줄 창에 다음과 유사한 명령을 입력합니다.

   ```
   aws glue create-dev-endpoint --endpoint-name "endpoint1" --role-arn
    "arn:aws:iam::account-id:role/role-name" --number-of-nodes "3" --glue-version
    "1.0" --arguments '{"GLUE_PYTHON_VERSION": "3"}' --region "region-name"
   ```

   이 명령은 AWS Glue 버전 1.0을 지정합니다. 이 버전은 Python 2와 Python 3를 모두 지원하므로 `arguments` 파라미터를 사용하여 원하는 Python 버전을 표시할 수 있습니다. `glue-version` 파라미터가 생략된 경우, AWS Glue 버전 0.9로 간주됩니다. AWS Glue 버전에 대한 자세한 내용은 Glue version job property 단원을 참조하십시오.

   추가적인 명령줄 파라미터에 대한 자세한 내용은 AWS CLI Command Reference의 create-dev-endpoint를 참조하세요.

2. (선택 사항) 개발 엔드포인트 상태를 확인하려면 다음 명령을 입력하십시오. 상태가 READY로 변경되면 개발 엔드포인트를 사용할 준비가 된 것입니다.

```
aws glue get-dev-endpoint --endpoint-name "endpoint1"
```

개발 엔드포인트 액세스

Virtual Private Cloud(VPC)에서 개발 엔드포인트를 생성하면, AWS Glue는 프라이빗 IP 주소만 반환합니다. 퍼블릭 IP 주소 필드는 채워지지 않습니다. 비 VPC 개발 엔드포인트를 생성할 때 AWS Glue은(는) 퍼블릭 IP 주소만 반환합니다.

개발 엔드포인트가 퍼블릭 주소가 있다면 다음 예와 같이 개발 엔드포인트의 SSH 프라이빗 키를 통해 접근 가능하다는 것을 확인합니다.

```
ssh -i dev-endpoint-private-key.pem glue@public-address
```

개발 엔드포인트에 프라이빗 주소가 있고 VPC 서브넷이 퍼블릭 인터넷에서 라우팅 가능하며 보안 그룹이 클라이언트의 인바운드 액세스를 허용한다고 가정하십시오. 이 경우, 다음 단계에 따라 탄력적 IP 주소를 개발 엔드포인트에 연결하여 인터넷 액세스를 허용하십시오.

> **Note**
>
> 탄력적 IP 주소를 사용하려면 사용 중인 서브넷은 라우팅 테이블을 통해 연결되는 인터넷 게이트웨이가 있어야 합니다.

탄력적 IP 주소를 연결하여 개발 엔드포인트에 액세스하려면

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 개발 엔드포인트를 선택하고 개발 엔드포인트 세부 정보 페이지로 이동하십시오. 다음 단계에서 사용할 수 있도록 [Private address(프라이빗 주소)]를 기록합니다.
3. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
4. 탐색 창의 Network & Security(네트워크 및 보안) 아래에서 Network Interfaces(네트워크 인터페이스)를 선택합니다.
5. AWS Glue 콘솔 개발 엔드포인트 세부 정보 페이지의 [프라이빗 주소(Private address)]에 상응하는 [프라이빗 DNS(IPv4)(Private DNS (IPv4))]를 검색합니다.

Amazon EC2 콘솔에 표시할 열을 수정해야 할 수 있습니다. 네트워크 인터페이스 ID(ENI)(예: `eni-12345678`)는 이 주소를 위한 것이라고 알아야 합니다.

6. Amazon EC2 콘솔의 [네트워크 및 보안(Network & Security)] 아래에서 [탄력적 IP(Elastic IPs)]를 선택합니다.
7. 새 주소 할당과 할당 순으로 선택하여 새로운 탄력적 IP 주소를 할당합니다.
8. [Elastic IPs(탄력적 IP)] 페이지에서 새롭게 할당된 [Elastic IP(탄력적 IP)]를 선택합니다. 그런 다음 [Actions(작업)]을 선택한 후 Associate address(관련 주소)를 선택합니다.
9. 주소 연결 페이지에서 다음과 같이 진행합니다.

   - [Resource type(리소스 유형)]에서 Network interface(네트워크 인터페이스)을 선택합니다.
   - 네트워크 인터페이스 상자에 프라이빗 주소를 위한 Network interface ID(네트워크 인터페이스 ID)(ENI)를 입력합니다.
   - 연결을 선택합니다.
10. 다음 예와 같이, 새롭게 연결된 탄력적 IP 주소가 개발 엔드포인트의 SSH 프라이빗 키를 통해 접근 가능하다면 인증합니다.

```
ssh -i dev-endpoint-private-key.pem glue@elastic-ip
```

Bastion Host를 이용해 개발 엔드포인트의 프라이빗 주소에 대한 SSH 액세스를 가져오는 데 대한 자세한 내용은 AWS 보안 블로그 게시물 Securely Connect to Linux Instances Running in a Private Amazon VPC를 참조하세요.

자습서: JupyterLab에 Jupyter Notebook을 설정하여 ETL 스크립트 테스트 및 디버깅

이 튜토리얼에서는 로컬 시스템에서 실행 중인 JupyterLab의 Jupyter Notebook을 개발 엔드포인트에 연결합니다. 이렇게 하면 AWS Glue 추출, 변환, 로드(ETL) 스크립트를 배포하기에 앞서 대화식으로 이를 실행, 디버깅 및 테스트할 수 있습니다. 이 자습서에서는 Secure Shell(SSH) 전송을 사용하여 로컬 시스템을 AWS Glue 개발 엔드포인트에 연결합니다. 자세한 내용은 Wikipedia의 Port forwarding을 참조하세요.

1단계: JupyterLab과 Sparkmagic 설치

JupyterLab을 설치하려면 `conda` 또는 `pip`를 사용하면 됩니다. `conda`는 오픈 소스 패키지 관리 시스템 겸 환경 관리 시스템으로 Windows, macOS 및 Linux에서 실행됩니다. `pip`는 Python용 패키지 설치 프로그램입니다.

macOS에 설치하는 경우, Xcode가 설치되어 있어야 Sparkmagic을 설치할 수 있습니다.

1. JupyterLab, Sparkmagic과 관련 확장 프로그램을 설치합니다.

```
$ conda install -c conda-forge jupyterlab
$ pip install sparkmagic
$ jupyter nbextension enable --py --sys-prefix widgetsnbextension
$ jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

2. Location에서 sparkmagic 디렉터리를 확인합니다.

```
$ pip show sparkmagic | grep Location
Location: /Users/username/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-
packages
```

3. 디렉터리를 Location에 대하여 반환된 것으로 변경하고, Scala와 PySpark의 커널을 설치합니다.

```
$ cd /Users/username/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages
$ jupyter-kernelspec install sparkmagic/kernels/sparkkernel
$ jupyter-kernelspec install sparkmagic/kernels/pysparkkernel
```

4. 샘플 config 파일을 다운로드합니다.

```
$ curl -o ~/.sparkmagic/config.json https://raw.githubusercontent.com/jupyter-
incubator/sparkmagic/master/sparkmagic/example_config.json
```

   이 구성 파일에서 driverMemory 및 executorCores와 같은 Spark 관련 파라미터를 구성할 수 있습니다.

2단계: JupyterLab 시작

JupyterLab을 시작하면 기본 웹 브라우저가 자동으로 열리고 URL http://localhost:8888/lab/workspaces/{workspace_name}이 표시됩니다.

```
$ jupyter lab
```

3단계: SSH 포트 전송을 시작하여 개발 엔드포인트에 연결

다음으로, SSH 로컬 포트 전송을 사용하여 로컬 포트(여기서는 8998)를 AWS Glue(169.254.76.1:8998)에 의해 정의된 원격 대상으로 전송합니다.

1. SSH에 액세스를 부여하는 별도의 터미널 창을 엽니다. Microsoft Windows에서라면 Git for Windows에서 제공하는 BASH 셸을 사용해도 되고, 아니면 Cygwin을 설치해도 됩니다.
2. 다음과 같이 수정된 다음 SSH 명령을 실행합니다.

   - *private-key-file-path* 대신 개발 엔드포인트를 생성하는 데 쓴 퍼블릭 키에 상응하는 프라이빗 키를 포함한 .pem 파일에 대한 경로를 사용합니다.
   - 8998 외의 다른 포트를 전송하는 경우, 8998 대신 로컬에서 실제로 사용 중인 포트 번호를 사용합니다. 주소 169.254.76.1:8998이 원격 포트이며 이는 사용자가 변경하는 것이 아닙니다.
   - *dev-endpoint-public-dns*를 개발 엔드포인트의 퍼블릭 DNS 주소로 바꿉니다. 이 주소를 찾으려면 AWS Glue 콘솔에서 개발 엔드포인트로 이동하여 이름을 선택하고 [엔드포인트 세부 정보(Endpoint details)] 페이지에 목록으로 기재된 [퍼블릭 주소(Public address)]를 복사합니다.

   ```
   ssh -i private-key-file-path -NTL 8998:169.254.76.1:8998 glue@dev-endpoint-public-dns
   ```

   다음과 같은 경고 메세지가 표시됩니다.

   ```
   The authenticity of host 'ec2-xx-xxx-xxx-xx.us-west-2.compute.amazonaws.com
    (xx.xxx.xxx.xx)'
   can't be established.  ECDSA key fingerprint is SHA256:4e97875Brt+1wKzRko
   +JflSnp21X7aTP3BcFnHYLEts.
   Are you sure you want to continue connecting (yes/no)?
   ```

   yes를 입력하고 JupyterLab을 사용하는 동안 터미널 창을 열어둡니다.
3. SSH 포트 전송이 개발 엔드포인트와 올바로 함께 작동하는지 확인합니다.

   ```
   $ curl localhost:8998/sessions
   {"from":0,"total":0,"sessions":[]}
   ```

4단계: Notebook Paragraph의 Simple Script Fragment 실행

이제 JupyterLab의 노트북이 개발 엔드포인트와 함께 작동하는 것이 정상입니다. 다음과 같은 스크립트 조각을 노트북에 입력하여 실행합니다.

1. Spark가 실행 중인지 확인합니다. 다음 명령을 사용하면 Spark에 1을 계산하여 값을 인쇄하도록 지시합니다.

   ```
   spark.sql("select 1").show()
   ```

2. AWS Glue Data Catalog 통합이 작동하는지 확인합니다. 다음 명령을 사용하면 테이블을 데이터 카탈로그에 목록으로 나열합니다.

   ```
   spark.sql("show tables").show()
   ```

3. AWS Glue 라이브러리를 사용하는 단순한 스크립트 조각이 작동하는지 확인합니다.

   다음 스크립트는 AWS Glue Data Catalog의 `persons_json` 테이블 메타데이터를 사용하여 샘플 데이터로부터 `DynamicFrame`을 생성합니다. 그런 다음 항목 수와 이 데이터 스키마를 출력합니다.

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create a Glue context
glueContext = GlueContext(SparkContext.getOrCreate())

# Create a DynamicFrame using the 'persons_json' table
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")

# Print out information about *this* data
print("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

스크립트의 출력은 다음과 같습니다.

```
 Count:  1961
 root
```

```
 |-- family_name: string
 |-- name: string
 |-- links: array
 |    |-- element: struct
 |    |    |-- note: string
 |    |    |-- url: string
 |-- gender: string
 |-- image: string
 |-- identifiers: array
 |    |-- element: struct
 |    |    |-- scheme: string
 |    |    |-- identifier: string
 |-- other_names: array
 |    |-- element: struct
 |    |    |-- note: string
 |    |    |-- name: string
 |    |    |-- lang: string
 |-- sort_name: string
 |-- images: array
 |    |-- element: struct
 |    |    |-- url: string
 |-- given_name: string
 |-- birth_date: string
 |-- id: string
 |-- contact_details: array
 |    |-- element: struct
 |    |    |-- type: string
 |    |    |-- value: string
 |-- death_date: string
```

문제 해결

- JupyterLab 설치 중에 컴퓨터가 회사 프록시나 방화벽 뒤에 위치한 경우, 회사 IT 부서에서 관리하는 사용자 정의 보안 프로파일로 인해 HTTP 및 SSL 오류가 발생할 수 있습니다.

  다음은 conda가 자체 리포지토리에 연결할 수 없을 때 발생하는 일반적인 오류의 예시입니다.

  ```
  CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://repo.anaconda.com/pkgs/
  main/win-64/current_repodata.json>
  ```

이것은 회사에서 Python 및 JavaScript 커뮤니티에서 광범위하게 사용되는 리포지토리로의 연결을 차단할 수 있기 때문에 발생할 가능성이 있습니다. 자세한 내용은 JupyterLab 웹 사이트의 Installation Problems를 참조하세요.

- 개발 엔드포인트에 연결을 시도할 때 [연결 거부됨(connection refused)] 오류가 발생하는 경우 날짜가 지난 개발 엔드포인트를 사용하고 있는 것일 수 있습니다. 새로운 개발 엔드포인트를 생성해보고 다시 연결하십시오.

자습서: 개발 엔드포인트와 SageMaker AI 노트북 함께 사용

AWS Glue에서는 개발 엔드포인트를 생성하고 SageMaker AI 노트북을 생성하여 ETL과 기계 학습 스크립트 개발에 도움을 얻을 수 있습니다. SageMaker AI 노트북은 Jupyter Notebook 애플리케이션을 실행하는 완전 관리형 기계 학습 컴퓨팅 인스턴스입니다.

1. AWS Glue 콘솔에서 Dev endpoints(개발 엔드포인트)를 선택하여 개발 엔드포인트 목록을 탐색합니다.
2. 사용할 개발 엔드포인트의 이름 옆에 있는 확인란을 선택하고 작업 메뉴에서 SageMaker 노트북 생성을 선택합니다.
3. 다음과 같이 노트북 생성 및 구성 페이지를 작성합니다.

   a. 노트북 이름을 입력합니다.

   b. 개발 엔드포인트에 연결에서 개발 엔드포인트를 확인합니다.

   c. AWS Identity and Access Management(IAM) 역할을 생성하거나 선택합니다.

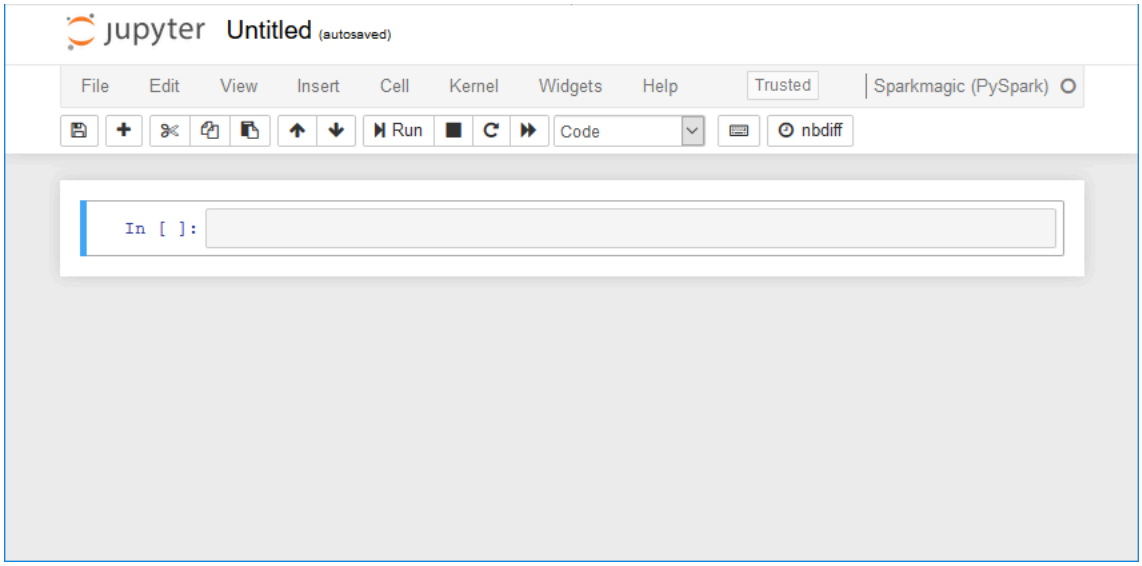
      역할을 생성하는 것이 좋습니다. 기존 역할을 사용하는 경우 필요한 권한이 있는지 확인합니다. 자세한 내용은 the section called “6단계: SageMaker AI 노트북용 IAM 정책 생성” 섹션을 참조하세요.

   d. (선택 사항) VPC, 서브넷 및 하나 이상의 보안 그룹을 선택합니다.

   e. (선택 사항) AWS Key Management Service 암호화 키를 선택합니다.

   f. (선택 사항) 노트북 인스턴스에 대한 태그를 추가합니다.
4. 노트북 생성을 선택합니다. 노트북 페이지의 오른쪽 상단에 있는 새로 고침 아이콘을 선택하고 상태가 `Ready`로 표시될 때까지 계속합니다.
5. 새 노트북 이름 옆의 확인란을 선택한 다음 노트북 열기를 선택합니다.
6. 새 노트북 생성: jupyter 페이지에서 신규를 선택한 다음 Sparkmagic (PySpark)을 선택합니다.

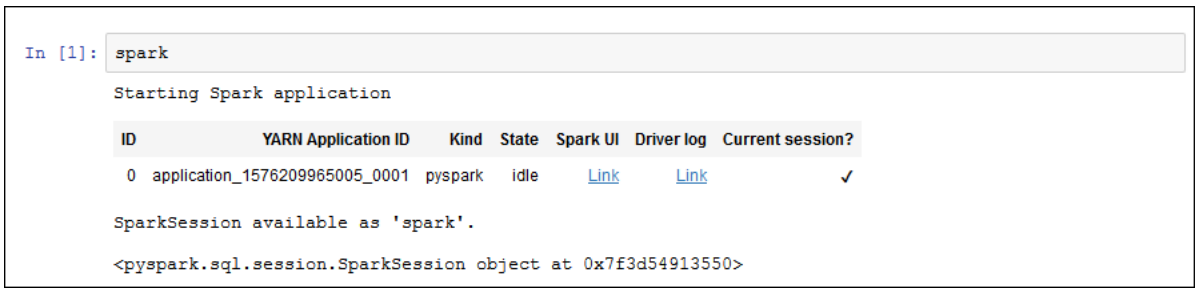
   이제 화면이 다음과 같이 보여야 합니다.



7. (선택 사항) 페이지 상단에서 Untitled를 선택하고 노트북에 이름을 지정합니다.
8. Spark 애플리케이션을 시작하려면 노트북에 다음 명령을 입력한 다음 도구 모음에서 실행을 선택합니다.

```
spark
```

잠시 후 다음과 같은 응답이 표시됩니다.



9. 동적 프레임을 생성하고 쿼리를 실행합니다. 복사, 붙여넣기 및 persons_json 테이블의 개수 및 스키마를 출력하는 다음 코드를 실행합니다.

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
glueContext = GlueContext(SparkContext.getOrCreate())
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")
print ("Count:  ", persons_DyF.count())
```

```
persons_DyF.printSchema()
```

자습서: 개발 엔드포인트를 통해 REPL 셸 사용하기

AWS Glue를 통해 개발 엔드포인트를 만들고 REPL(Read-Evaluate-Print Loop) 셸을 호출하여 점차 늘리면서 PySpark 코드를 실행합니다. 그러면 ETL 스크립트를 디버깅하기 전에 상호적으로 디버깅할 수 있습니다.

개발 엔드포인트에서 REPL을 사용하려면 엔드포인트에 대한 SSH에 권한을 부여해야 합니다.

1. 로컬 컴퓨터에서 SSH 명령어를 실행할 수 있는 터미널창을 열고 편집한 SSH 명령어에 붙여 넣습니다. 명령을 실행합니다.

   개발 엔드포인트에 대해 Python 3을 사용하는 AWS Glue 버전 1.0을 수락했다고 가정하면 출력은 다음과 같습니다.

   ```
   Python 3.6.8 (default, Aug  2 2019, 17:42:44)
   [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
   Type "help", "copyright", "credits" or "license" for more information.
   SLF4J: Class path contains multiple SLF4J bindings.
   SLF4J: Found binding in [jar:file:/usr/share/aws/glue/etl/jars/glue-assembly.jar!/
   org/slf4j/impl/StaticLoggerBinder.class]
   SLF4J: Found binding in [jar:file:/usr/lib/spark/jars/slf4j-log4j12-1.7.16.jar!/
   org/slf4j/impl/StaticLoggerBinder.class]
   SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
   SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
   Setting default log level to "WARN".
   To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
    setLogLevel(newLevel).
   2019-09-23 22:12:23,071 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
    - Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading
    libraries under SPARK_HOME.
   2019-09-23 22:12:26,562 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
    - Same name resource file:/usr/lib/spark/python/lib/pyspark.zip added multiple
    times to distributed cache
   2019-09-23 22:12:26,580 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
    - Same path resource file:///usr/share/aws/glue/etl/python/PyGlue.zip added
    multiple times to distributed cache.
   2019-09-23 22:12:26,581 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
    - Same path resource file:///usr/lib/spark/python/lib/py4j-src.zip added multiple
    times to distributed cache.
   ```

```
2019-09-23 22:12:26,581 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same path resource file:///usr/share/aws/glue/libs/pyspark.zip added multiple
 times to distributed cache.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.3
      /_/

Using Python version 3.6.8 (default, Aug  2 2019 17:42:44)
SparkSession available as 'spark'.
>>>
```

2. `print(spark.version)`을 입력하여 REPL 셀이 올바르게 작동하는지 테스트합니다. Spark 버전을 나타내는 한 REPL은 사용할 준비되어 있습니다.
3. 셀에서 다음 샘플 스크립트를 행마다 실행하고자 합니다.

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
glueContext = GlueContext(SparkContext.getOrCreate())
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")
print ("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

자습서: 개발 엔드포인트로 PyCharm Professional 설치하기

이 자습서를 통해 로컬 시스템에서 실행되는 PyCharm Professional Python IDE를 개발 엔드포인트로 연결하여 AWS Glue ETL(추출, 변환 및 로드) 스크립트를 배치하기 전에 상호적으로 실행, 디버깅, 및 테스트할 수 있습니다. 자습서의 지침 및 화면 캡처는 PyCharm Professional 버전 2019.3을 기반으로 합니다.

상호적으로 개발 엔드포인트를 연결하기 위해서 PyCharm Professional이 설치되어 있어야 합니다. 무료 에디션을 사용하면 아무것도 할 수 없습니다.

> **Note**
>
> 이 튜토리얼에서는 Amazon S3를 데이터 원본으로 사용합니다. 대신 JDBC 데이터 원본을 사용하려면 Virtual Private Cloud(VPC)에서 개발 엔드포인트를 실행해야 합니다. SSH를 사용하여 VPC의 개발 엔드포인트에 연결하려면 SSH 터널을 생성해야 합니다. 이 자습서에는 SSH 터널을 생성하는 지침이 포함되어 있지 않습니다. SSH를 사용하여 VPC의 개발 엔드포인트에 연결하는 방법에 대한 자세한 내용은 AWS 보안 블로그의 Securely Connect to Linux Instances Running in a Private Amazon VPC를 참조하세요.

주제

- PyCharm Professional을 개발 엔드포인트에 연결
- 스크린을 개발 엔드포인트에 배포
- 원격 인터프리터 구성
- 스크립트를 개발 엔드포인트에서 실행

PyCharm Professional을 개발 엔드포인트에 연결

1. `legislators`라는 PyCharm에서 새로운 순수 Python 프로젝트를 생성합니다.
2. `get_person_schema.py`라는 파일을 다음 내용으로 프로젝트에 생성합니다.

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


def main():
    # Create a Glue context
    glueContext = GlueContext(SparkContext.getOrCreate())

    # Create a DynamicFrame using the 'persons_json' table
    persons_DyF =
 glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")

    # Print out information about this data
    print("Count:  ", persons_DyF.count())
    persons_DyF.printSchema()
```

```
if __name__ == "__main__":
    main()
```

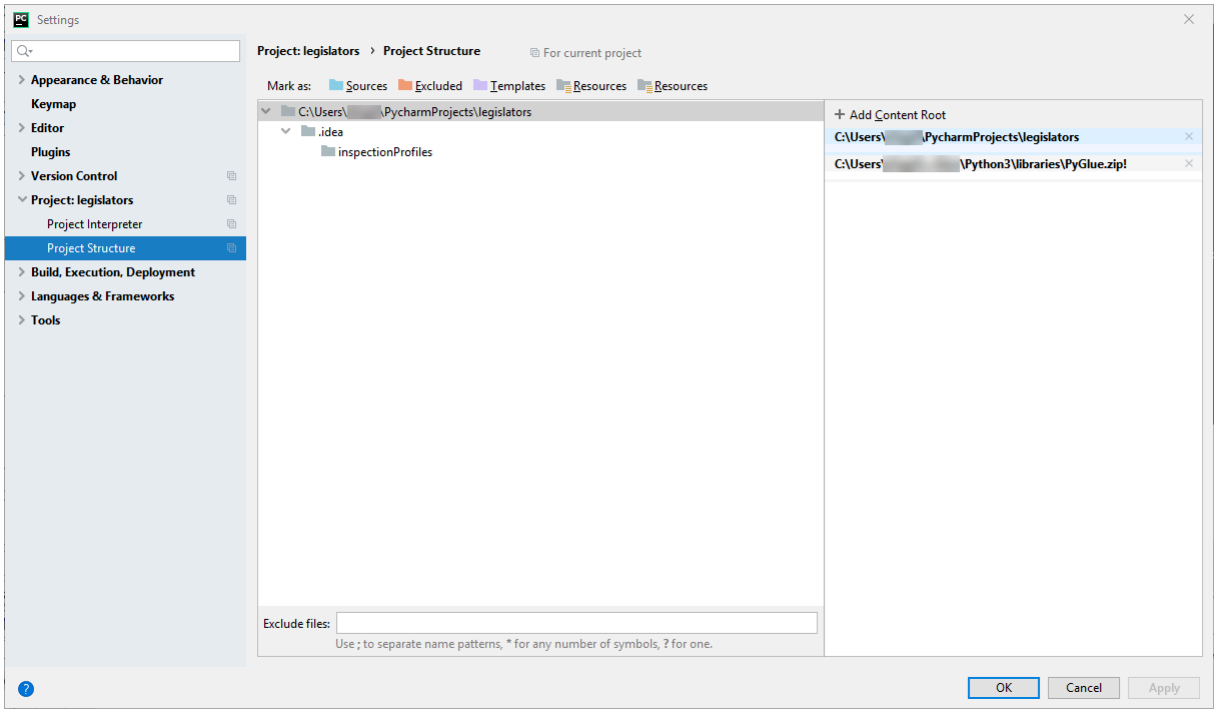
3. 다음 중 하나를 수행합니다.

   - AWS Glue 버전 0.9의 경우, AWS Glue Python 라이브러리 파일 `PyGlue.zip`을 `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl/python/PyGlue.zip`에서 로컬 시스템의 편리한 위치로 다운로드합니다.
   - AWS Glue 버전 1.0 이상의 경우, AWS Glue Python 라이브러리 파일 `PyGlue.zip`을 `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl-1.0/python/PyGlue.zip`에서 로컬 시스템의 편리한 위치로 다운로드합니다.

4. `PyGlue.zip`을 PyCharm의 프로젝트 내용 루트로써 추가합니다.

   - PyCharm에서 [Settings(설정)] 대화 상자를 열기 위해서 [File(파일)]과 [Settings(설정)]를 선택합니다. (`Ctrl+Alt+S`를 눌러도 됩니다.)
   - `legislators` 프로젝트를 확장하고 Project Structure(프로젝트 구조)를 선택합니다. 오른쪽 창에서 [+ Add Content Root(내용 루트 추가)]를 선택합니다.
   - `PyGlue.zip`을 저장한 위치로 이동하여 선택한 다음 [Apply(적용)]를 선택합니다.

   [Settings(설정)] 스크린은 다음과 비슷해야 합니다.



   [적용(Apply)]을 선택한 다음 [설정(Settings)] 대화 상자를 열어 둡니다.

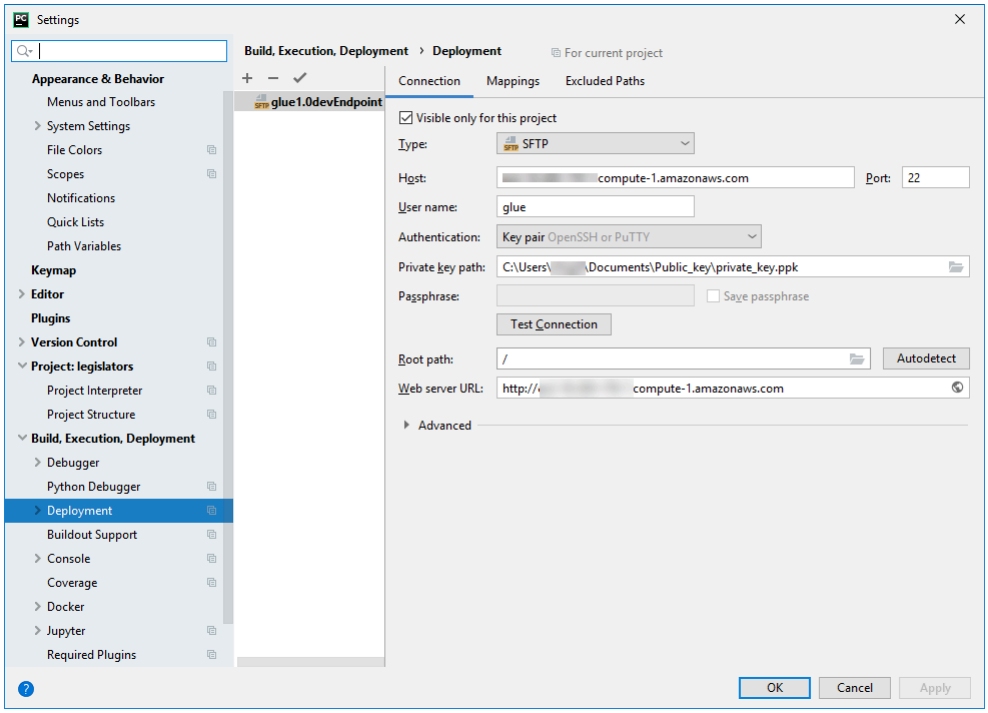
5. 개발 옵션을 구성하여 로컬 스크립트를 SFTP(이 기능은 PyCharm Professional에서만 사용 가능합니다)를 사용하여 개발 엔드포인트로 업로드합니다.

   - [Settings(설정)] 대화 상자에서 [Build, Execution, Deployment(설계, 실행, 빌)] 섹션을 확장합니다. Deployment(배치) 부 섹션을 선택합니다.
   - 중간 창의 맨 윗쪽에서 [+] 아이콘을 선택하여 새로운 서버를 추가합니다. 유형을 SFTP로 설정하고 이름을 지정합니다.
   - SFTP 호스트를 세부 정보 페이지에 나열된 대로 개발 엔드포인트의 퍼블릭 주소로 설정합니다. (AWS Glue 콘솔에서 개발 엔드포인트의 이름을 선택하여 세부 정보 페이지를 표시합니다.) VPC에서 실행 중인 개발 엔드포인트의 경우 SFTP 호스트를 호스트 주소로 설정하고 SSH 터널의 로컬 포트를 개발 엔드포인트로 설정합니다.
   - User name(사용자 이름)을 `glue`에 설치합니다.
   - Auth type(Auth 유형)을 Key pair (OpenSSH or Putty)(키 페어(OpenSSH 또는 Putty))에 설치합니다. 개발 엔드포인트 프라이빗 키 파일이 위치한 곳을 검색하여 [Private key file(프라이빗 키 파일)]을 설정합니다. PyCharm은 DSA, RSA 및 ECDSA OpenSSH 키 유형만 지원하므로, Putty의 프라이빗 형식의 키는 허용하지 않습니다. `ssh-keygen`의 업데이트 버전을 사용하여 PyCharm이 허용한 키 페어 유형을 다음과 같은 구문을 사용하여 생성합니다.

     ```
     ssh-keygen -t rsa -f <key_file_name> -C "<your_email_address>"
     ```

   - [Test connection(연결 테스트)]을 선택하고 연결을 테스트하도록 허용합니다. 연결이 성공하면 [Apply(적용)]를 선택합니다.

   [Settings(설정)] 스크린은 이제 다음과 비슷해야 합니다.

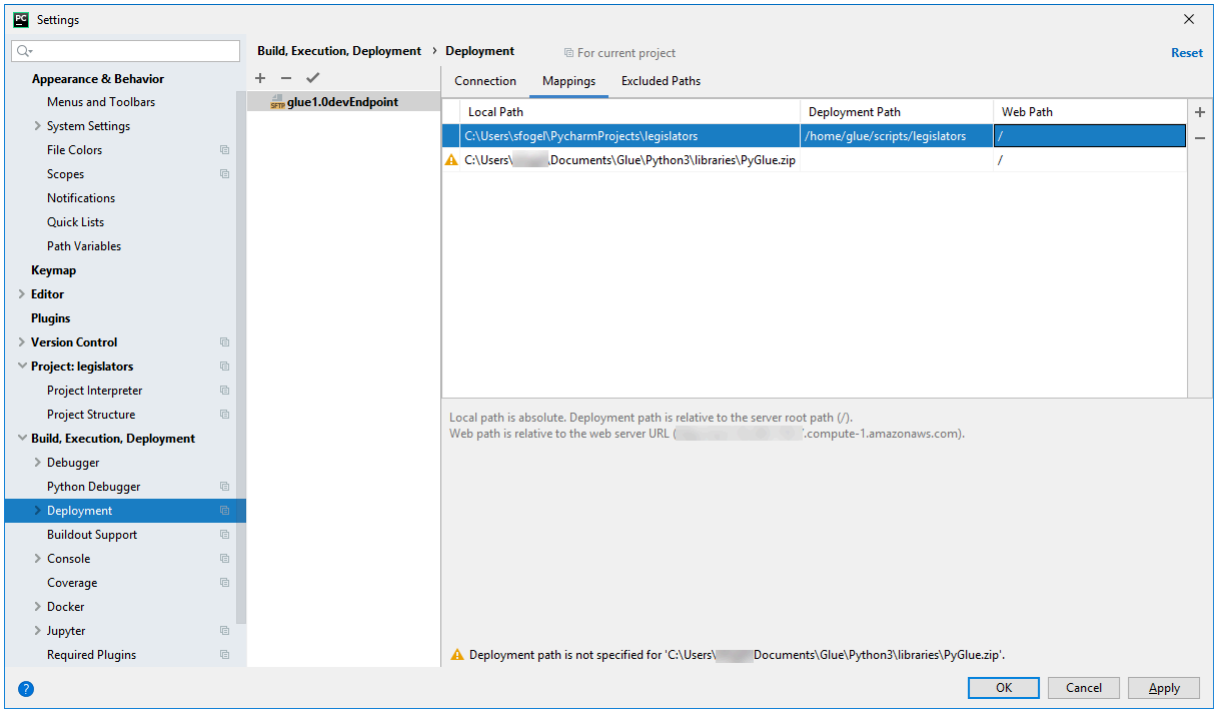


다시 한 번, [적용(Apply)]을 선택한 다음 [설정Settings)] 대화 상자를 열어 둡니다.

6. 로컬 디렉터리를 원격 디렉터리로 매핑하여 개발합니다.

   - [Deployment(배치)] 페이지의 오른쪽 화면에서 [Mappings(매핑)]d이라는 윗쪽 중간 탭을 선택합니다.
   - [Deployment Path(배치 경로)] 열에서 `/home/glue/scripts/`의 경로를 입력하여 프로젝트 경로를 배치합니다. 예: `/home/glue/scripts/legislators`.
   - 적용을 선택합니다.

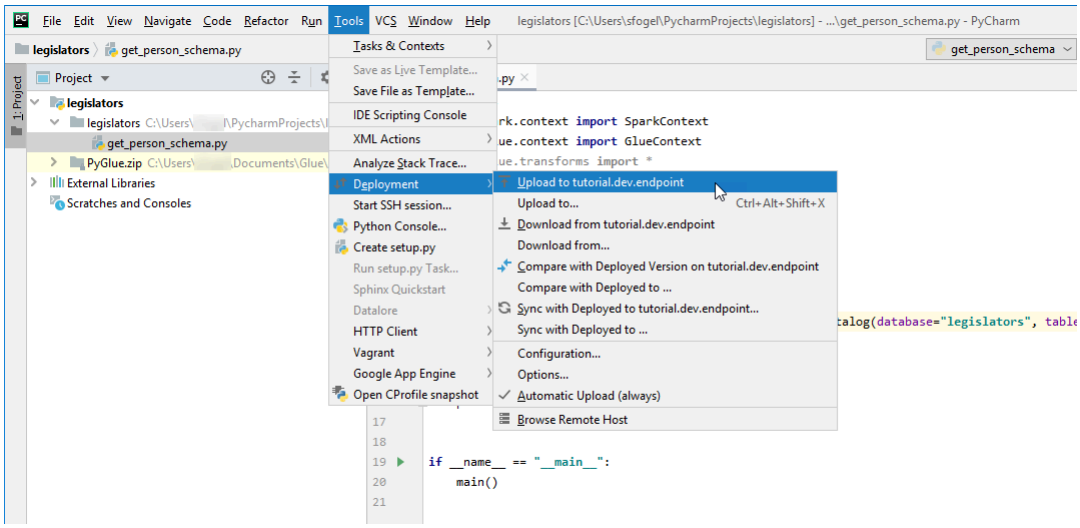
   [Settings(설정)] 스크린은 이제 다음과 비슷해야 합니다.



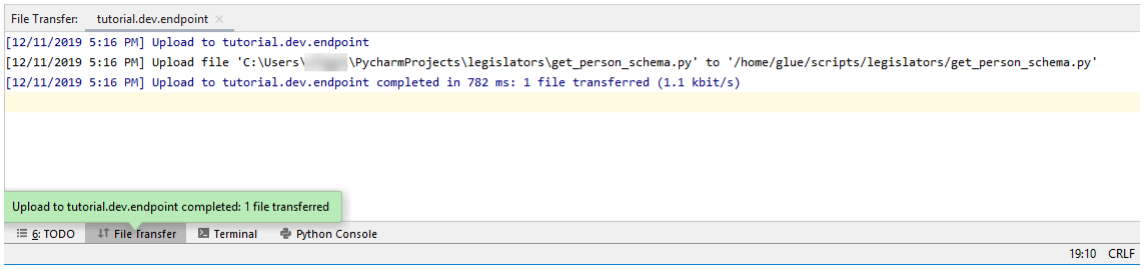
[OK(확인)]를 선택하여 [Settings(설정)] 대화 상자를 닫습니다.

스크린을 개발 엔드포인트에 배포

1. 다음 그림과 같이 [Tools(도구)], [Deployment(배포)]를 선택한 다음 개발 엔드포인트를 설정할 이름을 선택합니다.



스크린이 배치된 후 스크린 아래는 다음과 유사하게 보입니다.



2. 메뉴 모음에서 [Tools(도구)], [Deployment(배포)], [Automatic Upload (always)(자동 업데이트(항상))]를 선택합니다. [Automatic Upload (always)(자동 업데이트(항상))] 옆에 확인 표시가 나타나는지 확인합니다.

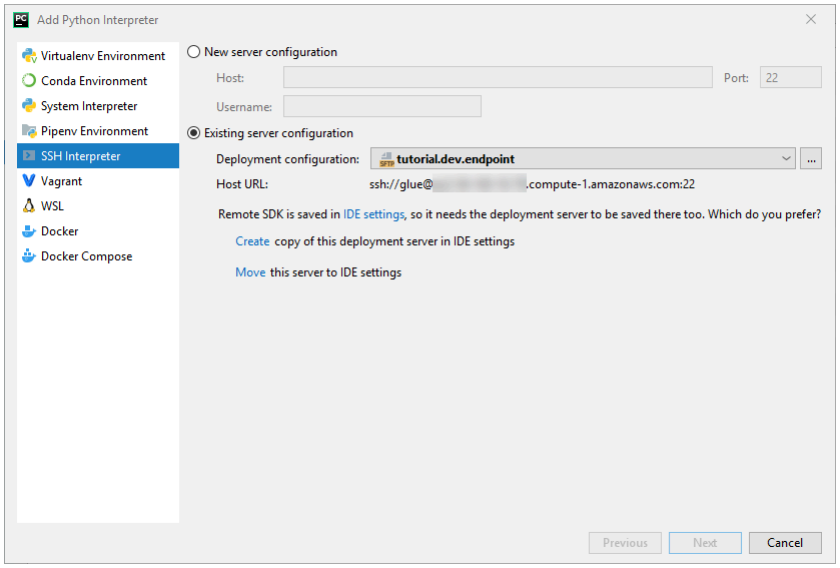
   이 옵션을 활성화하면 PyCharm은 변경된 파일을 개발 엔드포인트에 자동으로 업로드합니다.

원격 인터프리터 구성

개발 엔드포인트에서 Python 인터프리터를 사용하도록 PyCharm을 구성합니다.

1. [File(파일)] 메뉴에서 [Settings(설정)]를 선택합니다.
2. 프로젝트 [legislators(제정자)]를 확장하고 [Project Interpreter(프로젝트 인터프리터)]를 선택합니다.
3. [Project Interpreter(프로젝트 인터프리터)] 목록 옆에 있는 톱니바퀴 아이콘을 선택한 다음 [Add(추가)]를 선택합니다.
4. [Add Python Interpreter(Python 인터프리터 추가)] 대화 상자의 왼쪽 창에서 [SSH Interpreter(SSH 인터프리터)]를 선택합니다.
5. [Existing server configuration(기존 서버 구성)]을 선택하고 [Deployment configuration(배포 구성)] 목록에서 구성을 선택합니다.

   스크린은 다음과 비슷해야 합니다.



6. [Move this server to IDE settings(이 서버를 IDE 설정으로 이동)]를 선택하고 [Next(다음)]를 선택합니다.
7. [Interpreter(인터프리터)] 필드에서 Python 2를 사용하는 경우 경로를 `/usr/bin/gluepython`으로 변경하고 Python 3을 사용하는 경우 `/usr/bin/gluepython3`로 변경합니다. 그런 다음 Finish(완료)를 선택합니다.

스크립트를 개발 엔드포인트에서 실행

다음과 같이 스크립트를 실행합니다.

- 왼쪽 창에서 파일 이름을 마우스 오른쪽 버튼으로 클릭하고 [Run '***<filename>***'('<filename>' 실행)]을 선택합니다.

  일련의 메시지 후에 최종 출력에는 개수와 스키마가 표시되어야 합니다.

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
```

```
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string


Process finished with exit code 0
```

그런 다음 원격으로 개발 엔드포인트에 스크립트를 디버깅하도록 설정합니다.

고급 구성: 여러 사용자 간에 개발 엔드포인트 공유

이 섹션에서는 일반적인 사용 사례에서 SageMaker 노트북을 사용해 개발 엔드포인트를 유리하게 활용하여 여러 사용자 간에 개발 엔드포인트를 공유하는 법을 설명합니다.

단일 테넌시 구성

단일 테넌트 사용 사례의 경우, 개발자 환경을 간소화하고 리소스 경합을 피하려면 각각의 개발자가 자신이 작업하는 프로젝트에 맞춰 크기가 조정된 자기만의 개발 엔드포인트를 사용하도록 하는 것이 좋습니다. 이렇게 하면 작업자 유형 및 DPU 수와 관련된 결정도 간소화해줍니다. 개발자의 재량 및 개발자가 작업 중인 프로젝트에 따라 달라지도록 두기 때문입니다.

여러 노트북 파일을 동시에 실행하지 않는 한, 리소스 할당을 직접 처리하지 않아도 됩니다. 동시에 여러 노트북 파일에서 코드를 실행하는 경우, 여러 개의 Livy 세션을 동시에 실행해야 합니다. 여러 개의 Livy 세션을 동시에 실행하기 위해 Spark 클러스터 구성을 분리하려면 다중 테넌트 사용 사례에 소개된 단계를 따르면 됩니다.

예를 들어 개발 엔드포인트에 작업자가 10개이고 작업자 유형이 `G.1X`인 경우, Spark 실행기는 9개이고 클러스터 전체에는 실행기 메모리가 90G가 됩니다. 실행기마다 메모리가 10G씩이 때문입니다.

지정된 작업자 유형과 관계없이 Spark 동적 리소스 할당이 활성화됩니다. 데이터 집합이 충분히 크면 Spark가 모든 실행기를 하나의 Livy 세션에 할당할 가능성이 있습니다. `spark.dynamicAllocation.maxExecutors`가 기본적으로 설정되어 있지 않기 때문입니다. 이는 같은 개발 엔드포인트상의 다른 Livy 세션은 새 실행기 실행을 대기하게 된다는 뜻입니다. 데이터 집합 크기가 작은 경우 Spark가 실행기를 동시에 여러 Livy 세션에 할당할 수 있습니다.

> **Note**
>
> 다양한 사용 사례에서 리소스 할당 방식을 자세히 알아보고 동작을 수정하기 위해 구성을 변경하는 법을 알아보려면 고급 구성: 여러 사용자 간에 개발 엔드포인트 공유 섹션을 참조하세요.

다중 테넌시 구성

> **Note**
>
> 개발 엔드포인트는 AWS Glue ETL 환경에 단일 테넌트 환경으로 에뮬레이션되는 경향이 있다는 점을 알아두시기 바랍니다. 다중 테넌트 사용도 가능하기는 하지만, 이것은 고급 사용 사례이며 대부분의 사용자는 각각의 개발 엔드포인트에 단일 테넌시 패턴을 유지하는 것이 좋습니다.

다중 테넌트 사용 사례에서는 사용자가 리소스 할당을 직접 처리해야 할 가능성이 있습니다. 핵심 요인은 Jupyter Notebook을 동시에 사용하는 동시 사용자 수입니다. 'FTS(follow-the-sun)' 워크플로를 따르는 팀이고 각 시간대에 Jupyter 사용자가 한 명뿐인 경우, 동시 사용자 수가 하나뿐이므로 리소스 할당에 신경 쓰지 않아도 됩니다. 다만 노트북을 여러 사용자가 공유하고 각각의 사용자가 애드혹 방식으로 코드를 제출하는 경우라면, 아래와 같은 요점을 고려해야 할 수 있습니다.

Spark 클러스터 리소스를 여러 사용자 사이에서 파티셔닝하려면 SparkMagic 구성을 사용하면 됩니다. SparkMagic을 구성하는 방법은 두 가지입니다.

(A) %%configure -f 지시어 사용

노트북의 Livy 세션당 구성을 수정하고자 하는 경우, 노트북 단락에서 `%%configure -f` 지시어를 실행하면 됩니다.

예를 들어 5개의 실행기에서 Spark 애플리케이션을 실행하고자 하는 경우, 노트북 단락에서 다음과 같은 명령을 실행하면 됩니다.

```
%%configure -f
{"numExecutors":5}
```

그러면 Spark UI에 해당 작업에 대하여 실행기 5개만 실행되는 것을 확인할 수 있습니다.

동적인 리소스 할당을 위해서는 실행기 최대 수를 제한하는 것이 좋습니다.

```
%%configure -f
{"conf":{"spark.dynamicAllocation.maxExecutors":"5"}}
```

(B) SparkMagic Config 파일 수정

SparkMagic은 Livy API를 기반으로 작동합니다. SparkMagic은 `driverMemory`, `driverCores`, `executorMemory`, `executorCores`, `numExecutors`, `conf` 등의 구성으로 Livy 세션을 생성합니다. 이런 것이 Spark 클러스터 전체에서 리소스가 얼마나 사용되는지 판단하는 핵심 요인입니다. SparkMagic을 사용하면 Livy로 보내지는 그러한 파라미터를 지정할 Config 파일을 제공할 수 있습니다. 샘플 Config 파일은 이 Github 리포지토리에서 확인할 수 있습니다.

한 노트북의 모든 Livy 세션에 걸쳐 구성을 수정하고자 하는 경우, `/home/ec2-user/.sparkmagic/config.json`을 수정하여 `session_config`를 추가하도록 하면 됩니다.

SageMaker 노트북 인스턴스에서 Config 파일을 수정하려면 다음과 같은 단계를 따르면 됩니다.

1. SageMaker 노트북을 엽니다.
2. 터미널 커널을 엽니다.
3. 다음 명령을 실행합니다.

   ```
   sh-4.2$ cd .sparkmagic
   sh-4.2$ ls
   config.json logs
   sh-4.2$ sudo vim config.json
   ```

   예를 들어 이러한 줄을 `/home/ec2-user/.sparkmagic/config.json`에 추가한 다음 노트북에서 Jupyter 커널을 다시 시작하면 됩니다.

   ```
     "session_configs": {
   ```

```
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

지침 및 모범 사례

이런 종류의 리소스 충돌을 피하려면 다음과 같은 몇 가지 기본적인 접근법을 쓰면 됩니다.

- NumberOfWorkers(가로 방향으로 확장)를 늘리고 workerType(세로 방향으로 확장)을 업그레이드하여 Spark 클러스터 크기 키우기
- 사용자당 할당하는 리소스 수 줄이기(Livy 세션당 리소스 수 줄이기)

접근법은 사용 사례에 따라 다릅니다. 개발 엔드포인트 크기가 크고 데이터 양이 많지 않은 경우, Spark가 동적 할당 전략을 기반으로 리소스를 할당할 수 있기 때문에 리소스 충돌이 발생할 가능성이 대폭 감소합니다.

위에서 설명한 것과 같이, Spark 실행기 수는 DPU(또는 NumberOfWorkers)와 작업자 유형을 조합하여 자동으로 계산됩니다. 각각의 Spark 애플리케이션이 드라이버 하나와 실행기 여러 개를 실행합니다. 계산하려면 NumberOfWorkers = NumberOfExecutors + 1이어야 합니다. 아래 매트릭스에서는 동시 사용자 수를 바탕으로 개발 엔드포인트에 필요한 용량을 설명합니다.

동시 노트북 사용자 수	사용자당 할당하고자 하는 Spark 실행기 수	개발 엔드포인트의 NumberOfWorkers 총합
3	5	18
10	5	60
50	5	300

사용자당 적은 리소스를 할당하고자 하는 경우, Livy 세션 파라미터로 구성할 가장 간편한 파라미터는 spark.dynamicAllocation.maxExecutors(또는 numExecutors)일 것입니다. /home/ec2-user/.sparkmagic/config.json에서 아래 구성을 설정하는 경우, SparkMagic이 Livy 세션당 최대 5개의 실행기를 할당하게 됩니다. 이렇게 하면 Livy 세션당 리소스 분리에 도움이 됩니다.

```
"session_configs": {
```

```
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

작업자가 18명인 개발 엔드포인트가 하나 있고(G.1X) 동시에 3명의 동시 노트북 사용자가 있다고 가정합시다. 세션 구성에 `spark.dynamicAllocation.maxExecutors=5`가 포함된 경우, 각 사용자가 드라이버 1개, 실행기 5개를 사용할 수 있습니다. 동시에 여러 노트북 단락을 실행한다 해도 리소스 충돌이 발생하지 않습니다.

절충

이 세션 구성 `"spark.dynamicAllocation.maxExecutors":"5"`를 사용하면 리소스 충돌 오류를 피할 수 있으며 동시 사용자 액세스가 있는 경우 리소스 할당을 기다리지 않아도 됩니다. 다만 여유 리소스가 많이 있다 하더라도(예를 들어 다른 동시 사용자가 없는 경우) Spark는 Livy 세션에 실행기를 5개 이상 할당할 수 없습니다.

기타 참고 사항

노트북 사용을 중지할 때 Jupyter 커널을 중지하는 것이 좋습니다. 이렇게 하면 여유 리소스가 확보되어 다른 노트북 사용자가 커널 만료(자동 종료)를 기다릴 필요 없이 즉시 해당 리소스를 사용할 수 있습니다.

일반적인 문제

지침을 따라도 몇 가지 문제가 발생할 수 있습니다.

세션을 찾을 수 없음

Livy 세션이 이미 종료되었는데도 노트북 단락을 실행하려 시도하면 아래와 같은 메시지가 표시됩니다. Livy 세션을 활성화하려면 Jupyter 메뉴에서 [커널(Kernel)] > [다시 시작(Restart)]을 선택하여 Jupyter 커널을 다시 시작해야 합니다.

```
An error was encountered:
Invalid status code '404' from http://localhost:8998/sessions/13 with error payload:
 "Session '13' not found."
```

YARN 리소스 부족

Spark 클러스터에 새 Livy 세션을 시작하기에 충분한 리소스가 없는데도 노트북 단락을 실행하려 시도하는 경우, 아래와 같은 메시지가 표시됩니다. 이 문제는 지침을 따르면 피할 수 있을 때가 많지만, 그래

도 이 문제에 직면할 가능성이 있습니다. 이 문제를 피하려면 불필요한 활성 Livy 세션이 있는지 확인합니다. 불필요한 Livy 세션이 있는 경우, 이를 종료하면 여유 클러스터 리소스를 확보할 수 있습니다. 자세한 내용은 다음 단원을 참조하십시오.

```
Warning: The Spark session does not have enough YARN resources to start.
The code failed because of a fatal error:
    Session 16 did not start up in 60 seconds..

Some things to try:
a) Make sure Spark has enough available resources for Jupyter to create a Spark
 context.
b) Contact your Jupyter administrator to make sure the Spark magics library is
 configured correctly.
c) Restart the kernel.
```

모니터링 및 디버깅

이 섹션에서는 리소스와 세션을 모니터링하는 기법을 설명합니다.

클러스터 리소스 할당 모니터링 및 디버깅

Spark UI를 보면 Livy 세션당 할당된 리소스 양을 모니터링하여 해당 작업에서 효과적인 Spark 구성이 무엇인지 알아볼 수 있습니다. Spark UI를 활성화하려면 개발 엔드포인트에 Apache Spark 웹 UI 사용을 참조하세요.

(선택 사항) Spark UI를 실시간으로 보려면 Spark 클러스터에서 실행 중인 Spark 기록 서버에 대하여 SSH 터널을 구성하면 됩니다.

```
ssh -i <private-key.pem> -N -L 8157:<development endpoint public address>:18080
 glue@<development endpoint public address>
```

그러면 브라우저에서 http://localhost:8157을 열어 Spark UI를 확인할 수 있습니다.

사용 가능한 불필요한 Livy 세션

노트북이나 Spark 클러스터에서 불필요한 Livy 세션을 모두 종료하는 다음과 같은 절차를 검토하세요.

(a). 노트북에서 Livy 세션 종료

Jupyter Notebook에서 커널을 종료하여 불필요한 Livy 세션을 종료할 수 있습니다.

(b). Spark 클러스터에서 Livy 세션 종료

아직 실행 중인 불필요한 Livy 세션이 있는 경우, Spark 클러스터에서 해당 Livy 세션을 종료하면 됩니다.

이 절차를 수행하기 위한 전제 조건으로, 개발 엔드포인트에 대하여 SSH 퍼블릭 키를 구성해야 합니다.

Spark 클러스터에 로그인하려면 다음과 같은 명령을 실행하면 됩니다.

```
$ ssh -i <private-key.pem> glue@<development endpoint public address>
```

다음 명령을 실행하여 활성 Livy 세션을 확인합니다.

```
$ yarn application -list
20/09/25 06:22:21 INFO client.RMProxy: Connecting to ResourceManager at
 ip-255-1-106-206.ec2.internal/172.38.106.206:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED,
 RUNNING]):2
Application-Id Application-Name Application-Type User Queue State Final-State Progress
 Tracking-URL
application_1601003432160_0005 livy-session-4 SPARK livy default RUNNING UNDEFINED 10%
 http://ip-255-1-4-130.ec2.internal:41867
application_1601003432160_0004 livy-session-3 SPARK livy default RUNNING UNDEFINED 10%
 http://ip-255-1-179-185.ec2.internal:33727
```

그리고 다음 명령으로 Livy 세션을 종료하면 됩니다.

```
$ yarn application -kill application_1601003432160_0005
20/09/25 06:23:38 INFO client.RMProxy: Connecting to ResourceManager at
 ip-255-1-106-206.ec2.internal/255.1.106.206:8032
Killing application application_1601003432160_0005
20/09/25 06:23:39 INFO impl.YarnClientImpl: Killed application
 application_1601003432160_0005
```

노트북 관리

> **Note**
>
> 개발 엔드포인트는 AWS Glue의 2.0 이전 버전에서만 지원됩니다. ETL 스크립트를 작성하고 테스트할 수 있는 대화형 환경에서는 AWS Glue Studio에서 노트북을 사용합니다.

노트북을 사용하면 개발 엔드포인트에서 추출, 전환 및 적재(ETL) 스크립트를 대화식으로 개발 및 테스트할 수 있습니다. AWS Glue는 SageMaker AI Jupyter Notebook에 대한 인터페이스를 제공합니다. AWS Glue를 사용하여 SageMaker AI 노트북을 생성하고 관리합니다. AWS Glue 콘솔에서도 SageMaker AI 노트북을 열 수 있습니다.

또한 Apache Spark를 SageMaker AI를 지원하는(AWS Glue ETL 작업은 아니고) AWS Glue 개발 엔드포인트에서 SageMaker AI와 함께 사용해도 됩니다. SageMaker Spark는 SageMaker AI용 오픈 소스 Apache Spark 라이브러리입니다. 자세한 내용을 알아보려면 Using Apache Spark with Amazon SageMaker(Amazon SageMaker에서 Apache Spark 사용)를 참조하세요.

Important

AWS Glue 개발 엔드포인트로 SageMaker AI 노트북 관리는 다음 AWS 리전에서 사용할 수 있습니다.

리전	코드
미국 동부(오하이오)	`us-east-2`
미국 동부(버지니아 북부)	`us-east-1`
미국 서부(캘리포니아 북부)	`us-west-1`
미국 서부(오레곤)	`us-west-2`
아시아 태평양(도쿄)	`ap-northeast-1`
아시아 태평양(서울)	`ap-northeast-2`
아시아 태평양(뭄바이)	`ap-south-1`
아시아 태평양(싱가포르)	`ap-southeast-1`
아시아 태평양(시드니)	`ap-southeast-2`
캐나다(중부)	`ca-central-1`
유럽(프랑크푸르트)	`eu-central-1`
유럽(아일랜드)	`eu-west-1`

리전	코드
유럽(런던)	eu-west-2

AWS Glue Studio를 사용하여 시각적 ETL 작업 구축

AWS Glue 작업은 소스 데이터에 연결하여 처리한 다음 데이터 대상에 작성하는 스크립트를 캡슐화합니다. 일반적으로 작업은 추출, 변환 및 로드(ETL) 스크립트를 실행합니다. 작업은 Apache Spark 및 Ray 런타임 환경용으로 설계된 스크립트를 실행할 수 있습니다. 작업은 범용 Python 스크립트(Python 셸 작업)를 실행할 수도 있습니다. AWS Glue 트리거는 일정 또는 이벤트에 따라 또는 필요에 따라 작업을 시작할 수 있습니다. 작업 실행을 모니터링하여 완료 상태, 지속 시간, 시작 시간 같은 실행 시간 지표를 이해할 수 있습니다.

AWS Glue에서 생성하는 스크립트를 사용하거나 직접 제공할 수 있습니다. 소스 스키마와 대상 위치 또는 스키마가 있을 경우 AWS Glue Studio 코드 생성기는 Apache Spark API(PySpark) 스크립트를 자동적으로 생성할 수 있습니다. 이 스크립트를 시작 포인트로 사용할 수 있고 목적에 부합하기 위해 편집할 수도 있습니다.

AWS Glue에서는 여러 데이터 형식으로 출력 파일을 작성할 수 있습니다. 작업 유형마다 지원하는 출력 형식이 다를 수 있습니다. 몇 가지 데이터 포맷의 경우, 일반 압축 포맷이 작성될 수 있습니다.

AWS Glue 콘솔로 로그인합니다

AWS Glue 작업은 추출, 변환 및 로드(ETL) 작업을 수행하는 데 필요한 비즈니스 로직으로 구성됩니다. AWS Glue 콘솔의 [ETL] 섹션에서 작업을 생성할 수 있습니다.

기존 작업을 보려면 AWS Management Console에 로그인하고 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다. 그런 다음 AWS Glue에서 [작업(Jobs)] 탭을 선택합니다. 작업이 마지막으로 수정되고 현재 작업이 옵션을 표시할 경우, [Jobs(작업)] 목록은 각 작업과 관련된 스크립트 위치를 보여줍니다.

새 작업을 생성하는 동안 또는 작업을 저장한 후 AWS Glue Studio를 사용하여 ETL 작업을 수정할 수 있습니다. 시각적 편집기에서 노드를 편집하거나 개발자 모드에서 작업 스크립트를 편집하여 이를 수행할 수 있습니다. 시각적 편집기에서 노드를 추가하고 제거하여 더 복잡한 ETL 작업을 생성할 수도 있습니다.

AWS Glue Studio에서 작업을 생성하기 위한 다음 단계

시각적 작업 편집기를 사용하여 작업에 대한 노드를 구성합니다. 각 노드는 소스 위치에서 데이터 읽기 또는 데이터에 변환 적용과 같은 작업을 나타냅니다. 작업에 추가하는 각 노드에는 데이터 위치 또는 변환에 대한 정보를 제공하는 속성이 있습니다.

다음은 작업을 생성하고 관리하는 단계입니다.

- AWS Glue Studio에서 시각적 ETL 작업 시작
- 작업 스크립트 보기
- 작업 속성 수정
- 작업 저장
- 작업 실행 시작
- 최근 작업 실행에 대한 정보 보기
- 작업 모니터링 대시보드에 액세스

AWS Glue Studio에서 시각적 ETL 작업 시작

AWS Glue Studio의 간단한 시각적 인터페이스를 사용하여 ETL 작업을 생성할 수 있습니다. [작업(Jobs)] 페이지를 사용하여 새 작업을 생성합니다. 스크립트 편집기 또는 노트북을 사용하여 AWS Glue Studio ETL 작업 스크립트의 코드로 직접 작업할 수도 있습니다.

작업(Jobs) 페이지에서 AWS Glue Studio 또는 AWS Glue로 생성한 모든 작업을 볼 수 있습니다. 이 페이지에서 작업을 보고, 관리하고, 실행할 수 있습니다.

AWS Glue Studio에서 ETL 작업을 생성하는 방법에 대한 또 다른 예제는 블로그 자습서도 참조하세요.

AWS Glue Studio에서 작업 시작

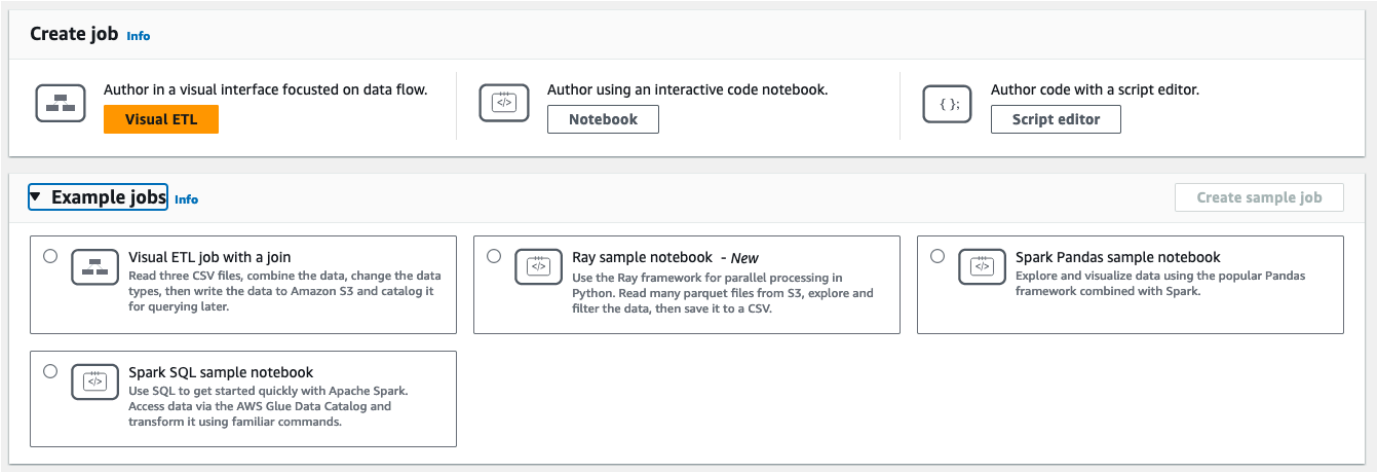
AWS Glue에서는 시각적 인터페이스, 대화형 코드 노트북 또는 스크립트 편집기를 통해 작업을 생성할 수 있습니다. 원하는 옵션을 클릭하여 작업을 시작하거나 샘플 작업을 기반으로 새 작업을 생성할 수 있습니다.

선택한 도구를 사용하여 샘플 작업에서 작업을 생성합니다. 예를 들어 샘플 작업을 사용하면 CSV 파일을 카탈로그 테이블에 결합하는 시각적 ETL 작업을 생성하거나 pandas에 대한 작업을 수행할 때 AWS Glue for Ray 또는 AWS Glue for Spark에서 대화형 코드 노트북에서 작업을 생성하거나 SparkSQL을 사용하여 대화형 코드 노트북에서 작업을 생성할 수 있습니다.

처음부터 AWS Glue Studio에서 작업 생성

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.
2. 탐색 창에서 ETL 작업을 선택합니다.

3. 작업 생성 섹션에서 작업에 대한 구성 옵션을 선택합니다.



처음부터 새로 작업을 생성하는 옵션:

- 시각적 ETL - 데이터 흐름에 초점을 맞춘 시각적 인터페이스에서 작성
- 대화형 코드 노트북을 사용하여 작성 - Jupyter Notebook 기반의 노트북 인터페이스에서 대화형으로 작업 작성

  이 옵션을 선택하는 경우 노트북 작성 세션을 생성하기 전에 추가 정보를 제공해야 합니다. 이 정보를 지정하는 방법에 대한 자세한 내용은 AWS Glue Studio에서 노트북 시작하기 섹션을 참조하세요.
- 스크립트 편집기에서 코드 작성 - ETL 스크립트 작성 및 프로그래밍에 익숙한 사용자의 경우 이 옵션을 선택하여 새 Spark ETL 작업을 생성합니다. 엔진(Python 쉘, Ray, Spark(Python) 또는 Spark(Scala))를 선택합니다. 그런 다음 새로 시작 또는 스크립트 업로드를 선택합니다. 그러면 로컬 파일에서 기존 스크립트를 업로드합니다. 스크립트 편집기를 사용하는 옵션을 선택한 경우 시각적 작업 편집기를 사용하여 작업을 설계하거나 편집할 수 없습니다.

  Spark 작업은 AWS Glue에서 관리하는 Apache Spark 환경에서 실행됩니다. 기본적으로 새 스크립트는 Python으로 코딩됩니다. 새 Scala 스크립트를 작성하려면 AWS Glue Studio에서 Scala 스크립트 생성 및 편집 섹션을 참조하세요.

예제 작업으로부터 AWS Glue Studio에서 작업 생성

예제 작업에서 작업을 생성하도록 선택할 수 있습니다. 예제 작업 섹션에서 샘플 작업을 선택한 다음 샘플 작업 생성을 선택합니다. 옵션 중 하나에서 샘플 작업을 생성하면 작업할 수 있는 빠른 템플릿이 제공됩니다.

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.
2. 탐색 창에서 ETL 작업을 선택합니다.
3. 샘플 작업에서 작업을 생성하는 옵션을 선택합니다.
   - 여러 소스를 결합하는 시각적 ETL 작업 - 세 개의 CSV 파일을 읽고, 데이터를 결합한 후 데이터 형식을 변경하고 Amazon S3에 데이터를 쓴 후에 나중에 쿼리할 수 있도록 카탈로그화합니다.
   - Pandas를 사용하는 Spark 노트북 - Spark와 결합된 인기 있는 Pandas 프레임워크를 사용하여 데이터를 탐색하고 시각화합니다.
   - SQL을 사용하는 Spark 노트북 - SQL을 사용하여 Apache Spark를 빠르게 시작할 수 있습니다. AWS Glue 데이터 카탈로그를 통해 데이터에 액세스하고 친숙한 명령을 사용하여 데이터를 변환합니다.
4. 샘플 작업 생성을 선택합니다.

작업 편집기 기능

작업 편집기는 작업 생성 및 편집을 위해 다음과 같은 기능을 제공합니다.

- 각 작업 태스크에 대한 노드가 있는 작업의 시각적 다이어그램: 데이터를 읽기 위한 데이터 원본 노드. 데이터를 수정하기 위한 변환 노드, 데이터를 쓰기 위한 데이터 대상 노드.

  작업 다이어그램에서 각 노드의 속성을 보고 구성할 수 있습니다. 작업 다이어그램에서 각 노드에 대한 스키마 및 샘플 데이터를 볼 수도 있습니다. 이러한 기능을 사용하면 작업을 실행할 필요 없이 작업이 올바른 방식으로 데이터를 수정하고 변환하는지 확인할 수 있습니다.
- 작업에 대해 생성된 코드를 수정할 수 있는 스크립트 보기 및 편집 탭.
- AWS Glue ETL 작업이 실행되는 환경을 사용자 정의하기 위해 다양한 설정을 구성할 수 있는 작업 세부 정보 탭.
- 작업의 현재 및 이전 실행을 보고, 작업 실행 상태를 보고, 작업 실행에 대한 로그에 액세스할 수 있는 실행 탭.
- 작업에 데이터 품질 규칙을 적용할 수 있는 데이터 품질 탭.
- 작업 시작 시간을 구성하거나 반복 작업 실행을 설정할 수 있는 일정 탭.
- 작업에 사용할 Git 서비스를 구성할 수 있는 버전 제어 탭.

시각적 작업 편집기에서 스키마 미리 보기 사용

작업을 생성하거나 편집하는 동안 [출력 스키마(Output schema)] 탭을 사용하여 데이터에 대한 스키마를 볼 수 있습니다.

스키마를 보려면 먼저 작업 편집기에 데이터 원본에 액세스할 수 있는 권한이 필요합니다. 편집기의 작업 세부 정보 탭이나 노드의 [출력 스키마(Output schema)] 탭에서 IAM 역할을 지정할 수 있습니다. IAM 역할에 데이터 원본에 액세스하는 데 필요한 모든 권한이 있는 경우 노드의 [출력 스키마(Output schema)] 탭에서 스키마를 볼 수 있습니다.

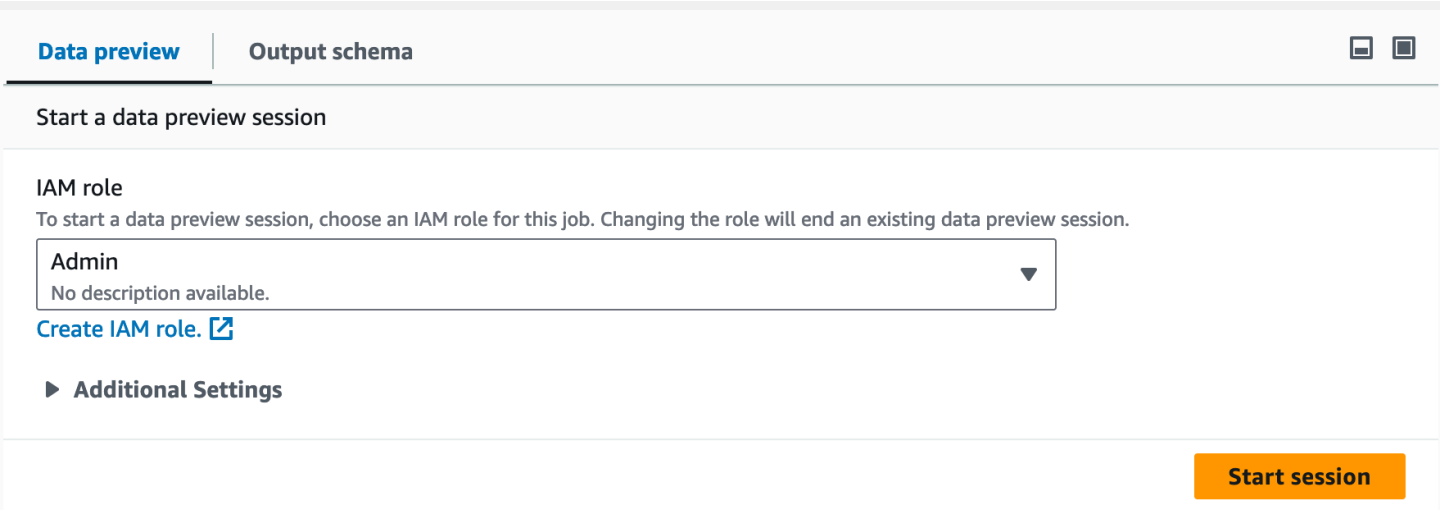
시각적 작업 편집기에서 데이터 미리 보기 사용

데이터 미리 보기를 사용하면 작업을 반복적으로 실행할 필요 없이 데이터 샘플을 이용하여 작업을 생성하고 테스트할 수 있습니다. 데이터 미리 보기를 사용하여 다음을 수행할 수 있습니다.

- IAM 역할을 테스트하여 데이터 소스 또는 데이터 대상에 대한 액세스 권한이 있는지 확인할 수 있습니다.
- 변환이 의도한 방식으로 데이터를 수정하고 있는지 확인할 수 있습니다. 예를 들어 필터 변환을 사용하는 경우 필터가 올바른 데이터 하위 집합을 선택하는지 확인할 수 있습니다.
- 데이터를 확인하십시오. 데이터 집합에 여러 유형의 값이 있는 열이 포함된 경우 데이터 미리 보기에 이러한 열에 대한 튜플 목록이 표시됩니다. 각 튜플에는 데이터 유형과 해당 값이 포함됩니다.

> **Note**
>
> 데이터 미리 보기 세션과 사용자 지정 SQL 또는 사용자 지정 코드 노드를 사용하는 경우 데이터 미리 보기 세션은 전체 데이터세트에 대해 SQL 또는 코드 블록을 있는 그대로 실행합니다.

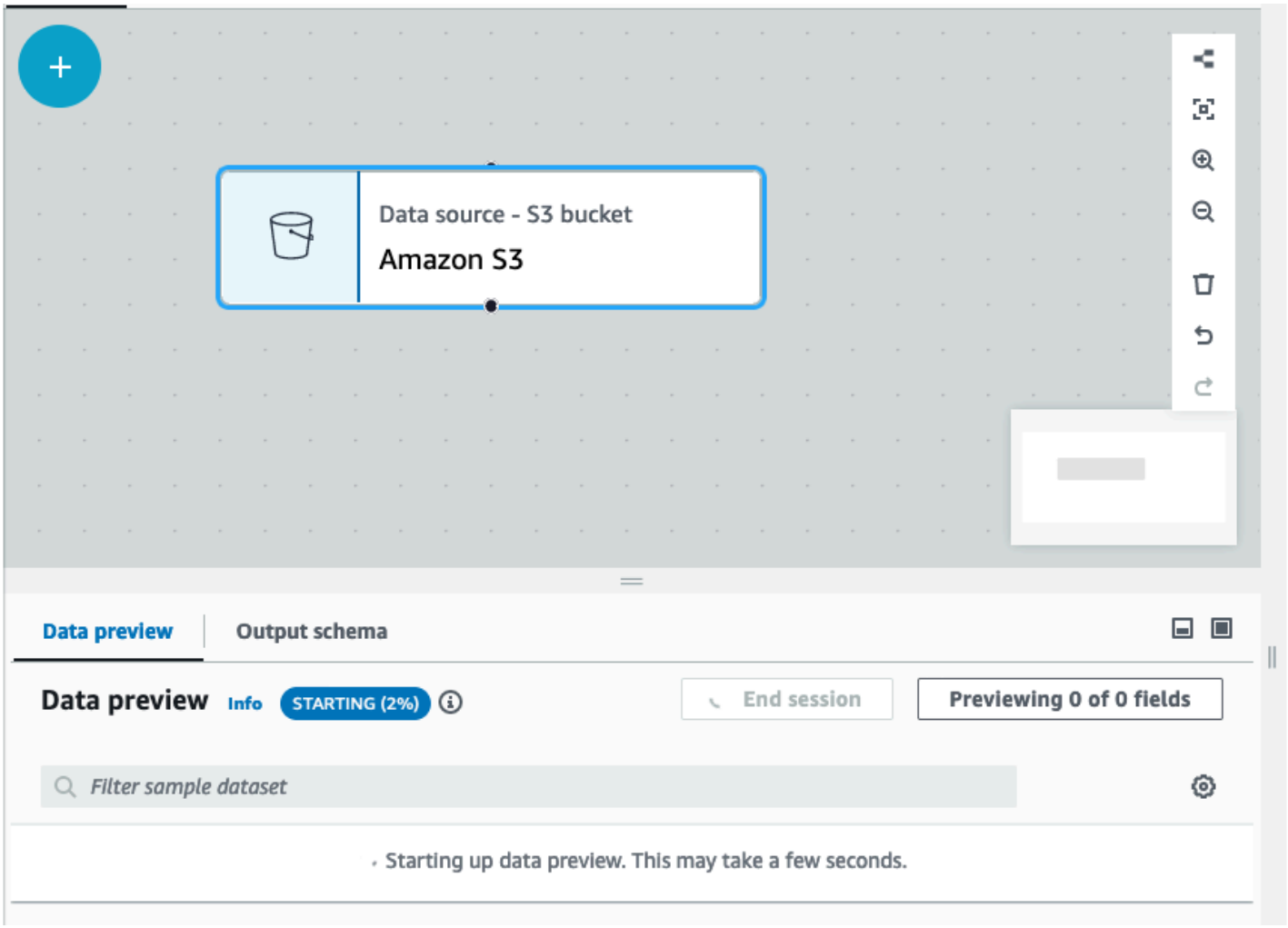
작업을 생성하거나 편집하는 동안 작업 캔버스 아래에 있는 데이터 미리 보기 탭을 사용하여 데이터 샘플을 볼 수 있습니다. 작업에 역할이 이미 구성되어 있거나 계정에 기본 IAM 역할이 설정된 경우 새 데이터 미리 보기 세션이 자동으로 시작됩니다. 역할이 이전에 구성되지 않은 경우 역할을 선택하여 세션을 시작할 수 있습니다.



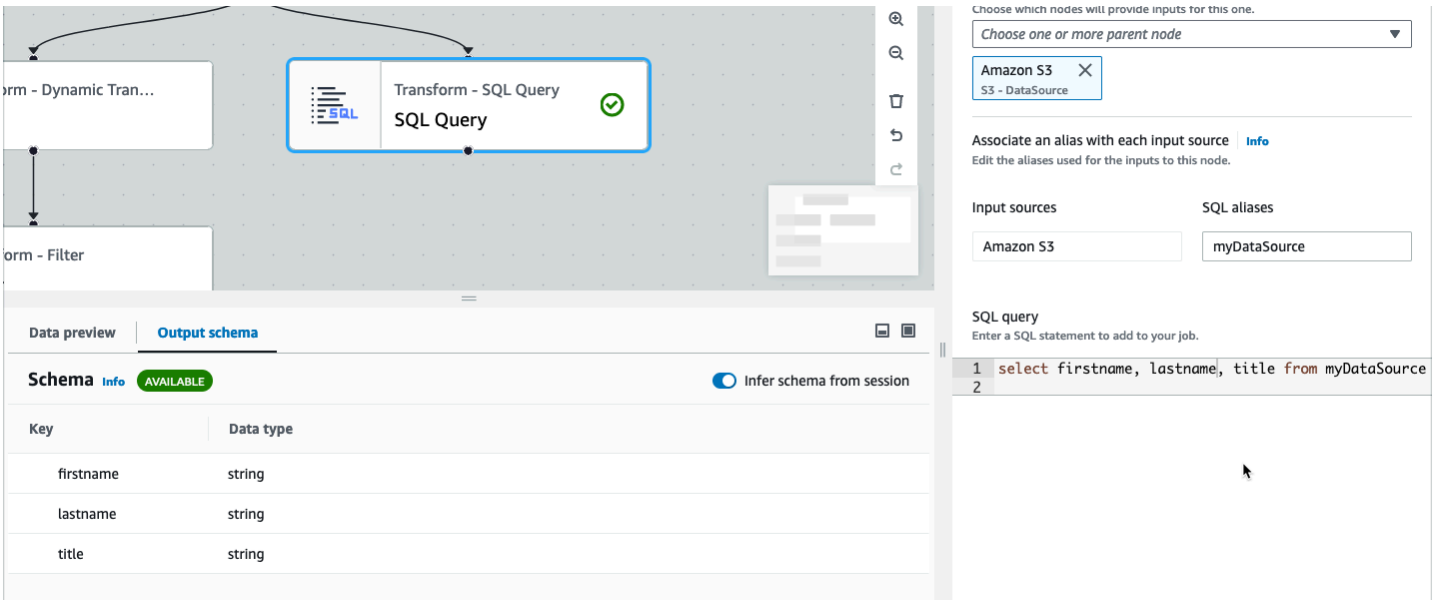
> **Note**
>
> 데이터 미리 보기 세션에서 선택한 역할이 작업에도 사용됩니다.

정보 아이콘을 클릭하면 세션 상태와 진행 상황, 세션 세부 정보를 볼 수 있습니다.

세션이 준비되면 AWS Glue Studio가 선택한 노드에 대한 데이터를 로드합니다. 진행 상황에 따라 완료율 %을 확인할 수 있습니다.



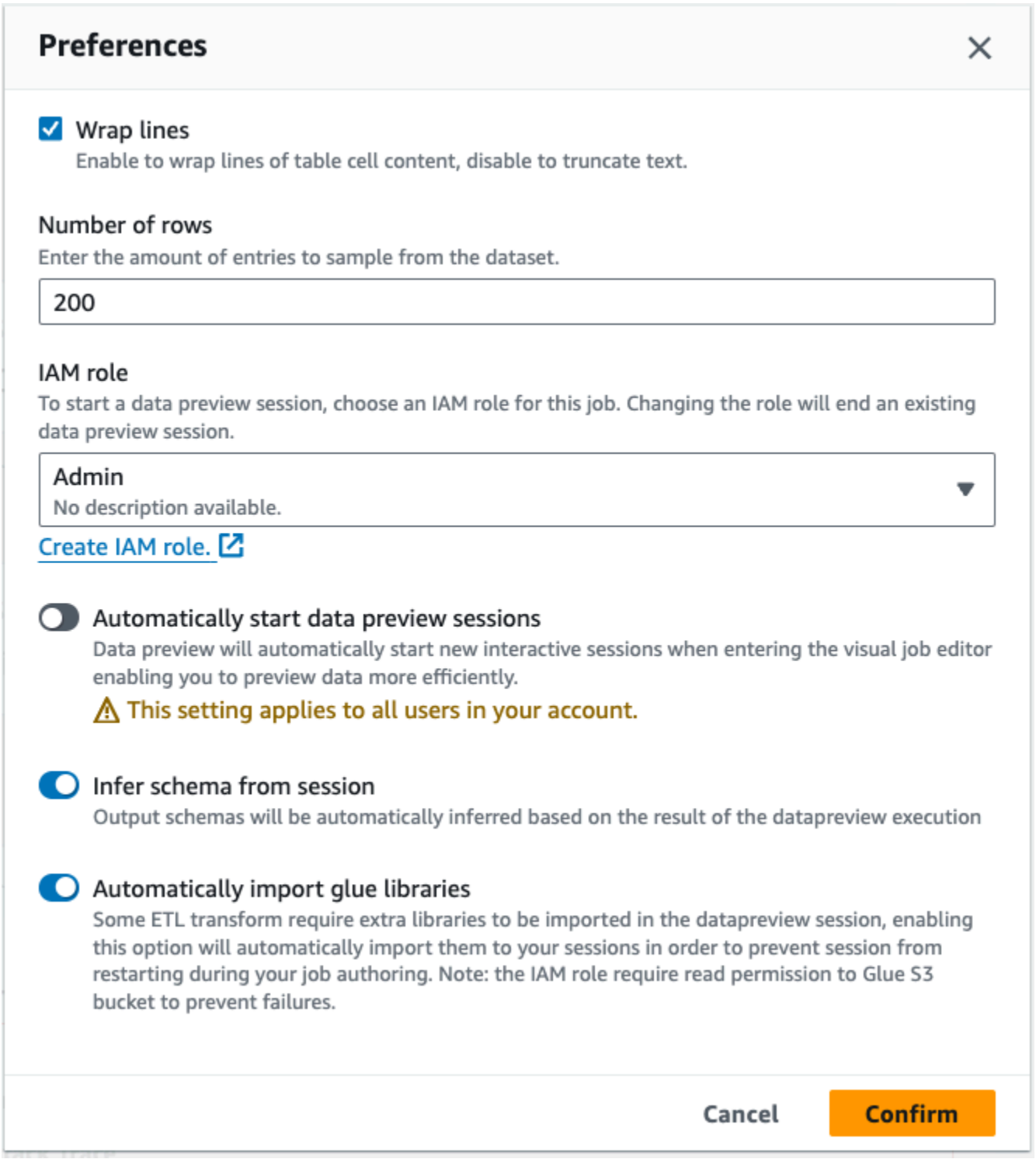
시각적 작업을 작성할 때 출력 스키마 탭에서 세션에서 스키마 추론을 전환하면 선택한 노드의 스키마를 AWS Glue Studio가 자동으로 업데이트합니다.



데이터 미리 보기 기본 설정을 구성하는 방법:

설정 아이콘(기어 기호)을 선택하여 데이터 미리 보기에 대한 기본 설정을 구성합니다. 이러한 설정은 작업 다이어그램의 모든 노드에 적용됩니다. 다음을 할 수 있습니다.

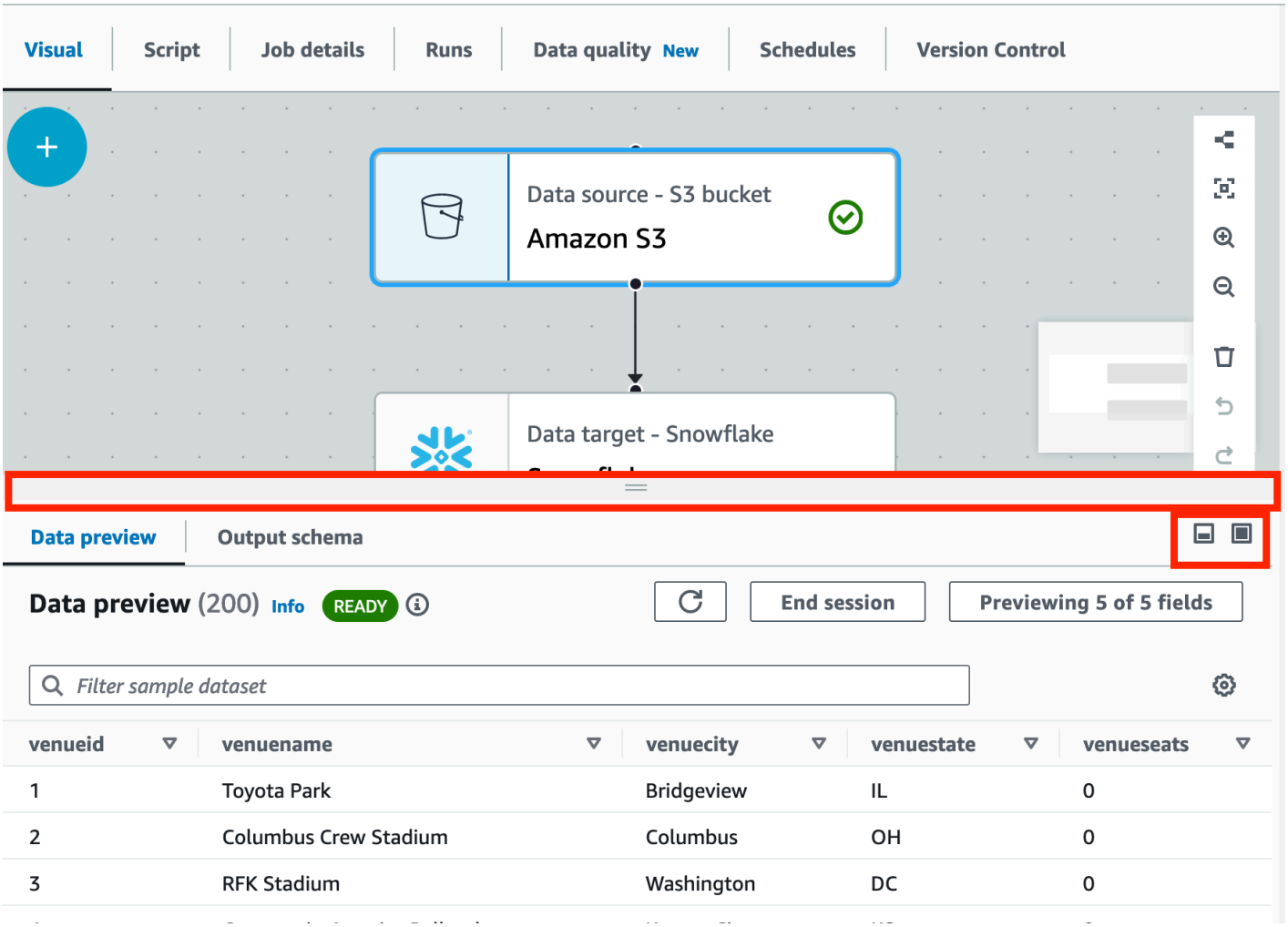
- 한 줄에서 다음 줄로 텍스트를 줄 바꿈하도록 선택합니다. 이 옵션은 기본적으로 활성화되어 있습니다
- 행 수 변경(기본값 200개)
- 필요한 경우 IAM 역할을 선택하거나 IAM 역할을 생성합니다
- 작업을 작성할 때 새 세션을 자동으로 시작하도록 선택합니다. 이렇게 하면 작업을 작성할 때 새 대화형 세션이 프로비저닝됩니다. 이 설정은 계정 수준에서 적용됩니다. 일단 설정하면 작업을 편집할 때 계정의 모든 사용자에게 적용됩니다.
- 스키마를 자동으로 유추하도록 선택합니다. 선택한 노드에 대해 출력 스키마가 자동으로 추론됩니다
- AWS Glue 라이브러리를 자동으로 가져오도록 선택합니다. 이는 세션을 다시 시작해야 하는 새 변환을 추가할 때 데이터 미리 보기에서 새 세션이 다시 시작되지 않도록 하므로 유용합니다.



추가 기능에는 다음과 같은 기능이 포함됩니다.

- [필드 y개 중 x개 미리 보기(Previewing x of y fields)] 버튼을 선택하여 미리 보려는 열(필드)을 선택합니다. 기본 설정을 사용하여 데이터를 미리 보면 작업 편집기에 데이터 집합의 처음 5개 열이 표시됩니다. 모두 표시하거나 표시하지 않도록 변경할 수 있습니다(권장하지 않음).

- 데이터 미리 보기 창을 가로 및 세로로 스크롤할 수 있습니다.
- 최대화 버튼을 사용하여 데이터 미리보기 탭을 오버레이 작업 그래프로 확장하면 데이터 및 데이터 구조를 더 잘 볼 수 있습니다. 이와 유사하게 최소화 버튼을 사용하여 데이터 미리 보기 탭을 최소화합니다. 핸들 창을 잡고 위로 드래그하여 데이터 미리 보기 탭을 확장할 수도 있습니다.



- 세션 종료를 사용하여 데이터 미리 보기를 중지합니다. 세션을 중지할 때 새 IAM 역할을 선택하고, 새 세션을 자동으로 시작하거나, 스키마를 유추하거나, AWS Glue 라이브러리를 가져와서 세션을 다시 시작하도록 추가 설정(예: 설정 켜기 또는 끄기)을 설정할 수 있습니다.

데이터 미리 보기 사용 시 제한 사항

데이터 미리 보기를 사용할 때 다음과 같은 제한 사항이 있을 수 있습니다.

- [데이터 미리 보기(Data preview)] 탭을 처음 선택할 때 IAM 역할을 선택해야 합니다. 이 역할에는 데이터 미리 보기를 만드는 데 필요한 데이터 및 기타 리소스에 액세스하는 데 필요한 권한이 있어야 합니다.

- IAM 역할을 제공한 후 데이터를 볼 수 있을 때까지 시간이 걸립니다. 데이터가 1GB 미만인 데이터 집합의 경우 최대 1분이 소요될 수 있습니다. 큰 데이터 집합이 있는 경우 파티션을 사용하여 로드 시간을 개선해야 합니다. Amazon S3에서 직접 데이터를 로드하는 것이 성능이 가장 좋습니다.
- 매우 큰 데이터 집합이 있고 데이터 미리 보기를 위해 데이터를 쿼리하는 데 15분 이상 걸리는 경우 요청 시간이 초과됩니다. 데이터 미리 보기의 유휴 제한 시간은 30분입니다. 이를 완화하려면 데이터 미리 보기 사용에 데이터 세트 크기를 줄이세요.
- 기본으로 처음 50 열이 데이터 미리 보기 탭에 표시됩니다. 열에 데이터 값이 없는 경우 표시 할 데이터가 없다는 메시지가 나타납니다. 샘플링된 행 수를 늘리거나 다른 열을 선택하여 데이터 값을 볼 수 있습니다.
- 데이터 미리 보기는 현재 스트리밍 데이터 원본 또는 사용자 정의 커넥터를 사용하는 데이터 원본에 대해 지원되지 않습니다.
- 한 노드의 오류는 전체 작업에 영향을 줍니다. 데이터 미리 보기에서 한 노드에 오류가 있는 경우 이를 수정할 때까지 모든 노드에 오류가 표시됩니다.
- 작업의 데이터 원본을 변경하는 경우 해당 데이터 원본의 하위 노드를 새 스키마와 일치하도록 업데이트해야 할 수 있습니다. 예를 들어 열을 수정하는 ApplyMapping 노드가 있고 해당 열이 대체 데이터 원본에 없는 경우 ApplyMapping 변환 노드를 업데이트해야 합니다.
- SQL 쿼리 변환 노드에 대한 데이터 미리 보기 탭을 볼 때 SQL 쿼리에서 잘못된 필드 이름을 사용하면 데이터 미리 보기 탭에 오류가 표시됩니다.

스크립트 코드 생성

시각적 편집기를 사용하여 작업을 생성하는 경우 자동으로 ETL 코드가 생성됩니다. AWS Glue Studio는 기능적이고 완전한 작업 스크립트를 생성하여 Amazon S3 위치에 저장합니다.

AWS Glue Studio에서는 원본 또는 클래식 버전과 간소화된 최신 버전인 두 가지 형식의 코드를 생성합니다. 기본값으로 새 코드 생성기가 작업 스크립트를 생성하는 데 사용됩니다. 스크립트(Script) 탭에서 클래식 스크립트 생성(Generate classic script) 토글 버튼을 선택하여 클래식 코드 생성기로 작업 스크립트를 생성할 수 있습니다.

새 버전의 생성된 코드에서 유의할 몇 가지 차이점은 다음과 같습니다.

- 더 이상 스크립트에 큰 주석 블록이 추가되지 않습니다.
- 시각적 편집기에서 지정한 노드 이름이 코드의 출력 구조에 사용됩니다. 클래스 스크립트에서 출력 구조의 이름이 간단히 `DataSource0`, `DataSource1`, `Transform0`, `Transform1`, `DataSink0`, `DataSink1` 등으로 지정됩니다.
- 긴 명령이 여러 줄로 분할되므로 전체 명령을 보기 위해 페이지를 스크롤할 필요가 없습니다.

AWS Glue Studio의 새로운 기능을 사용하려면 새 버전의 코드를 생성해야 하며 클래식 코드 스크립트에서는 작동하지 않습니다. 이러한 작업을 실행하려고 하면 업데이트하라는 메시지가 표시됩니다.

AWS Glue 관리형 변환으로 데이터 변환

AWS Glue Studio에서는 두 가지 유형의 변환을 제공합니다.

- AWS Glue 네이티브 변환 - 모든 사용자가 사용할 수 있으며 AWS Glue에서 관리합니다.
- 사용자 지정 시각적 변환 - 자체 변환을 업로드하여 AWS Glue Studio에서 사용할 수 있습니다.

AWS Glue 관리형 데이터 변환 노드

AWS Glue Studio는 기본 설정 변환 세트를 제공하여 데이터를 사용할 수 있습니다. 데이터는 작업 다이어그램의 한 노드에서 Apache Spark SQL `DataFrame`의 확장인 `DynamicFrame`이라는 데이터 구조의 다른 노드로 전달됩니다.

작업에 대해 미리 채워진 다이어그램에서 데이터 소스와 데이터 대상 노드 사이에는 스키마 변경 변환 노드가 있습니다. 이 변환 노드를 구성하여 데이터를 수정하거나 추가 변환을 사용할 수 있습니다.

AWS Glue Studio에서 사용 가능한 기본 제공 변환은 다음과 같습니다.

- ChangeSchema: 데이터 소스의 데이터 속성 키를 데이터 대상의 데이터 속성 키에 매핑합니다. 키의 이름을 바꾸고 키의 데이터 유형을 수정하고 데이터 집합에서 삭제할 키를 선택할 수 있습니다.
- SelectFields: 유지할 데이터 속성 키를 선택합니다.
- DropFields: 삭제할 데이터 속성 키를 선택합니다.
- RenameField: 단일 데이터 속성 키의 이름을 바꿉니다.
- Spigot: Amazon S3 버킷에 데이터 샘플을 씁니다.
- Join: 지정된 데이터 속성 키의 비교 구문을 사용하여 두 데이터 집합을 하나의 데이터 집합으로 조인합니다. 내부, 외부, 왼쪽, 오른쪽, 왼쪽 반 및 왼쪽 안티 조인을 사용할 수 있습니다.
- Union: 스키마가 동일한 둘 이상의 데이터 소스에서 행을 결합합니다.
- SplitFields: 데이터 속성 키를 두 개의 `DynamicFrames`로 분할합니다. 출력은 `DynamicFrames`의 컬렉션입니다. 하나는 선택한 데이터 속성 키가 있고 다른 하나는 나머지 데이터 속성 키가 있습니다.
- SelectFromCollection: `DynamicFrames` 컬렉션에서 `DynamicFrame`을 하나 선택합니다. 출력은 선택된 `DynamicFrame`입니다.

- FillMissingValues: 데이터 집합에서 누락 값이 있는 레코드를 찾고 대체를 통해 결정된 제안 값으로 새 필드를 추가합니다.
- 필터(Filter): 필터 조건에 따라 하나의 데이터 집합을 두 개로 분할합니다.
- Null 필드 삭제: 열의 모든 값이 'null'인 경우 데이터 집합에서 열을 제거합니다.
- 중복 삭제: 전체 행을 일치시키거나 키를 지정하도록 선택하여 데이터 소스에서 행을 제거합니다.
- SQL: SQL 쿼리를 사용하여 데이터를 변환하려면 텍스트 입력 필드에 SparkSQL 코드를 입력합니다. 출력은 단일 `DynamicFrame`입니다.
- 집계: 선택한 필드와 행에서 계산(예: 평균, 합계, 최소, 최대)을 수행하고 새로 계산된 값으로 새 필드를 생성합니다.
- Flatten: 구조체 내부의 필드를 최상위 필드로 추출합니다.
- UUID: 각 행에 범용 고유 식별자가 있는 열을 추가합니다.
- 식별자: 각 행에 숫자 식별자가 있는 열을 추가합니다.
- 타임스탬프로 변환: 열을 타임스탬프 유형으로 변환합니다.
- 타임스탬프 형식 지정: 타임스탬프 열을 형식이 지정된 문자열로 변환합니다.
- 조건부 라우터 변환: 수신 데이터에 여러 조건을 적용합니다. 수신 데이터의 각 행은 그룹 필터 조건을 기준으로 평가되고 해당 그룹으로 처리됩니다.
- 열 연결 변환: 선택적 스페이서가 있는 다른 열의 값을 사용하여 새 문자열 열을 구축합니다.
- 문자열 분할 변환: 정규식을 사용하여 문자열을 토큰 배열로 분할해 분할 방식을 정의합니다.
- 배열을 열로 변환: 배열 유형의 열에 있는 일부 또는 모든 요소를 새 열로 추출합니다.
- 현재 타임스탬프 추가 변환: 데이터가 처리된 시간으로 행을 표시합니다. 이는 감사 목적이나 데이터 파이프라인에서 지연 시간을 추적하는 데 유용합니다.
- 행을 열로 피벗 변환: 선택한 열에서 고유 값을 교체하여 숫자 열을 집계합니다. 이 열은 새 열이 됩니다. 열을 여러 개 선택하면 값이 연결되어 새 열의 이름이 지정됩니다.
- 열을 행으로 피벗 취소 변환: 열을 새 열의 값으로 변환하여 각 고유 값에 대한 행을 생성합니다.
- 처리 균형 자동 조절 변환: 더 나은 성능을 위해 작업자 사이에서 데이터를 재배포합니다. 이는 데이터가 불균형하거나 소스에서 가져온 데이터로 인해 충분한 병렬 처리가 불가능한 경우에 유용합니다.
- 파생 열 변환: 상수 및 리터럴뿐만 아니라 데이터의 다른 열을 사용할 수 있는 수학 공식 또는 SQL 표현식을 기반으로 새 열을 정의합니다.
- 조회 변환: 키가 데이터에 정의된 조회 열과 일치하는 경우 정의된 카탈로그 테이블의 열을 추가합니다.
- 배열 또는 맵을 행으로 분해 변환: 중첩된 구조에서 조작하기 쉬운 개별 행으로 값을 추출합니다.

- 레코드 일치 변환: 기존 레코드 일치 기계 학습 데이터 분류 변환을 간접 호출합니다.
- null 행 제거 변환 제거: 모든 열이 null이거나 비어 있는 행을 데이터세트에서 제거합니다.
- JSON 열 구문 분석 변환: JSON 데이터를 포함하는 문자열 열을 구문 분석하고 JSON이 객체인지 또는 배열인지에 따라 각각 해당 문자열 열을 구문 또는 배열 열로 변환합니다.
- JSON 경로 추출 변환: JSON 문자열 열에서 새 열을 추출합니다.
- 정규식에서 문자열 조각 추출: 정규식을 사용하여 문자열 조각을 추출하고 문자열 조각에서 새 열을 생성하거나 정규식 그룹을 사용하는 경우 여러 열을 생성합니다.
- 사용자 지정 변환(Custom transform): 사용자 지정 변환을 사용하려면 텍스트 입력 필드에 코드를 입력합니다. 출력은 `DynamicFrames`의 컬렉션입니다.

AWS Glue Studio에서 데이터 준비 레시피 사용

데이터 준비 레시피 변환을 사용하면 대화형 그리드 스타일 작성 인터페이스를 사용하여 처음부터 데이터 준비 레시피를 작성할 수 있습니다. 또한 기존 AWS Glue DataBrew 레시피를 가져온 다음 AWS Glue Studio에서 편집할 수 있습니다.

데이터 준비 레시피 노드는 리소스 패널에서 사용할 수 있습니다. 데이터 준비 레시피 노드를 시각적 워크플로의 다른 노드 (데이터 소스 노드이거나 다른 변환 노드)에 연결할 수 있습니다. AWS Glue DataBrew 레시피와 버전을 선택하면 레시피에 적용된 단계가 노드 속성 탭에 표시됩니다.

사전 조건

- AWS Glue DataBrew 레시피를 가져오는 경우 AWS Glue Studio에서 AWS Glue DataBrew 레시피 가져오기에 설명된 대로 필요한 IAM 권한이 있어야 합니다.
- 데이터 미리 보기 세션이 생성되어 있어야 합니다.

제한 사항

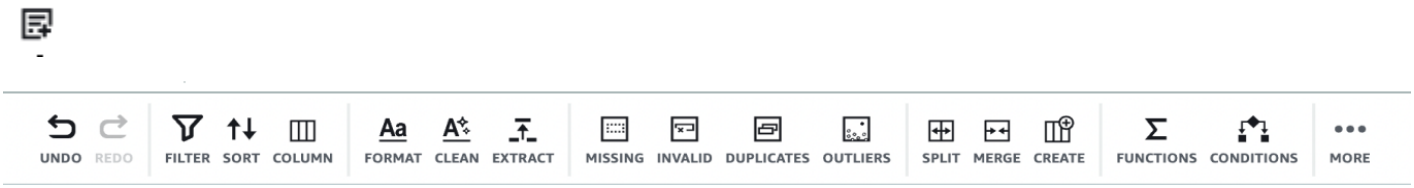
- AWS Glue DataBrew 레시피는 상용 DataBrew 리전에서만 지원됩니다.
- AWS Glue에서 모든 AWS Glue DataBrew 레시피가 지원되는 것은 아닙니다. 일부 레시피는 AWS Glue Studio에서 실행할 수 없습니다.
  - `UNION` 및 `JOIN` 변환이 포함된 레시피는 지원되지 않습니다. 대신, AWS Glue Studio에는 데이터 준비 레시피 노드 이전 또는 이후에 사용할 수 있는 '조인' 및 '집합' 변환 노드가 이미 있습니다.
- 데이터 준비 레시피 노드는 AWS Glue 버전 4.0 이상 작업에 대해 지원됩니다. 이 버전은 데이터 준비 레시피 노드가 작업에 추가된 후에 자동으로 선택됩니다.

- 데이터 준비 레시피 노드에는 Python이 필요합니다. 데이터 준비 레시피 노드가 작업에 추가될 때 자동으로 설정됩니다.
- 시각적 그래프에 새 데이터 준비 레시피 노드를 추가하면 올바른 라이브러리로 데이터 미리 보기 세션을 자동으로 재시작하여 데이터 준비 레시피 노드를 사용할 수 있습니다.
- `GROUP_BY`, `PIVOT`, `UNPIVOT` 및 `TRANSPOSE` 변환은 데이터 준비 레시피 노드에서 가져오기 또는 편집이 지원되지 않습니다.

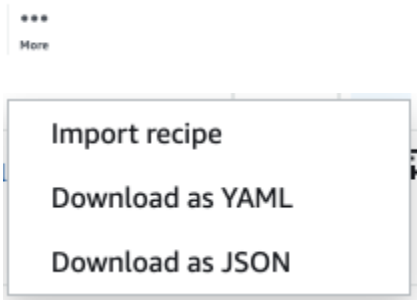
기타 기능

데이터 준비 레시피 변환을 선택하면 레시피 작성을 선택한 후 추가 작업을 수행할 수 있습니다.

- 단계 추가 - 필요에 따라 단계 추가 아이콘을 선택하여 레시피에 단계를 추가하거나, 작업을 선택하여 미리 보기 창의 도구 모음을 사용할 수 있습니다.



- 레시피 가져오기 - 더 보기를 선택한 다음 레시피 가져오기를 선택하여 AWS Glue Studio 작업에 사용합니다.

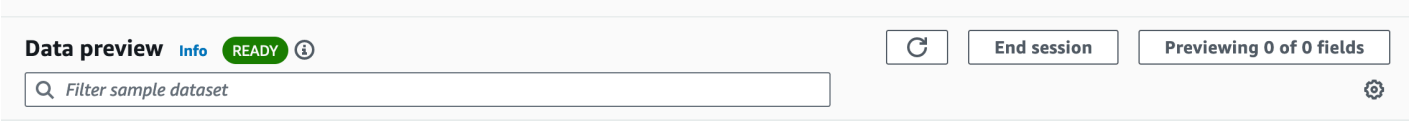


- YAML로 다운로드 - 더 보기를 선택한 다음 YAML로 다운로드를 선택하여 레시피를 다운로드하고 AWS Glue Studio 외부에 저장합니다.
- JSON으로 다운로드 - 더 보기를 선택한 다음 JSON으로 다운로드를 선택하여 레시피를 다운로드하고 AWS Glue Studio 외부에 저장합니다.
- 레시피 단계 실행 취소 및 재실행 - 그리드의 데이터를 사용할 때 미리 보기 창에서 레시피 단계를 실행 취소하고 다시 실행할 수 있습니다.

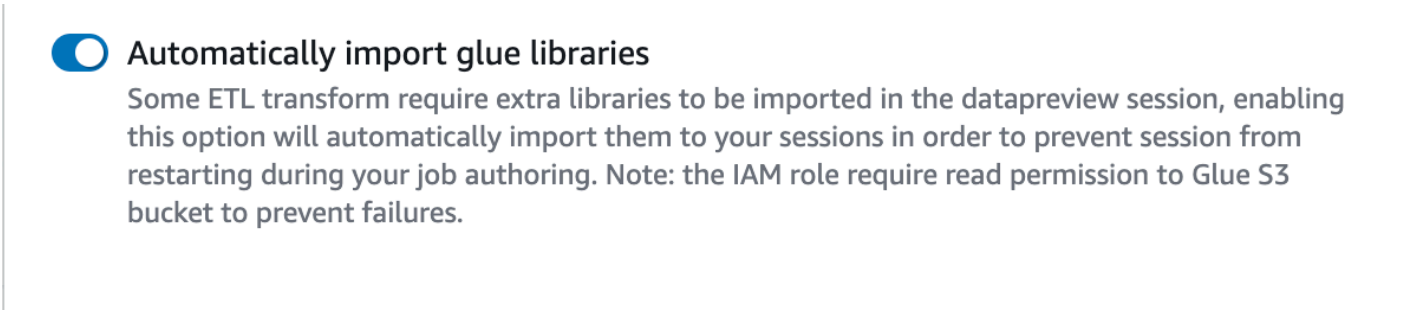
시각적 ETL AWS Glue 작업에서 데이터 준비 레시피 작성 및 실행

이 시나리오에서는 DataBrew에서 먼저 만들지 않고도 데이터 준비 레시피를 작성할 수 있습니다. 레시피 작성을 시작하려면 먼저 다음을 수행해야 합니다.

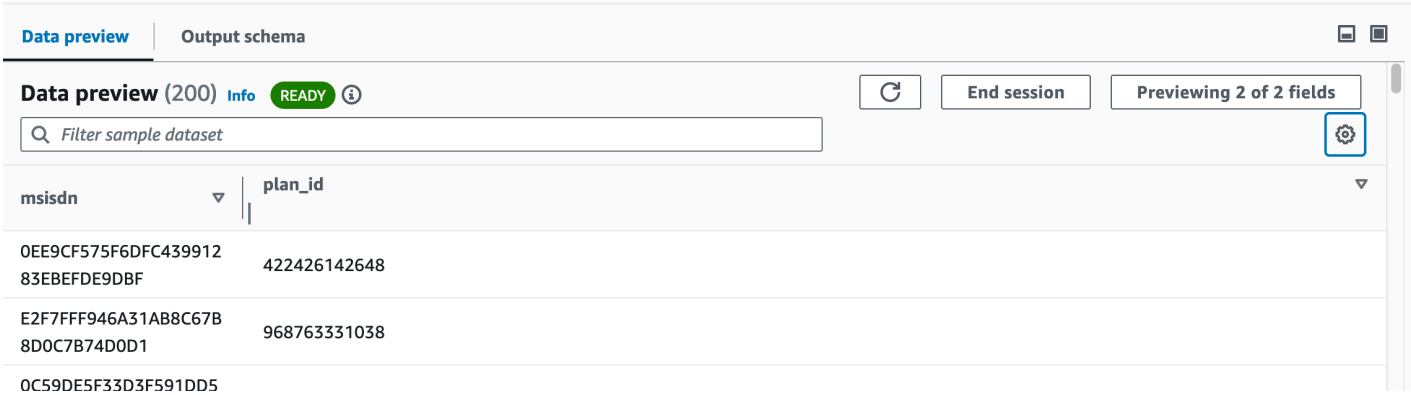
- 활성 데이터 미리 보기 세션을 실행합니다. 데이터 미리 보기 세션이 준비되면 레시피 작성이 활성화되고 레시피 작성 또는 편집을 시작할 수 있습니다.



- Glue 라이브러리 자동 가져오기 토글이 활성화되어 있는지 확인합니다.

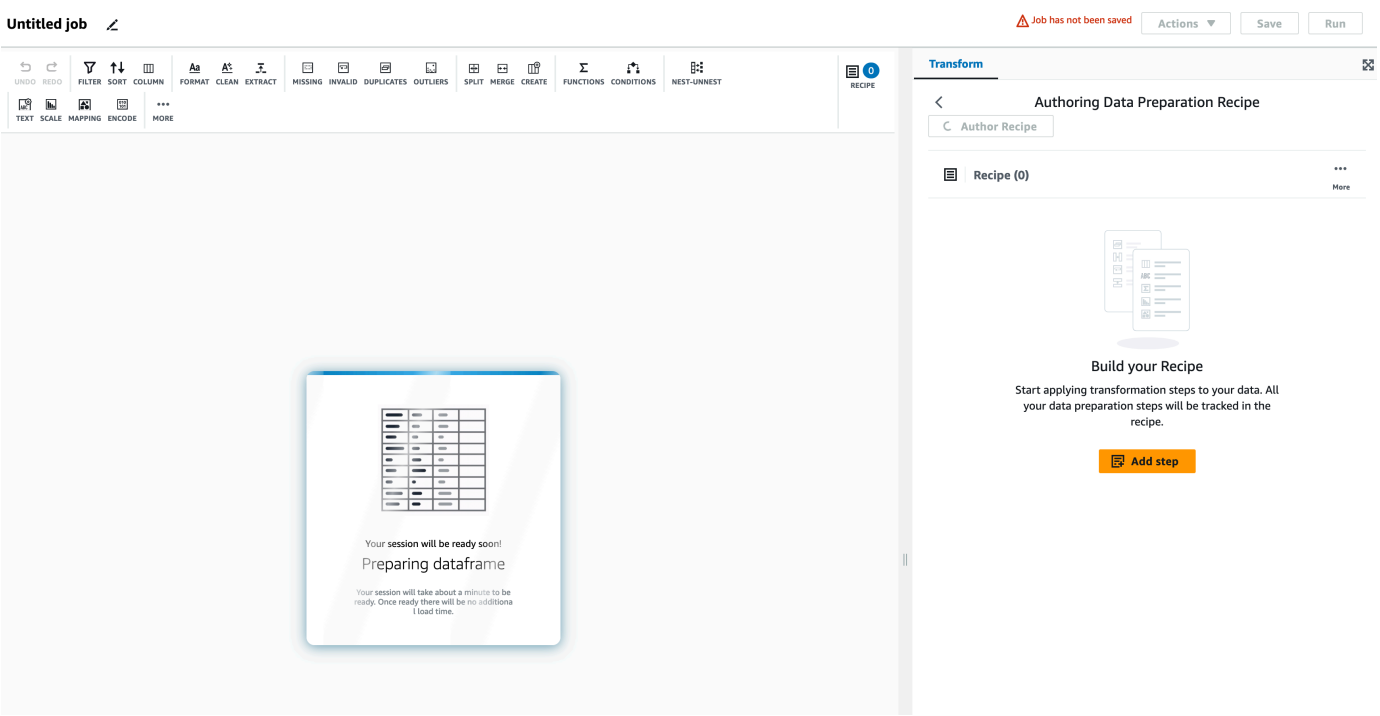


데이터 미리 보기 패널에서 톱니바퀴 아이콘을 선택하여 이 작업을 수행할 수 있습니다.

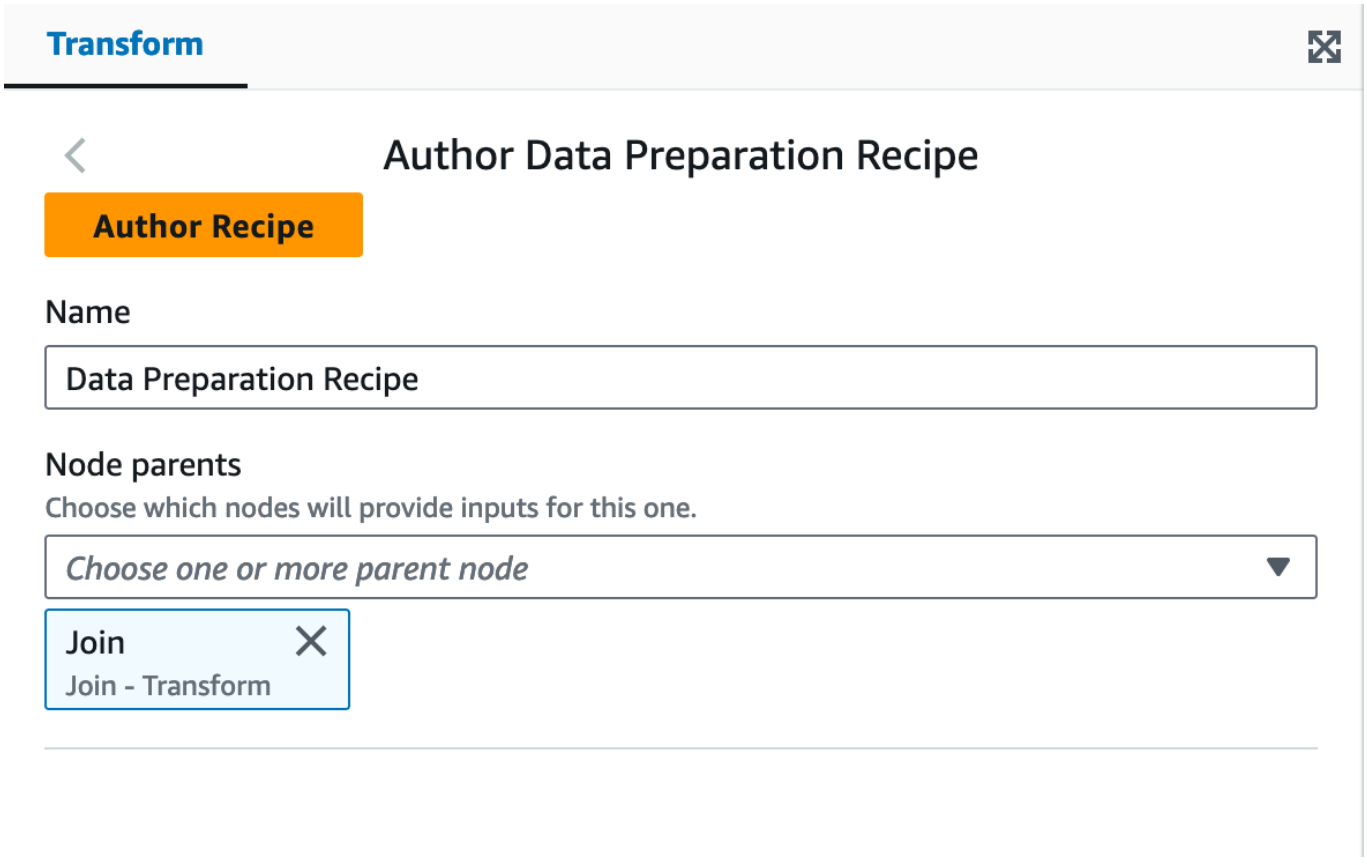


AWS Glue Studio에서 데이터 준비 레시피 노드 작성:

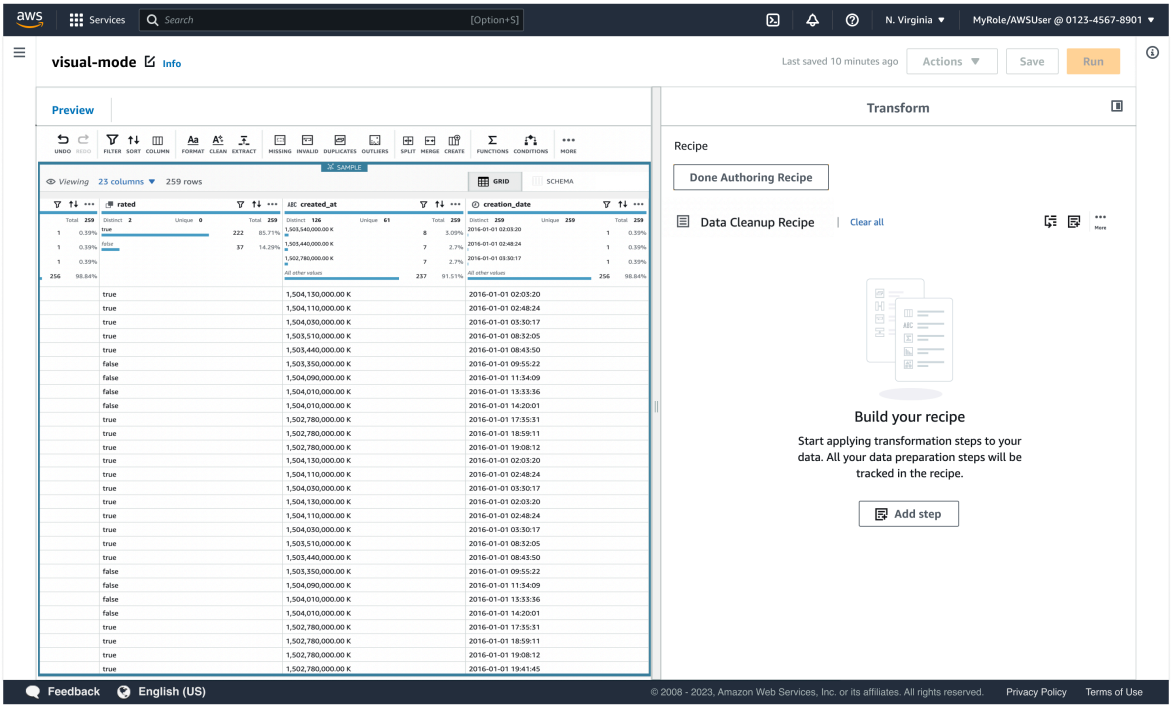
1. 데이터 준비 레시피 변환을 작업 캔버스에 추가합니다. 변환은 데이터 소스 노드 상위 항목에 연결되어야 합니다. 데이터 준비 레시피 노드를 추가하면 노드가 적절한 라이브러리와 함께 다시 시작되고 데이터 프레임이 준비되는 것을 볼 수 있습니다.



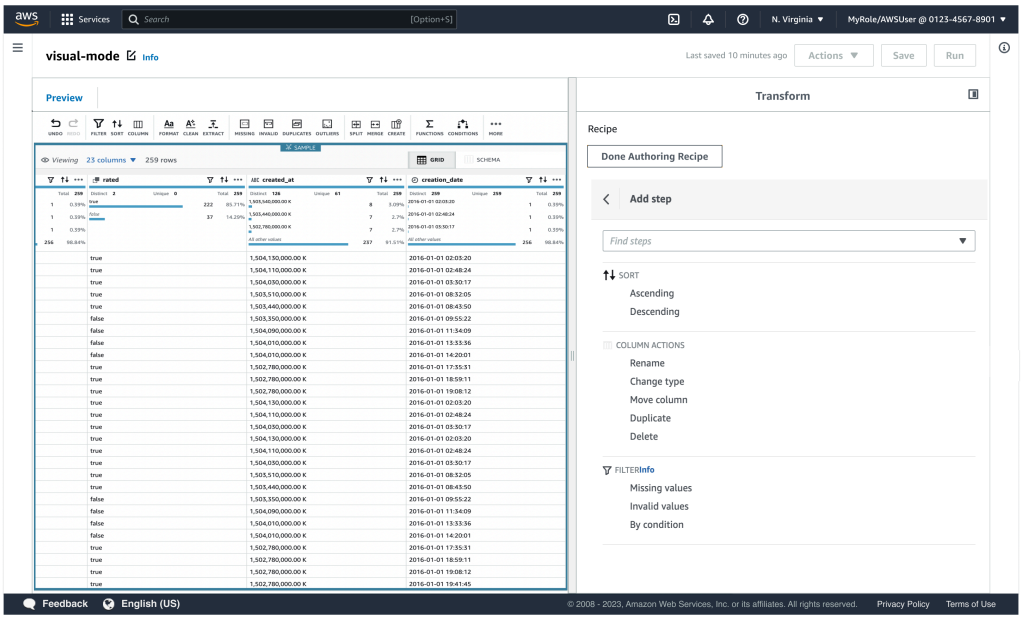
2. 데이터 미리 보기 세션이 준비되면 이전 단계가 적용된 데이터가 화면 하단에 표시됩니다.
3. 레시피 작성을 선택합니다. 이렇게 하면 AWS Glue Studio에서 새 레시피를 시작할 수 있습니다.



4. 작업 캔버스 오른쪽에 있는 변환 패널에서 데이터 준비 레시피의 이름을 입력합니다.
5. 왼쪽의 캔버스는 데이터의 그리드 보기로 대체됩니다. 오른쪽의 변환 패널이 변경되어 레시피 단계가 표시됩니다. 단계 추가를 선택하여 레시피에 첫 번째 단계를 추가합니다.

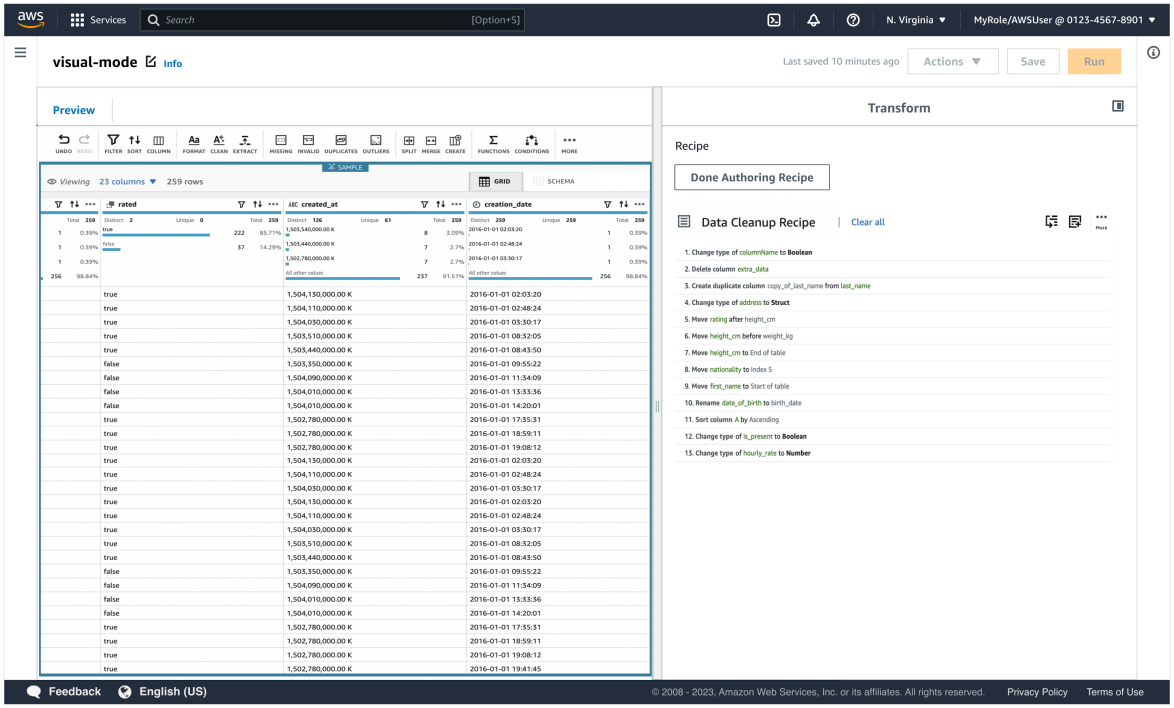


6. 변환 패널에서 정렬하고, 열에 대한 작업을 수행하고, 값을 필터링합니다. 예를 들어 열 이름 변경을 선택합니다.



7. 오른쪽의 변환 패널에서 열 이름 변경 옵션을 사용하여 이름을 변경할 소스 열을 선택하고 새 열 이름을 입력할 수 있습니다. 완료하면 적용을 선택합니다.

   각 단계를 미리 보고, 단계를 취소하고, 단계를 재정렬하고, 필터, 정렬, 분할, 병합 등과 같은 작업 아이콘을 사용할 수 있습니다. 데이터 그리드에서 작업을 수행하면 변환 패널의 레시피에 단계가 추가됩니다.

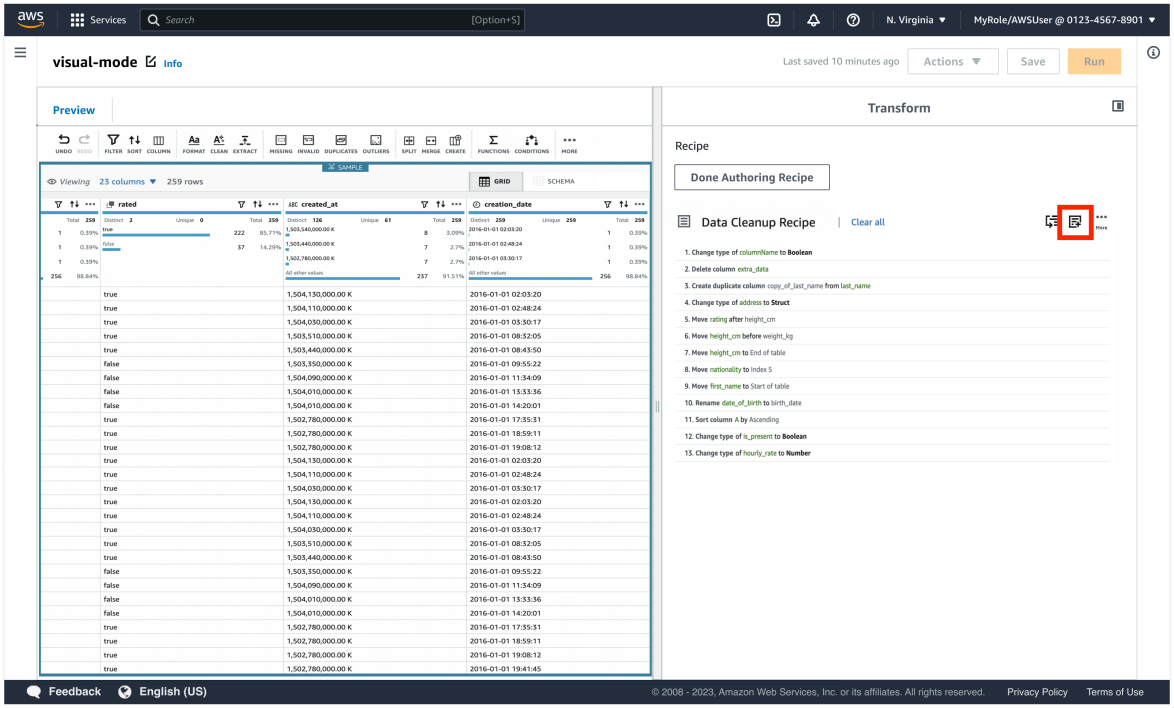


변경이 필요한 경우 미리 보기 창에서 각 단계의 결과를 미리 보고, 단계를 실행 취소하고, 단계를 재정렬하여 변경할 수 있습니다. 예:

- 단계 실행 취소/다시 실행 – 실행 취소 아이콘을 선택하여 단계를 실행 취소합니다. 다시 실행 아이콘을 선택하면 단계를 반복할 수 있습니다.

- 단계 순서 변경 단계 - 단계 순서를 변경하면 AWS Glue Studio에서 각 단계를 검증하고 단계가 유효하지 않은지 알려줍니다.

8. 단계를 적용하면 변환 패널에 레시피의 모든 단계가 표시됩니다. 모든 단계를 지우고 다시 시작하고, 추가 아이콘을 선택하여 단계를 더 추가하거나, 레시피 작성 완료를 선택할 수 있습니다.



9. 화면 오른쪽 상단에서 저장을 선택합니다. 작업을 저장할 때까지 레시피 단계는 저장되지 않습니다.

AWS Glue Studio에서 AWS Glue DataBrew 레시피 가져오기

AWS Glue DataBrew에서 레시피는 데이터 변환 단계 세트입니다. AWS Glue DataBrew 레시피에서는 이미 읽은 데이터를 변환하는 방법을 규정하지만, 데이터를 읽는 위치와 방법, 그리고 데이터를 쓰는 방법과 위치에 대해서는 설명하지 않습니다. 레시피는 AWS Glue Studio의 소스 및 대상 노드에서 구성됩니다. 레시피에 대한 자세한 내용은 Creating and using AWS Glue DataBrew recipes를 참조하세요.

AWS Glue Studio에서 AWS Glue DataBrew 레시피를 사용하려면 먼저 AWS Glue DataBrew에서 레시피를 생성합니다. 사용할 레시피가 이미 있으면 이 단계를 건너뛸 수 있습니다.

AWS Glue DataBrew에 대한 IAM 권한

이 주제에서는 IAM 관리자가 데이터 준비 레시피 변환에 대한 AWS Identity and Access Management(IAM) 정책에서 사용할 수 있는 작업과 리소스를 이해하는 데 도움이 되는 정보를 제공합니다.

AWS Glue에서 보안에 대한 자세한 내용은 액세스 관리를 참조하세요.

> **Note**
>
> 다음 표에는 기존 AWS Glue DataBrew 레시피를 가져올 때 사용자에게 필요한 권한이 나와 있습니다.

데이터 준비 레시피 변환 작업

작업	설명
`databrew:ListRecipes`	AWS Glue DataBrew 레시피를 검색할 수 있는 권한을 부여합니다.
`databrew:ListRecipeVersions`	AWS Glue DataBrew 레시피 버전을 검색할 수 있는 권한을 부여합니다.
`databrew:DescribeRecipe`	AWS Glue DataBrew 레시피 설명을 검색할 수 있는 권한을 부여합니다.

이 기능에 액세스하는 데 사용하는 역할에는 여러 AWS Glue DataBrew 작업을 허용하는 정책이 있어야 합니다. 필요한 작업이 포함된 `AWSGlueConsoleFullAccess` 정책을 사용하거나 역할에 다음 인라인 정책을 추가하여 이를 달성할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "databrew:ListRecipes",
                "databrew:ListRecipeVersions",
```

```
                "databrew:DescribeRecipe"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

데이터 준비 레시피 변환을 사용하려면 권한 정책에 IAM:PassRole 작업을 추가해야 합니다.

필요한 추가 권한

작업	설명
iam:PassRole	사용자가 승인된 역할을 전달할 수 있도록 IAM에 권한을 부여합니다.

이 권한이 없으면 다음과 같은 오류가 발생합니다.

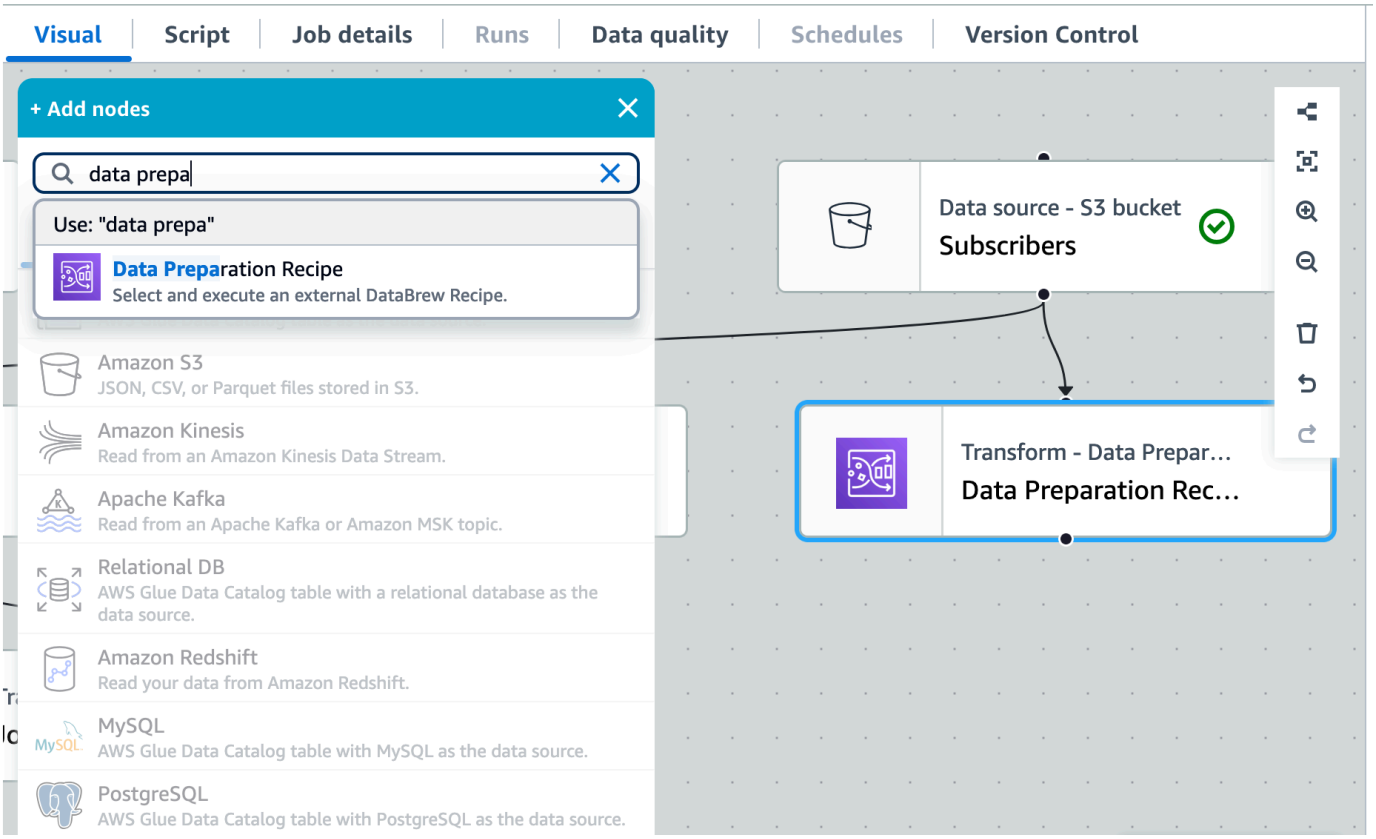
```
"errorCode": "AccessDenied"
"errorMessage": "User: arn:aws:sts::account_id:assumed-role/AWSGlueServiceRole is not
authorized to perform: iam:PassRole on resource: arn:aws:iam::account_id:role/service-
role/AWSGlueServiceRole
because no identity-based policy allows the iam:PassRole action"
```

AWS Glue DataBrew 레시피 가져오기

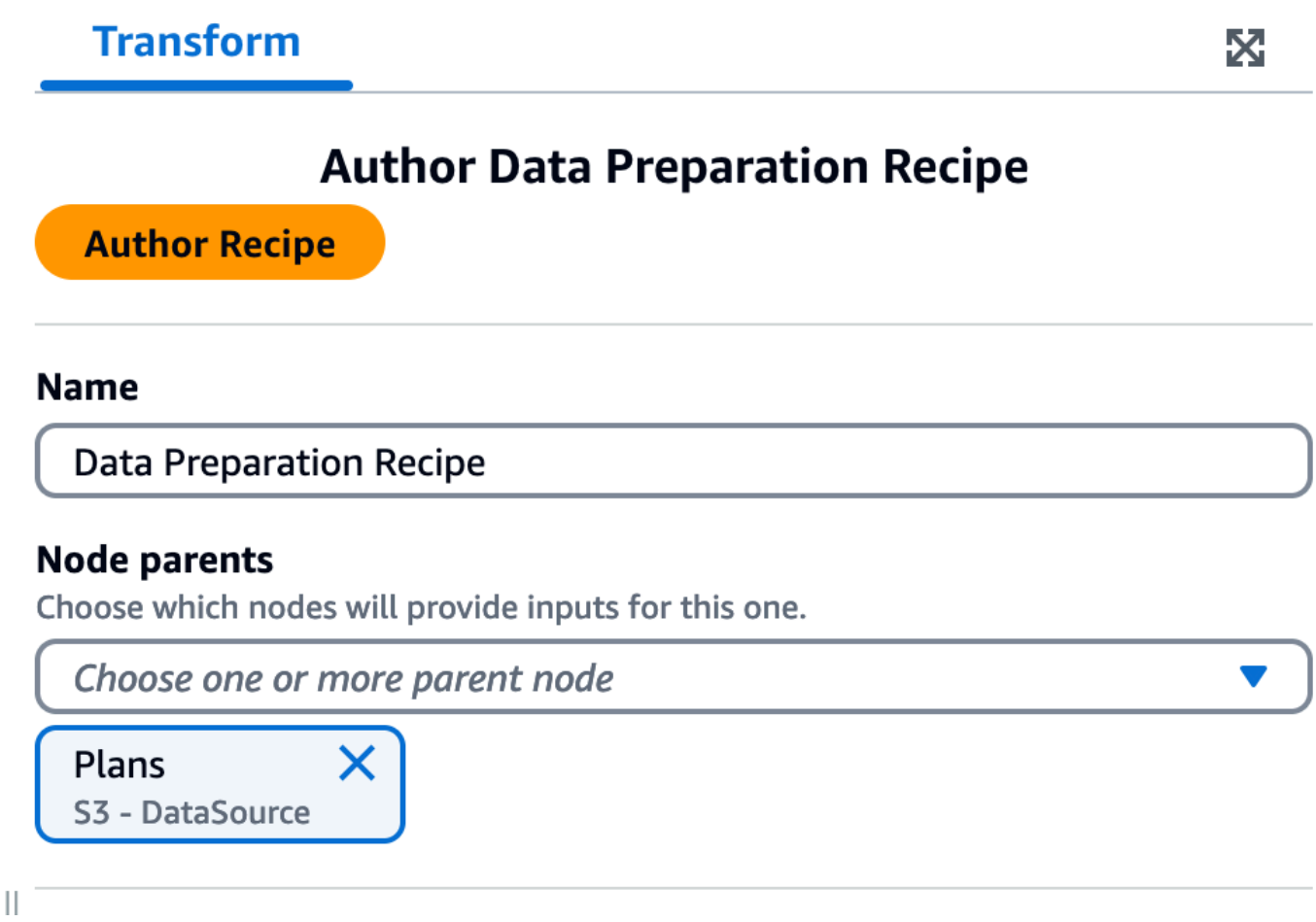
AWS Glue DataBrew 레시피를 가져와 AWS Glue Studio에서 사용하기:

기존 데이터 준비 레시피 노드가 있고 AWS Glue Studio에서 레시피 단계를 직접 편집하려면 AWS Glue Studio 작업으로 레시피 단계를 가져와야 합니다.

1. AWS Glue Studio에서 데이터 소스와 함께 AWS Glue 작업을 시작합니다.
2. 데이터 준비 레시피 노드를 작업 캔버스에 추가합니다.

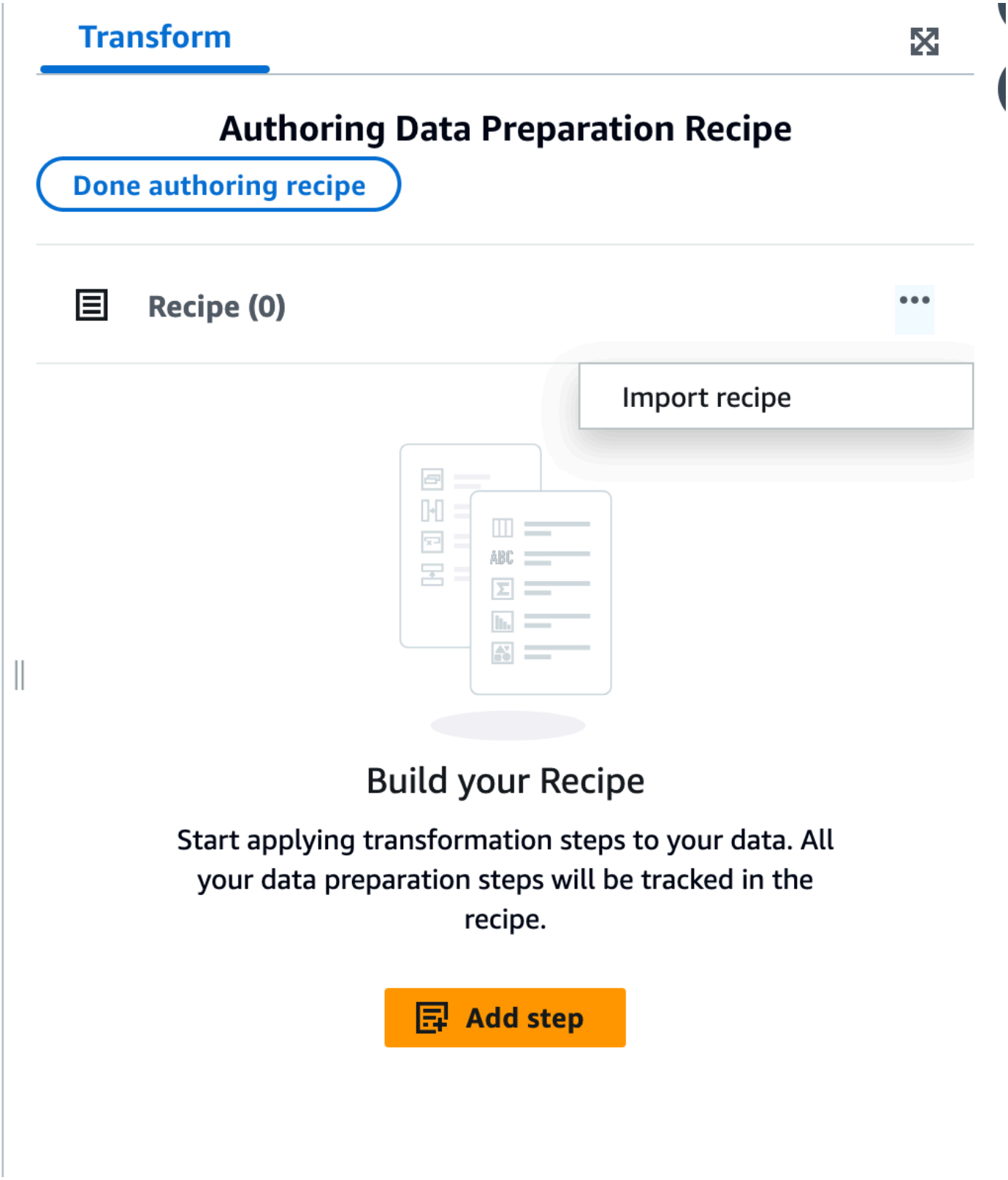


3. 변환 패널에 레시피의 이름을 입력합니다.
4. 캔버스에서 사용 가능한 노드를 드롭다운 목록에서 선택하여 하나 이상의 상위 노드를 선택합니다.
5. 레시피 작성을 선택합니다. 작성자 레시피가 회색이면 노드 상위를 선택하고 데이터 미리 보기 세션이 완료될 때까지 사용할 수 없습니다.

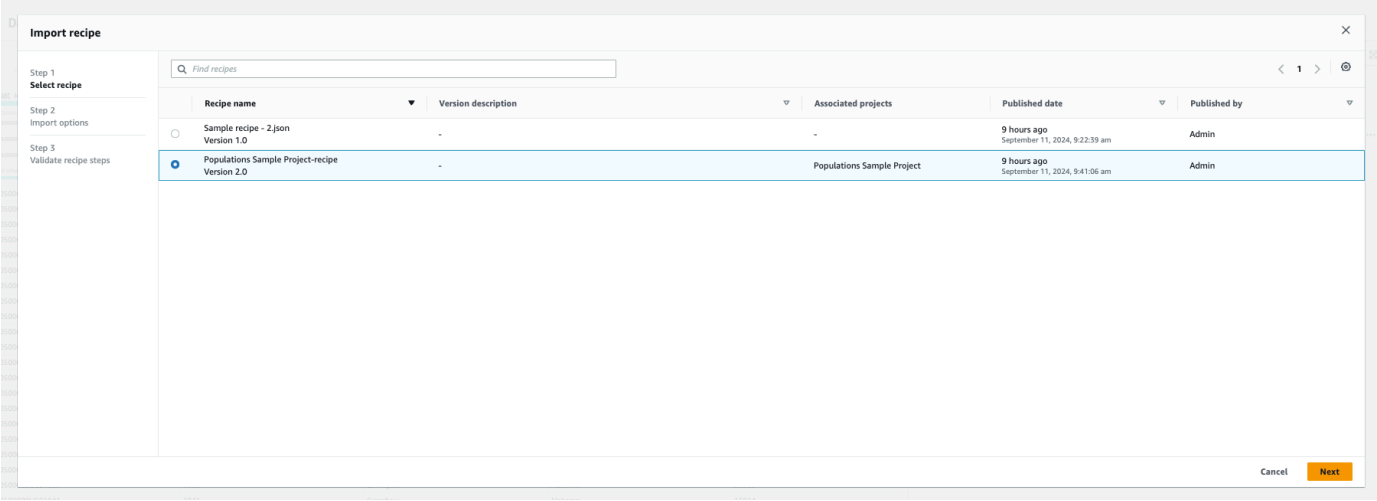


6. DataFrame이 로드되고 소스 데이터에 대한 자세한 정보가 표시됩니다.

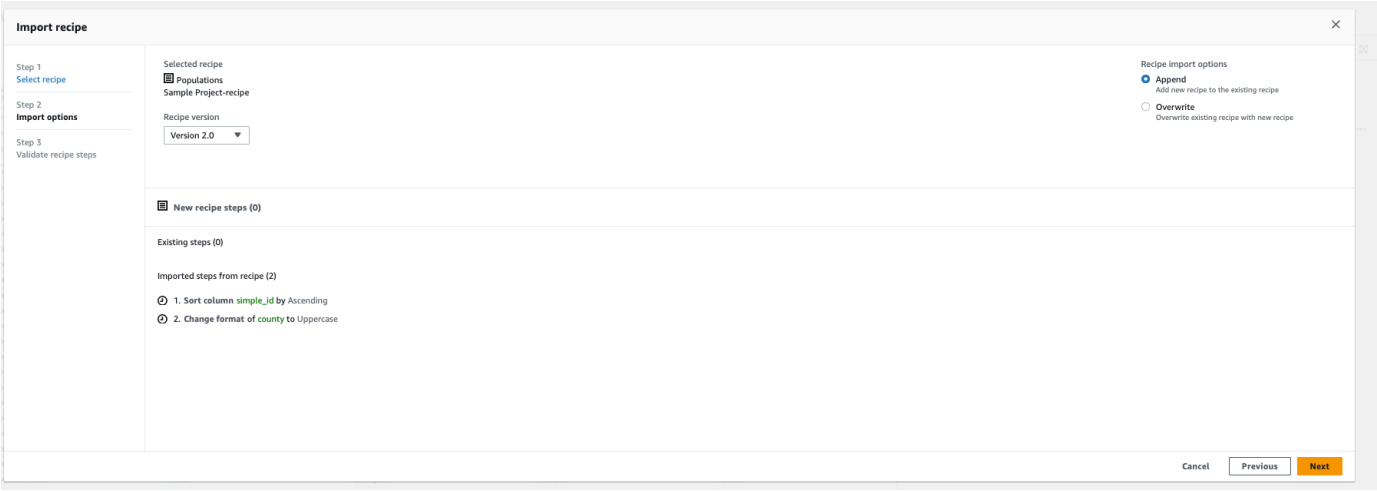
   추가 작업 아이콘을 선택하고 레시피 가져오기를 선택합니다.



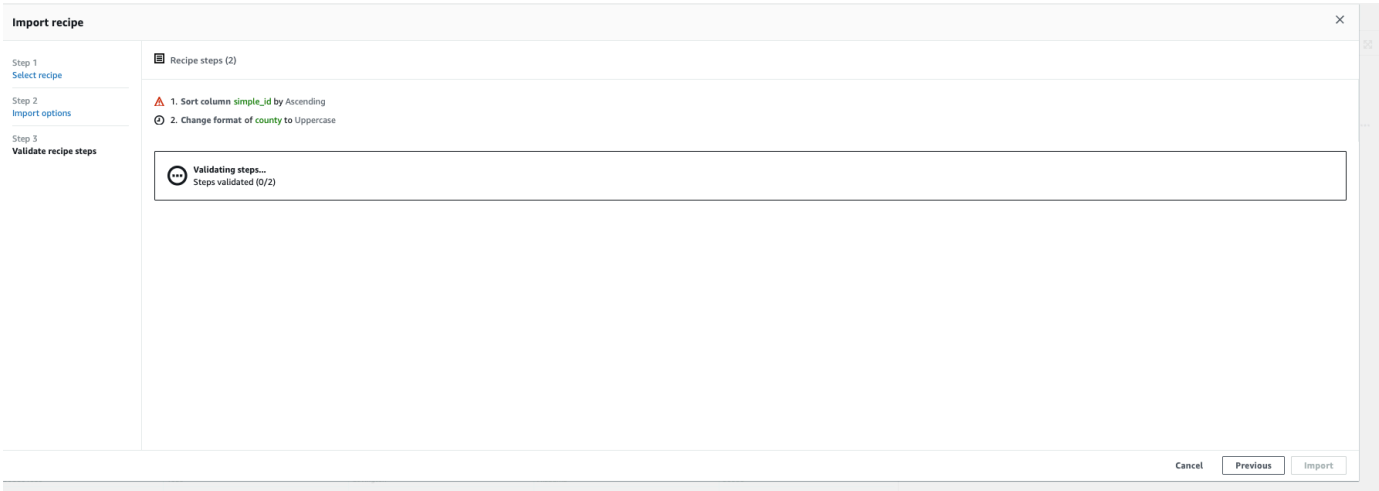
7. 레시피 가져오기 마법사를 사용하여 단계를 완료합니다. 1단계에서 레시피를 검색하고 선택한 후, 다음을 선택합니다.

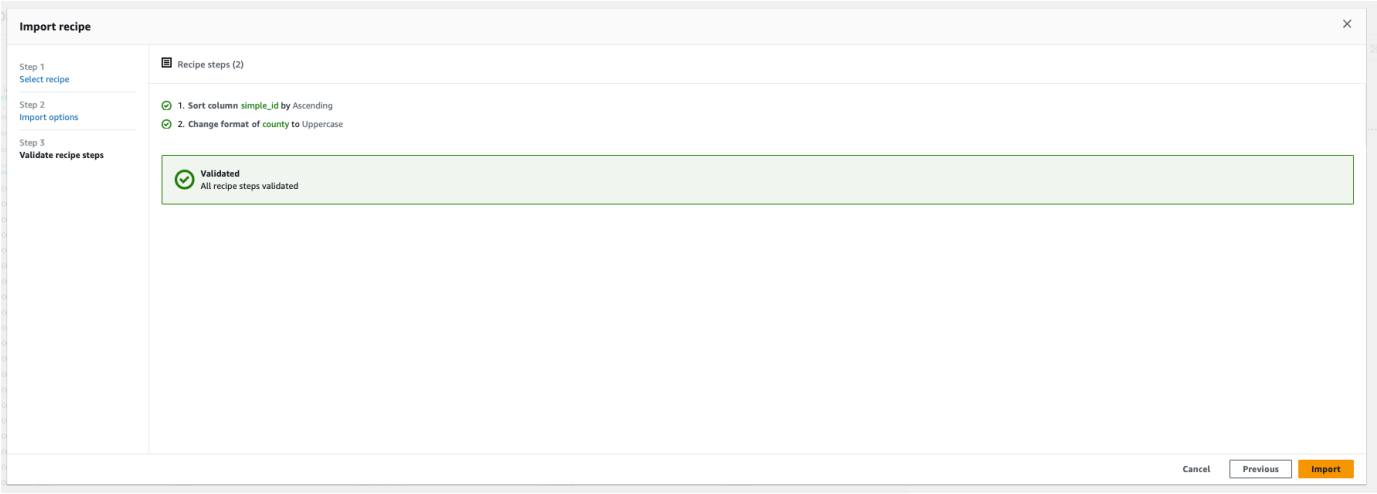


8. 2단계에서 가져오기 옵션을 선택합니다. 기존 레시피에 새 레시피를 추가하거나 기존 레시피를 덮어쓰도록 선택할 수 있습니다. Next(다음)를 선택합니다.



9. 3단계에서 레시피 단계를 검증합니다. AWS Glue DataBrew 레시피를 가져온 후에는 AWS Glue Studio에서 직접 이 레시피를 편집할 수 있습니다.





10. 이후에는 AWS Glue 작업의 일부로 단계를 가져오게 됩니다. 필요한 경우 작업 세부 정보 탭에서 작업 이름 지정 및 할당된 용량 조정과 같은 필요한 구성 변경을 수행합니다. 저장을 선택하여 작업과 레시피를 저장합니다.

> **Note**
>
> JOIN, UNION, GROUP_BY, PIVOT, UNPIVOT, TRANSPOSE는 레시피 가져오기에 지원되지 않으며, 레시피 작성 모드에서도 사용할 수 없습니다.

11. 필요한 경우 다른 변환 노드를 추가하여 작업 작성을 완료하고 데이터 대상 노드를 추가할 수 있습니다.

    레시피를 가져온 후 단계를 재정렬하면 AWS Glue에서 해당 단계에 대한 검증을 수행합니다. 예를 들어 이름을 변경한 다음 열을 삭제하고, 삭제 단계를 맨 위로 이동한 경우, 이름 변경 단계는 유효하지 않게 됩니다. 이후 단계를 편집하여 검증 오류를 수정할 수 있습니다.

AWS Glue DataBrew에서 AWS Glue Studio로 마이그레이션

AWS Glue DataBrew에 레시피가 있는 경우 다음 체크리스트를 사용하여 레시피를 AWS Glue Studio로 마이그레이션합니다.

다음을 수행하려는 경우...	수행할 작업
사용자가 AWS Glue DataBrew 레시피, 레시피 버전 및 레시피 설명을 검색할 수 있습니다.	역할이 필요한 작업에 액세스할 수 있도록 허용하는 정책에 IAM 권한을 추가합니다. AWS Glue DataBrew에 대한 IAM 권한 섹션을 참조하세요.

다음을 수행하려는 경우…	수행할 작업
기존 AWS Glue DataBrew 레시피를 AWS Glue Studio로 가져옵니다.	AWS Glue DataBrew 레시피 가져오기 단원의 단계를 따르세요.
JOIN 및 UNION을 사용하여 레시피를 가져옵니다.	UNION 및 JOIN 변환이 있는 레시피는 지원되지 않습니다. 데이터 준비 레시피 노드 전후에 AWS Glue Studio에서 조인 및 유니온 변환을 사용합니다.

스키마 변경을 사용하여 데이터 속성 키 다시 매핑

스키마 변경 변환은 소스 데이터 속성 키를 대상 데이터에 대해 구성된 원하는 항목으로 다시 매핑합니다. 스키마 변경 변환 노드에서는 다음을 수행할 수 있습니다.

- 여러 데이터 속성 키의 이름을 변경합니다.
- 새 데이터 유형이 지원되고 두 데이터 유형 사이에 변환 경로가 있는 경우 데이터 속성 키의 데이터 유형을 변경합니다.
- 삭제할 데이터 속성 키를 표시하여 데이터 속성 키의 하위 집합을 선택합니다.

필요한 경우 추가 스키마 변경 노드를 작업 다이어그램에 추가할 수도 있습니다. 예를 들어 추가 데이터 소스를 수정하거나 조인 변환을 따릅니다.

소수 데이터 유형에서 스키마 변경 사용

소수 데이터 유형에서 스키마 변경 변환을 사용하는 경우 스키마 변경 변환은 정밀도를 (10,2)의 기본값으로 수정합니다. 이를 수정하고 사용 사례의 정밀도를 설정하려면 SQL 쿼리 변환을 사용하고 열을 특정 정밀도로 캐스팅할 수 있습니다.

예를 들어 소수 유형의 'DecimalCol'이라는 입력 열이 있고 특정 정밀도가 (18,6)인 'OutputDecimalCol'이라는 출력 열에 다시 매핑하려는 경우 다음을 수행합니다.

1. 스키마 변경 변환 후 후속 SQL 쿼리 변환을 추가합니다.
2. SQL 쿼리 변환에서 SQL 쿼리를 사용하여 다시 매핑된 열을 원하는 정밀도로 캐스팅합니다. SQL 쿼리는 다음과 같습니다.

```
SELECT col1, col2, CAST(DecimalCol AS DECIMAL(18,6)) AS OutputDecimalCol
```

```
FROM __THIS__
```

위의 SQL 쿼리에서:

- `col1` 및 `col2`는 수정 없이 전달하려는 데이터의 다른 열입니다.
- `DecimalCol`은 입력 데이터의 원래 열 이름입니다.
- `CAST(DecimalCol AS DECIMAL(18,6))`는 `DecimalCol`을 숫자 18자리 및 소수 6자리의 정밀도인 소수 유형으로 캐스팅합니다.
- `AS OutputDecimalCol`은 캐스팅된 열의 이름을 `OutputDecimalCol`으로 바꿉니다.

SQL 쿼리 변환을 사용하면 스키마 변경 변환으로 설정된 기본 정밀도를 재정의하고 소수 열을 원하는 정밀도로 명시적으로 캐스팅할 수 있습니다. 이 접근 방식을 사용하면 후속 SQL 쿼리 변환을 통해 소수 열의 정밀도 요구 사항을 처리하는 동시에 데이터 이름 변경 및 구조 조정을 위해 스키마 변경 변환을 활용할 수 있습니다.

작업에 스키마 변경 변환 추가

> **Note**
>
> 스키마 변경 변환은 대소문자를 구분하지 않습니다.

작업 다이어그램에 스키마 변경 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 스키마 변경을 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. 노드 속성 패널에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 노드 속성 패널에서 변환 탭을 선택합니다.
4. 입력 스키마를 수정합니다.
   - 데이터 속성 키의 이름을 바꾸려면 [대상 키(Target key)] 필드에 키의 새 이름을 입력합니다.
   - 데이터 속성 키의 데이터 유형을 변경하려면 [데이터 유형(Data type)] 목록에서 키의 새 데이터 유형을 선택합니다.
   - 대상 스키마에서 데이터 속성 키를 제거하려면 해당 키의 [삭제(Drop)] 확인란을 선택합니다.

5. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.
6. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

중복 삭제 사용

중복 삭제 변환은 두 가지 옵션을 제공하여 데이터 소스에서 행을 제거합니다. 완전히 동일한 중복 행을 제거하거나 일치시킬 필드를 선택하고 선택한 필드를 기반으로 해당 행만 제거하도록 선택할 수 있습니다.

예를 들어 이 데이터 세트에는 일부 행에서는 모든 값이 다른 행과 완전히 동일하고, 또 다른 행에서는 값이 같거나 다른 중복 행이 있습니다.

열	명칭	이메일	Age	State	참고
1	Joy	joy@gmail	33	NY	
2	Tim	tim@gmail	45	OH	
3	Rose	rose@gmail	23	NJ	
4	Tim	tim@gmail	42	OH	
5	Rose	rose@gmail	23	NJ	
6	Tim	tim@gmail	42	OH	이것은 중복된 행이며 4번째 행의 모든 값과 완전히 일치합니다.
7	Rose	rose@gmail	23	NJ	이것은 중복된 행이며 5번

열	명칭	이메일	Age	State	참고
					째 행의 모든 값과 완전히 일치합니다.

전체 행을 일치시키도록 선택하면 6번째 행과 7번째 행이 데이터 세트에서 제거됩니다. 이제 데이터 세트는 다음과 같습니다.

열	명칭	이메일	Age	State
1	Joy	joy@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rose	rose@gmail	23	NJ
4	Tim	tim@gmail	42	OH
5	Rose	rose@gmail	23	NJ

키를 지정하기로 선택한 경우 'name' 및 'email'에서 일치하는 행을 제거하도록 선택할 수 있습니다. 이렇게 하면 데이터 세트의 '중복 행'을 더 세밀하게 제어할 수 있습니다. 'name' 및 'email'을 지정하면 이제 데이터 세트는 다음과 같습니다.

열	명칭	이메일	Age	State
1	Joy	joy@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rose	rose@gmail	23	NJ

다음 사항에 유의하세요.

- 중복 행으로 인식하기 위해 값은 대소문자를 구분합니다. 행의 모든 값은 동일한 대소문자를 사용해야 합니다. 이는 선택한 옵션(전체 행 일치 또는 키 지정)에 적용됩니다.

- 모든 값은 문자열로 읽습니다.
- 중복 삭제 변환은 Spark dropDuplicates 명령을 활용합니다.
- 중복 삭제 변환을 사용하는 경우 첫 번째 행은 유지되고 다른 행은 삭제됩니다.
- 중복 삭제 변환은 데이터 프레임의 스키마를 변경하지 않습니다. 키를 지정하도록 선택한 경우 모든 필드가 그 결과로 나타나는 데이터 프레임에서 유지됩니다.

SelectFields를 사용하여 대부분의 데이터 속성 키 제거

SelectFields 변환을 사용 사용하여 데이터 집합에서 데이터 속성 키의 하위 집합을 생성할 수 있습니다. 유지하려는 데이터 속성 키를 지정하면 나머지는 데이터 집합에서 제거됩니다.

> **Note**
>
> SelectFields는 대/소문자를 구분합니다. 필드를 선택하기 위해 대/소문자를 구분하지 않는 방법이 필요한 경우 ApplyMapping을 사용합니다.

작업 다이어그램에 SelectFields 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 SelectFields를 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 노드 세부 정보 패널에서 [변환(Transform)] 탭을 선택합니다.
4. 머리글 [SelectFields] 아래에서 유지하려는 데이터 집합의 데이터 속성 키를 선택합니다. 선택하지 않은 데이터 속성 키는 데이터 집합에서 모두 삭제됩니다.

   열 머리글 [필드(Field)] 옆에 있는 확인란을 선택하여 데이터 집합의 모든 데이터 속성 키를 자동으로 선택할 수도 있습니다. 그런 다음 개별 데이터 속성 키를 선택 취소하여 데이터 집합에서 제거할 수 있습니다.
5. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.

6. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

DropFields를 사용하여 대부분의 데이터 속성 키 유지

DropFields 변환을 사용 사용하여 데이터 집합에서 데이터 속성 키의 하위 집합을 생성할 수 있습니다. 데이터 집합에서 제거하려는 데이터 속성 키를 지정하면 나머지 키는 유지됩니다.

> **Note**
>
> DropFields 변환은 대/소문자를 구분합니다. 필드를 선택할 때 대소문자를 구분하지 않는 방법이 필요한 경우 스키마 변경을 사용합니다.

작업 다이어그램에 DropFields 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 DropFields를 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 노드 세부 정보 패널에서 [변환(Transform)] 탭을 선택합니다.
4. 머리글 [DropFields] 아래에서 데이터 원본에서 삭제할 데이터 속성 키를 선택합니다.

   열 머리글 [필드(Field)] 옆에 있는 확인란을 선택하여 데이터 집합의 모든 데이터 속성 키를 자동으로 선택할 수도 있습니다. 그런 다음 개별 데이터 속성 키를 선택 취소하여 데이터 집합에 보존할 수 있습니다.
5. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.
6. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을

처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

데이터 집합에서 필드 이름 바꾸기

RenameField 변환을 사용하여 데이터 집합의 개별 속성 키 이름을 변경할 수 있습니다.

> **Note**
>
> RenameField는 대/소문자를 구분합니다. 대/소문자를 구분하지 않는 변환이 필요한 경우 ApplyMapping을 사용합니다.

> **Tip**
>
> 스키마 변경 변환을 사용하는 경우 단일 변환으로 데이터 세트에 있는 여러 데이터 속성 키의 이름을 바꿀 수 있습니다.

작업 다이어그램에 RenameField 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 RenameField를 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. [변환(Transform)] 탭을 선택합니다.
4. 머리글 [데이터 필드(Data field)] 아래에서 소스 데이터의 속성 키를 선택한 다음 [새 필드 이름(New field name)] 필드에 새 이름을 입력합니다.
5. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.
6. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을

처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

Spigot을 사용하여 데이터 집합 샘플링

작업에서 수행한 변환을 테스트하기 위해 데이터 샘플을 가져와 변환이 의도한 대로 작동하는지 확인할 수 있습니다. Spigot 변환은 데이터 집합의 레코드 하위 집합을 Amazon S3 버킷의 JSON 파일에 기록합니다. 데이터 샘플링 방법은 파일 시작 부분의 특정 레코드 수 또는 레코드 선택에 사용되는 확률 요소일 수 있습니다.

작업 다이어그램에 Spigot 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 Spigot을 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 노드 세부 정보 패널에서 [변환(Transform)] 탭을 선택합니다.
4. Amazon S3 경로를 입력하거나 [S3 찾아보기(Browse S3)]를 선택하여 Amazon S3에서 위치를 선택합니다. 작업이 데이터 샘플이 포함된 JSON 파일을 작성하는 위치입니다.
5. 샘플링 방법에 대한 정보를 입력합니다. 데이터 집합의 시작부터 쓸 [레코드 수(Number of records)]에 대한 값과 지정된 레코드를 선택하는 [확률 임계값(Probability threshold)](최대값이 1인 십진수 값으로 입력됨)을 지정할 수 있습니다.

   예를 들어 데이터 집합에서 처음 50개의 레코드를 쓰려면 [레코드 수(Number of records)]를 50으로 설정하고 [확률 임계값(Probability threshold)]을 1(100%)로 설정합니다.

데이터 집합 조인

Join 변환을 사용하면 2개의 데이터 집합을 하나로 결합 할 수 있습니다. 비교할 각 데이터 집합의 스키마에 키 이름을 지정합니다. 출력 `DynamicFrame`에는 키가 조인 조건을 충족하는 행이 포함됩니다. 조인 조건을 충족하는 각 데이터 집합의 행은 두 데이터 집합에서 찾은 모든 열을 포함하는 출력 `DynamicFrame`의 단일 행으로 결합됩니다.

작업 다이어그램에 Join 변환 노드를 추가하려면

1. 사용 가능한 데이터 원본이 하나만 있는 경우 작업 다이어그램에 새 데이터 원본 노드를 추가해야 합니다.
2. 조인을 위한 소스 노드 중 하나를 선택합니다. 리소스 패널을 열고 Join을 선택하여 작업 다이어그램에 새 변환을 추가합니다.
3. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다.
4. 조인에 대한 입력을 제공하는 2개의 데이터 집합이 있도록 [노드 속성(Node properties)] 탭의 머리글 [노드 상위 항목(Node parents)] 아래에서 상위 노드를 추가합니다. 상위 항목은 데이터 원본 노드 또는 변환 노드일 수 있습니다.

   > **Note**
   >
   > 조인은 2개의 상위 노드만 가질 수 있습니다.

5. [변환(Transform)] 탭을 선택합니다.

   충돌하는 키 이름이 있다는 메시지가 나타나면 다음 중 하나를 수행할 수 있습니다.

   - 작업 다이어그램에 ApplyMapping 변환 노드를 자동으로 추가하려면 [해결(Resolve it)]을 선택합니다. ApplyMapping 노드는 다른 데이터 집합의 키와 이름이 같은 데이터 집합의 모든 키에 접두사를 추가합니다. 예를 들어 기본값인 **`right`**를 사용하면 왼쪽 데이터 집합의 키와 이름이 같은 오른쪽 데이터 집합의 모든 키 이름이 `(right)`*`key name`*으로 바뀝니다.
   - 충돌하는 키를 제거하거나 이름을 바꾸려면 작업 다이어그램 앞부분에 변환 노드를 수동으로 추가합니다.
6. [조인 유형(Join type)] 목록에서 조인 유형을 선택합니다.

   - [내부 조인(Inner join)]: 조인 조건에 따라 모든 일치 항목에 대해 두 데이터 집합의 열이 있는 행을 반환합니다. 조인 조건을 만족하지 않는 행은 반환되지 않습니다.
   - [왼쪽 조인(Left join)]: 왼쪽 데이터 집합의 모든 행과 조인 조건을 충족하는 오른쪽 데이터 집합의 행만.
   - [오른쪽 조인(Right join)]: 오른쪽 데이터 집합의 모든 행과 조인 조건을 충족하는 왼쪽 데이터 집합의 행만.
   - [외부 조인(Outer join)]: 두 데이터 집합의 모든 행.
   - [왼쪽 세미 조인(Left semi join)]: 조인 조건에 따라 오른쪽 데이터 집합과 일치하는 왼쪽 데이터 집합의 모든 행.

- [왼쪽 안티 조인(Left semi join)]: 조인 조건에 따라 오른쪽 데이터 집합과 일치하지 않는 왼쪽 데이터 집합의 모든 행.

7. [변환(Transform)] 탭의 [조인 조건(Join conditions)] 아래에서 [조건 추가(Add condition)]를 선택합니다. 각 데이터 집합에서 비교할 속성 키를 선택합니다. 비교 연산자의 왼쪽에 있는 속성 키를 왼쪽 데이터 집합이라고 하고 오른쪽에 있는 속성 키를 오른쪽 데이터 집합이라고 합니다.

   더 복잡한 조인 조건의 경우 [조건 추가(Add condition)]를 두 번 이상 선택하여 일치하는 키를 추가할 수 있습니다. 실수로 조건을 추가한 경우 삭제 아이콘 ( ) 을 클릭하여 제거할 수 있습니다.

8. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.

9. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

조인 출력 스키마의 예를 들어 다음 속성 키를 사용하여 두 데이터 집합 간의 조인을 고려합니다.

```
Left: {id, dept, hire_date, salary, employment_status}
Right: {id, first_name, last_name, hire_date, title}
```

조인은 = 비교 연산자를 사용하여 `id` 및 `hire_date` 키에서 일치하도록 구성됩니다.

두 데이터 집합 모두 `id` 및 `hire_date` 키를 포함하므로 [해결(Resolve it)]을 선택하여 올바른 데이터 집합의 키에 접두사 **`right`**를 자동으로 추가합니다.

출력 스키마의 키는 다음과 같습니다.

```
{id, dept, hire_date, salary, employment_status,
(right)id, first_name, last_name, (right)hire_date, title}
```

집합을 사용하여 행 결합

스키마가 같은 둘 이상의 데이터 소스에서 행을 결합하려는 경우 집합(Union) 변환 노드를 사용합니다.

집합 변환에는 두 가지 유형이 있습니다.

1. 모두 - 모두를 적용하면 그 결과로 나타나는 집합에서 중복된 행을 제거하지 않습니다.
2. 고유 - 고유를 적용하면 그 결과로 나타나는 집합에서 중복된 행을 제거합니다.

집합과 조인 비교

집합을 사용하여 행을 결합합니다. 조인을 사용하여 열을 결합합니다.

시각적 ETL 캔버스에서 집합 변환 사용

1. 하나 이상의 데이터 소스를 추가하여 집합 변환을 수행합니다. 데이터 소스를 추가하려면 리소스 패널을 열고 소스 탭에서 데이터 소스를 선택합니다. 집합 변환을 사용하기 전에 집합과 관련된 모든 데이터 소스의 스키마와 구조가 동일한지 확인해야 합니다.
2. 집합 변환을 사용하여 결합하려는 데이터 소스가 둘 이상 있는 경우 집합 변환을 캔버스에 추가하여 집합 변환을 생성합니다. 캔버스에서 리소스 패널을 열고 '집합'을 검색합니다. 리소스 패널에서 변환 탭을 선택하고 집합 변환을 찾을 때까지 아래로 스크롤한 후 집합을 선택할 수도 있습니다.
3. 작업 캔버스에서 집합 노드를 선택합니다. 노드 속성 창에서 집합 변환에 연결할 상위 노드를 선택합니다.
4. AWS Glue에서는 집합 변환을 모든 데이터 소스에 적용할 수 있는지 확인하기 위해 호환성을 검사합니다. 데이터 소스의 스키마가 동일하면 작업이 허용됩니다. 데이터 소스의 스키마가 같지 않으면 유효하지 않음을 나타내는 오류 메시지가 표시됩니다. 'The input schemas of this union are not the same Consider using ApplyMapping to match the schemas.' 이 문제를 해결하려면 ApplyMapping 사용을 선택합니다.
5. 집합 유형을 선택합니다.
    1. 모두 - 기본적으로 모두 집합 유형이 선택됩니다. 이 경우 데이터 조합에 중복된 행이 있을 경우 행이 중복됩니다.
    2. 고유 - 그 결과로 나타나는 데이터 조합에서 중복된 행을 제거하려면 고유를 선택합니다.

SplitFields를 사용하여 데이터 집합을 2개로 분할

SplitFields 변환을 사용하면 입력 데이터 집합에서 일부 데이터 속성 키를 선택하여 하나의 데이터 집합에 넣고 선택되지 않은 키를 별도의 데이터 집합에 넣을 수 있습니다. 이 변환의 출력은 `DynamicFrames`의 컬렉션입니다.

> **Note**
>
> 출력을 대상 위치로 보내기 전에 SelectFromCollection 변환을 사용하여 `DynamicFrames` 컬렉션을 단일 `DynamicFrame`으로 변환해야 합니다.

SplitFields는 대/소문자를 구분합니다. 대/소문자를 구분하지 않는 속성 키 이름이 필요한 경우 ApplyMapping 변환을 상위 노드로 추가합니다.

작업 다이어그램에 SplitFields 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 SplitFields를 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. [변환(Transform)] 탭을 선택합니다.
4. 첫 번째 데이터 집합에 넣을 속성 키를 선택합니다. 선택하지 않는 키는 두 번째 데이터 집합에 배치됩니다.
5. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.
6. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.
7. 결과 데이터 집합을 처리하도록 SelectFromCollection 변환 노드를 구성합니다.

SelectFromCollection 변환의 개요

특정 변환에는 단일 데이터 집합 대신 여러 데이터 집합이 출력으로 포함됩니다(예: SplitFields). SelectFromCollection 변환은 데이터 집합(`DynamicFrames`의 배열)에서 하나의 데이터 집합(`DynamicFrame`)을 선택합니다. 변환에 대한 출력은 선택된 `DynamicFrame`입니다.

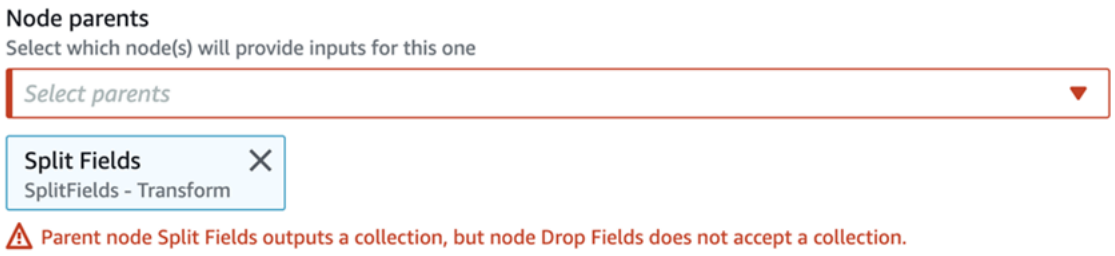
다음과 같이 DynamicFrames의 컬렉션을 생성하는 변환을 사용한 후에는 이 변환을 사용해야 합니다.

- 사용자 정의 코드 변환
- SplitFields

이러한 변환 후 작업 다이어그램에 SelectFromCollection 변환 노드를 추가하지 않으면 작업에 대한 오류가 발생합니다.

이 변환의 상위 노드는 DynamicFrames의 컬렉션을 반환하는 노드여야 합니다. Join 변환과 같이 단일 DynamicFrame을 반환하는 이 변환 노드의 상위 항목을 선택하면 작업에서 오류를 반환합니다.

마찬가지로, 단일 DynamicFrame을 입력으로 예상하는 변환의 상위 항목으로 작업 다이어그램에서 SelectFromCollection 노드를 사용하는 경우 작업에서 오류를 반환합니다.



SelectFromCollection을 사용하여 유지할 데이터 집합 선택

SelectFromCollection 변환을 사용하여 DynamicFrames의 컬렉션을 단일 DynamicFrame으로 변환합니다.

작업 다이어그램에 SelectFromCollection 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 SelectFromCollection을 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. [변환(Transform)] 탭을 선택합니다.
4. [프레임 인덱스(Frame index)] 머리글 아래에서 DynamicFrames의 컬렉션에서 선택하려는 DynamicFrame에 해당하는 배열 인덱스 번호를 선택합니다.

예를 들어 이 변환의 상위 노드가 SplitFields 변환인 경우 해당 노드의 [출력 스키마(Output schema)] 탭에서 각 `DynamicFrame`에 대한 스키마를 볼 수 있습니다. [출력 2(Output 2)]의 스키마와 연결된 `DynamicFrame`을 유지하려면 목록의 두 번째 값인 [프레임 인덱스(Frame index)] 값으로 **1**을 선택합니다.

선택하는 `DynamicFrame`만 출력에 포함됩니다.

5. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.
6. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

데이터 집합에서 누락된 값 찾기 및 채우기

FillMissingValues 변환을 사용하여 데이터 집합에서 누락된 값이 있는 레코드를 찾고 대치에 의해 결정된 값으로 새 필드를 추가할 수 있습니다. 입력 데이터 집합은 누락 값을 결정하는 기계 학습 모델을 훈련하는 데 사용됩니다. 증분 데이터 집합을 사용하는 경우 각 증분 집합은 기계 학습 모델의 훈련 데이터로 사용되므로 결과가 정확하지 않을 수 있습니다.

작업 다이어그램에서 FillMissingValues 변환 노드를 사용하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 FillMissingValues를 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. [변환(Transform)] 탭을 선택합니다.
4. [데이터 필드(Data field)]에서 누락된 값을 분석할 소스 데이터의 열 또는 필드 이름을 선택합니다.
5. (선택 사항) [새 필드 이름(New field name)] 필드에 분석된 필드의 예상 대체 값을 보유할 각 레코드에 추가된 필드의 이름을 입력합니다. 분석된 필드에 누락된 값이 없으면 분석된 필드의 값이 새 필드에 복사됩니다.

새 필드의 이름을 지정하지 않으면 기본 이름은 `_filled`가 추가된 분석된 열의 이름입니다. 예를 들어 [데이터 필드(Data field)]에 **`Age`**를 입력하고 [새 필드 이름(New field name)]에 값을 지정하지 않으면 **`Age_filled`**라는 새 필드가 각 레코드에 추가됩니다.

6. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.
7. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

데이터 집합 내의 키 필터링

Filter 변환을 사용하여 정규식을 기반으로 입력 데이터 집합의 레코드를 필터링하여 새 데이터 집합을 생성합니다. 필터 조건을 만족하지 않는 행은 출력에서 제거됩니다.

- 문자열 데이터 유형의 경우 키 값이 지정된 문자열과 일치하는 행을 필터링할 수 있습니다.
- 숫자 데이터 유형의 경우 비교 연산자 <, >, =, !=, <= 및 >=로 키 값을 지정된 값과 비교하여 행을 필터링할 수 있습니다.

여러 필터 조건을 지정하면 기본적으로 AND 연산자를 사용하여 결과가 결합되지만 대신 OR를 선택할 수 있습니다.

Filter 변환은 대/소문자를 구분합니다. 대/소문자를 구분하지 않는 속성 키 이름이 필요한 경우 ApplyMapping 변환을 상위 노드로 추가합니다.

작업 다이어그램에 Filter 변환 노드를 추가하려면

1. (선택 사항) 필요한 경우 리소스 패널을 열고 필터를 선택하여 작업 다이어그램에 새 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. [변환(Transform)] 탭을 선택합니다.

4. [전역 AND(Global AND)] 또는 [전역 OR(Global OR)]를 선택합니다. 이에 따라 여러 필터 조건이 결합되는 방법이 결정됩니다. 모든 조건은 AND 또는 OR 연산을 사용하여 결합됩니다. 필터 조건이 하나만 있는 경우 둘 중 하나를 선택할 수 있습니다.
5. [필터 조건(Filter condition)] 섹션에서 [조건 추가(Add condition)] 버튼을 선택하여 필터 조건을 추가합니다.

   [키(Key)] 필드에서 데이터 집합의 속성 키 이름을 선택합니다. [연산(Operation)] 필드에서 비교 연산자를 선택합니다. [값(Value)] 필드에 비교 값을 입력합니다. 다음은 필터 조건의 몇 가지 예입니다.

   - `year >= 2018`
   - `State matches 'CA*'`

   문자열 값을 필터링할 때 비교 값이 작업 속성(Python 또는 Scala)에서 선택한 스크립트 언어와 일치하는 정규식 포맷을 사용하는지 확인합니다.
6. 필요에 따라 필터 조건을 추가합니다.
7. (선택 사항) 변환 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 수정된 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.
8. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

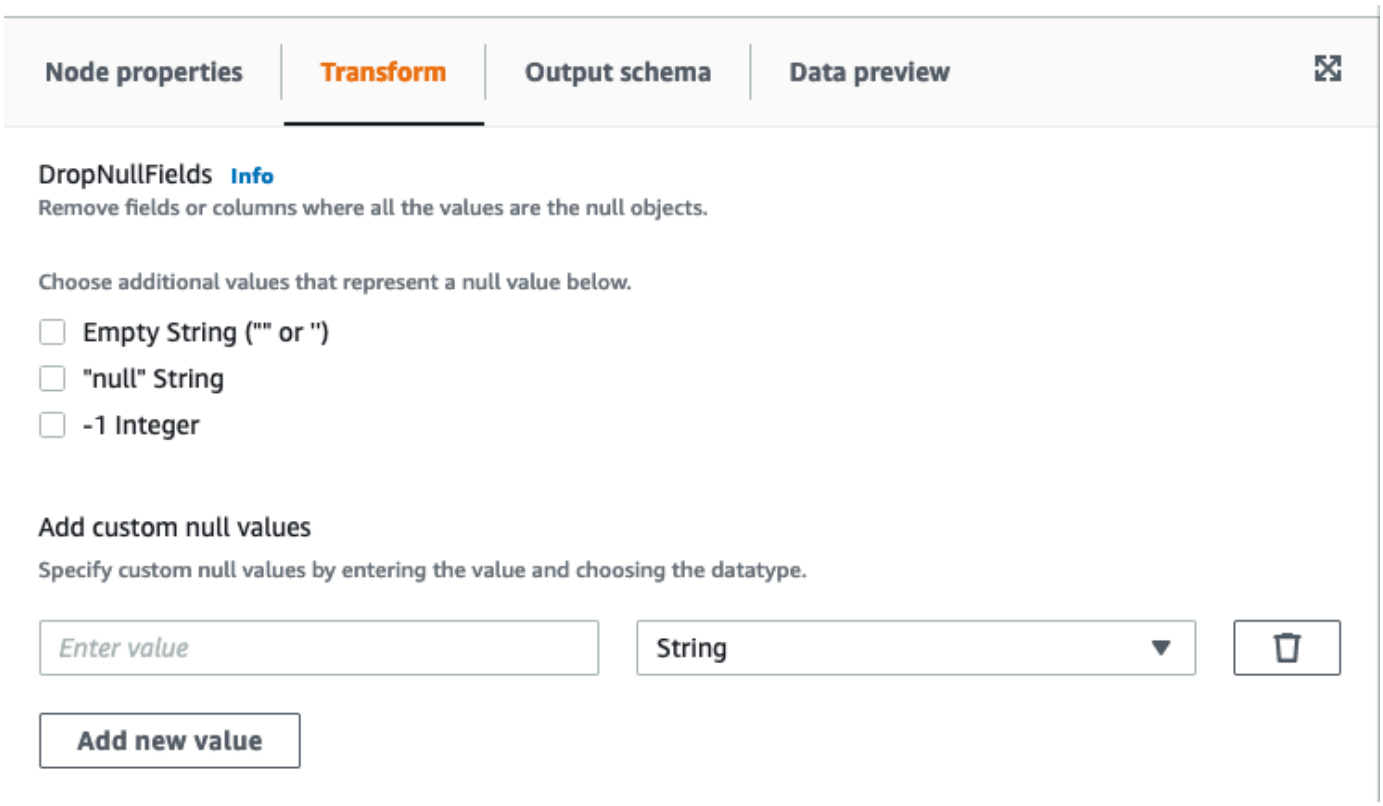
DropNullField를 사용하여 Null 값이 포함된 필드 제거

필드의 모든 값이 'null'인 경우 데이터 집합에서 필드를 제거하려면 DropNullFields 변환을 사용합니다. 기본값으로 AWS Glue Studio는 Null 객체를 인식하지만 빈 문자열, 'null'인 문자열, -1 정수 또는 0과 같은 다른 자리 표시자 등의 일부 값은 자동으로 Null로 인식되지 않습니다.

DropNullFields를 사용하려면

1. 작업 다이어그램에 DropNullFields 노드를 추가합니다.

2. 노드 속성(Node properties) 탭에서 Null 값을 나타내는 추가 값을 선택합니다. 값을 선택하지 않거나 모두 선택할 수 있습니다.



- 빈 문자열("" 또는 '') - 빈 문자열이 포함된 필드가 제거됩니다.
- 'Null 문자열' - 'null'이라는 단어를 포함하는 문자열이 포함된 필드가 제거됩니다.
- -1 정수 - -1(음수 1) 정수가 포함된 필드가 제거됩니다.

3. 필요한 경우 사용자 지정 Null 값을 지정할 수도 있습니다. 이러한 Null 값은 해당 데이터 집합에 고유할 수 있습니다. 사용자 지정 Null 값을 추가하려면 새 값 추가(Add new value)를 선택합니다.
4. 사용자 지정 Null 값을 입력합니다. 예를 들어 0이거나 데이터 집합에서 Null을 나타내는 데 사용되는 임의 값일 수 있습니다.
5. 드롭다운 필드에서 데이터 유형을 선택합니다. 데이터 유형은 문자열 또는 정수일 수 있습니다.

> **Note**
>
> 필드가 Null 값으로 인식되고 필드가 제거되려면 사용자 지정 Null 값과 해당 데이터 유형이 정확히 일치해야 합니다. 사용자 지정 null 값만 일치하고 데이터 형식은 일치하지 않는 부분 일치의 경우 필드가 제거되지 않습니다.

SQL 쿼리를 사용하여 데이터 변환

SQL 변환을 사용하여 SQL 쿼리 형식으로 고유한 변환을 작성할 수 있습니다.

SQL 변환 노드는 여러 데이터 집합을 입력으로 가질 수 있지만 출력으로 단일 데이터 집합만 생성합니다. Apache SparkSQL 쿼리를 입력하는 텍스트 필드가 있습니다. 단순히 SQL 쿼리를 돕기 위해 입력으로 사용되는 각 데이터 집합에 별칭을 할당할 수 있습니다. SQL 구문에 대한 자세한 내용은 Spark SQL 설명서를 참조하세요.

> **Note**
>
> VPC에 위치한 데이터 원본으로 Spark SQL 변환을 사용하는 경우 데이터 원본이 포함된 VPC에 AWS Glue VPC 엔드포인트를 추가합니다. 개발 엔드포인트 구성에 대한 자세한 내용은 AWS Glue Developer Guide의 Adding a Development Endpoint, Setting Up Your Environment for Development Endpoints, and Accessing Your Development Endpoint를 참조하세요.

작업 다이어그램에서 SQL 변환 노드를 사용하려면

1. (선택 사항) 필요한 경우 작업 다이어그램에 변환 노드를 추가합니다. 노드 유형으로 SQL Query를 선택합니다.

   > **Note**
   >
   > 데이터 미리 보기 세션과 사용자 지정 SQL 또는 사용자 지정 코드 노드를 사용하는 경우 데이터 미리 보기 세션은 전체 데이터세트에 대해 SQL 또는 코드 블록을 있는 그대로 실행합니다.

2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않았거나 SQL 변환에 대해 여러 입력을 원하는 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다. 필요에 따라 상위 노드를 추가합니다.
3. 노드 세부 정보 패널에서 [변환(Transform)] 탭을 선택합니다.
4. SQL 쿼리의 소스 데이터 집합은 각 노드의 [이름(Name)] 필드에 지정한 이름으로 식별됩니다. 이러한 이름을 사용하지 않으려는 경우나 이름이 SQL 쿼리에 적합하지 않은 경우 각 데이터 집합에 이름을 연결할 수 있습니다. 콘솔은 `MyDataSource`와 같은 기본 별칭을 제공합니다.

예를 들어, SQL 변환 노드의 상위 노드 이름이 `Rename Org PK field`인 경우 `org_table`이라는 이름을 이 데이터 집합과 연결할 수 있습니다. 그런 다음 이 별칭을 노드 이름 대신 SQL 쿼리에서 사용할 수 있습니다.

5. 머리글 [코드 블록(Code block)] 아래의 텍스트 입력 필드에 SQL 쿼리를 붙여넣거나 입력합니다. 텍스트 필드에는 SQL 구문 강조 표시 및 키워드 제안 사항이 표시됩니다.
6. SQL 변환 노드를 선택한 상태에서 [출력 스키마(Output schema)] 탭을 선택한 다음 [편집(Edit)]을 선택합니다. SQL 쿼리의 출력 필드를 설명하는 열과 데이터 유형을 제공합니다.

   페이지의 [출력 스키마(Output schema)] 섹션에서 다음 작업을 사용하여 스키마를 지정합니다.

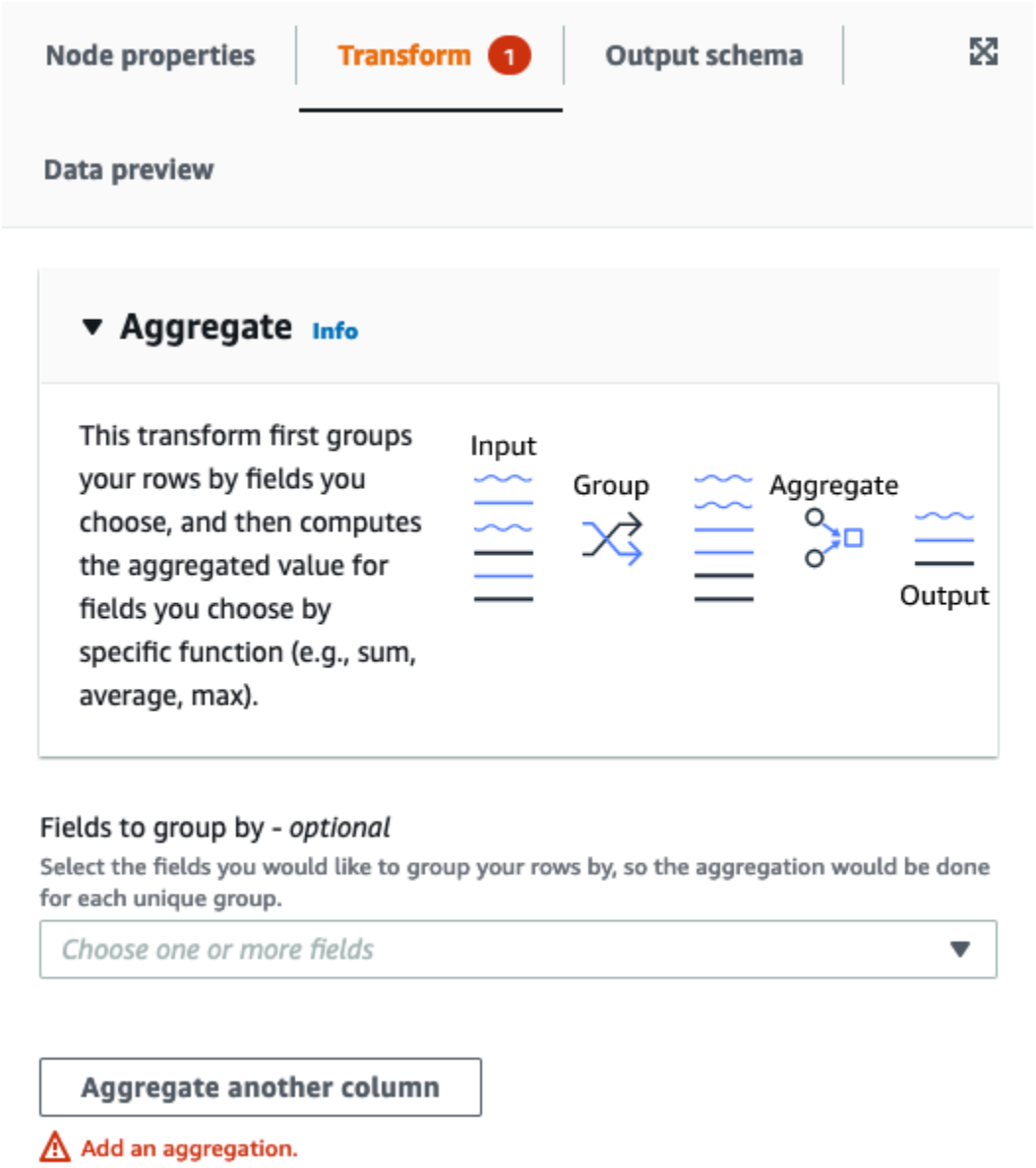
   - 열 이름을 바꾸려면 열의 [키(Key)] 텍스트 상자(필드 또는 속성 키라고도 함)에 커서를 놓고 새 이름을 입력합니다.
   - 열의 데이터 유형을 변경하려면 드롭다운 목록에서 열의 새 데이터 유형을 선택합니다.
   - 스키마에 새 최상위 열을 추가하려면 오버플로 (⋯) 버튼을 선택한 다음 [루트 키 추가(Add root key)]를 선택합니다. 새 열이 스키마 맨 위에 추가됩니다.
   - 스키마에서 열을 제거하려면 키 이름의 맨 오른쪽에 있는 삭제 아이콘 (🗑) 을 선택합니다.
7. 출력 스키마 지정을 마치면 [적용(Apply)]을 선택하여 변경 사항을 저장하고 스키마 편집기를 종료합니다. 변경 사항을 저장하지 않으려면 [취소(Cancel)]를 선택하여 스키마 편집기를 편집합니다.
8. (선택 사항) 노드 속성과 변환 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 수정된 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

집계를 사용하여 선택한 필드에서 요약 계산 수행

집계(Aggregate) 변환을 사용하려면

1. 작업 다이어그램에 집계(Aggregate) 노드를 추가합니다.
2. 노드 속성(Node properties) 탭에서 드롭다운 필드를 선택하여 그룹화할 필드를 선택합니다(선택 사항). 한 번에 둘 이상의 필드를 선택하거나 검색 창에 입력하여 필드 이름을 검색할 수 있습니다.

필드를 선택하면 이름과 데이터 유형이 표시됩니다. 필드를 제거하려면 필드에서 'X'를 선택합니다.



3. 다른 열 집계(Aggregate another column)를 선택합니다. 하나 이상의 필드를 선택해야 합니다.

4. 집계할 필드(Field to aggregate) 드롭다운에서 필드를 선택합니다.
5. 선택한 필드에 적용할 집계 함수를 선택합니다.

   - avg - 평균을 계산합니다.
   - countDistinct - Null이 아닌 고유 값 수를 계산합니다.
   - count - Null이 아닌 값 수를 계산합니다.
   - first - 'group by' 기준을 충족하는 첫 번째 값을 반환합니다.
   - last - 'group by' 기준을 충족하는 마지막 값을 반환합니다.
   - kurtosis - 빈도 분포 곡선의 정점 첨도를 계산합니다.
   - max - 'group by' 기준을 충족하는 최대값을 반환합니다.
   - min - 'group by' 기준을 충족하는 최소값을 반환합니다.
   - skewness - 정규 분포의 확률 분포 비대칭 측정값입니다.
   - stddev_pop - 모집단 표준 편차를 계산하고 모집단 분산의 제곱근을 반환합니다.
   - sum - 그룹에 있는 모든 값의 합계입니다.
   - sumDistinct - 그룹에 있는 고유 값의 합계입니다.
   - var_samp - 그룹의 표본 분산입니다(Null 무시).
   - var_pop - 그룹의 모집단 분산입니다(Null 무시).

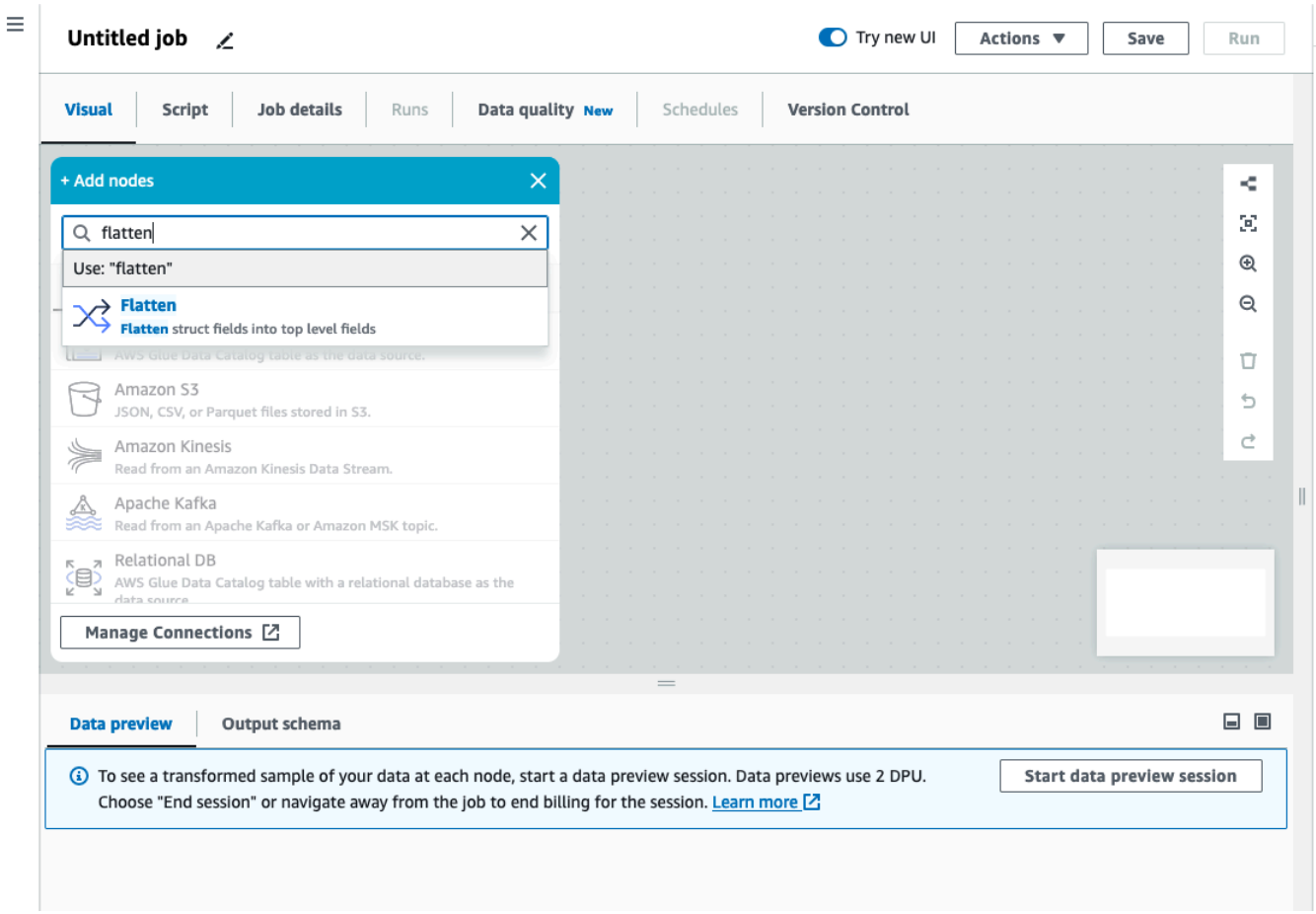
중첩된 구조체 평면화

데이터에 있는 중첩 구조체의 필드를 평면화하여 최상위 필드가 되도록 합니다. 새 필드의 이름은 필드 이름 앞에 해당 필드에 전달될 구조체 필드 이름을 넣고 점으로 구분하여 지정합니다.

예를 들어, 데이터에 이름이 'phone_number'인 Struct 유형의 필드가 있고, 다른 필드에는 'country_code'와 'number'라는 두 개의 필드가 있는 'home_phone'이라는 'Struct' 유형의 필드가 있는 경우를 예로 들어 보겠습니다. 일단 평면화되면 이 두 필드는 각각 'phone_numbers.home_phone.country_code' 및 'phone_numbers.home_phone.number'라는 이름의 최상위 필드가 됩니다.

작업 다이어그램에 Flatten 변환 노드를 추가하려면

1. 리소스 패널을 열고 변환 탭을 선택한 후 평면화를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 검색 표시줄을 사용하여 'Flatten'을 입력하고 평면화 노드를 클릭할 수도 있습니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.



2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. (선택 사항) 변환 탭에서 최대 중첩 수준을 평면화하도록 제한할 수 있습니다. 예를 들어 이 값을 1로 설정하면 최상위 구조체만 평면화됩니다. 최대값을 2로 설정하면 최상위 레벨과 그 바로 아래에 있는 구조체가 평면화됩니다.

UUID 열 추가

UUID (범용 고유 식별) 열을 추가하면 각 행에 고유한 36자 문자열이 할당됩니다.

작업 다이어그램에 UUID 변환 노드를 추가하려면

1. 리소스 패널을 열고 UUID를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.

2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. (선택 사항) 변환 탭에서 새 열의 이름을 사용자 지정할 수 있습니다. 기본적으로 이름은 'uuid'로 지정됩니다.

식별자 열 추가

데이터 세트의 각 행에 숫자 식별자를 할당합니다.

작업 다이어그램에서 식별자 변환 노드를 추가하려면

1. 리소스 패널을 열고 식별자를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. (선택 사항) 변환 탭에서 새 열의 이름을 사용자 지정할 수 있습니다. 기본적으로 이름은 'id'로 지정됩니다.
4. (선택 사항) 작업에서 데이터를 증분 방식으로 처리하고 저장하는 경우 작업 실행 간에 동일한 ID가 재사용되지 않도록 해야 합니다.

   변환 탭에서 고유 확인란 옵션을 선택합니다. 식별자에 작업 타임스탬프가 포함되므로 여러 번 실행할 때 고유하게 표시됩니다. 더 큰 숫자를 입력하도록 허용하려면 long을 입력하는 대신 열을 십진수로 입력합니다.

열을 타임스탬프 형식으로 변환

변환 대상인 타임스탬프로 변환을 사용하여 숫자 또는 문자열 열의 데이터 유형을 타임스탬프로 변경하여 해당 데이터 유형과 함께 저장하거나 타임스탬프가 필요한 다른 변환에 적용할 수 있습니다.

작업 다이어그램에 타임스탬프로 변환 변환 노드를 추가하려면

1. 리소스 패널을 열고 타임스탬프로 변환을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.

2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 변환할 열의 이름을 입력합니다.
4. 변환 탭에서 유형을 선택하여 선택한 열의 구문을 분석하는 방법을 정의합니다.

   값이 숫자인 경우 초(Unix/Python 타임스탬프), 밀리초 또는 마이크로초로 표현할 수 있습니다. 해당 옵션을 선택하세요.

   값이 형식이 지정된 문자열인 경우 'iso' 유형을 선택합니다. 문자열은 ISO 형식의 변형 중 하나를 준수해야 합니다(예: '2022-11-02T 14:40:59 .915Z').

   이 시점에서 유형을 모르거나 행마다 다른 유형을 사용하는 경우 'autodetect'를 선택하면 시스템이 성능 저하를 줄이면서 최선의 추정을 할 수 있습니다.
5. (선택 사항) 변환 탭에서 선택한 열을 변환하는 대신 새 열을 만들고 새 열의 이름을 입력하여 원본을 유지할 수 있습니다.

타임스탬프 열을 형식이 지정된 문자열로 변환

타임스탬프 열을 패턴에 따라 문자열로 형식을 지정합니다. 타임스탬프 형식 지정을 사용하여 날짜 및 시간을 원하는 형식의 문자열로 가져올 수 있습니다. Spark 날짜 구문과 대부분의 Python 날짜 코드를 사용하여 형식을 정의할 수 있습니다.

예를 들어, 날짜 문자열을 '2023-01-01 00:00'과 같은 형식으로 지정하려면 Spark 구문을 사용하여 'YYYYY-MM-DD HH:mm' 같은 형식으로 정의하거나 이에 상응하는 Python 날짜 코드를 '%Y-%m-%d %H: %M' 같은 형식으로 정의하면 됩니다.

작업 다이어그램에서 타임스탬프 형식 지정 변환 노드를 추가하려면

1. 리소스 패널을 열고 타임스탬프 형식 지정을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 변환할 열의 이름을 입력합니다.
4. 변환 탭에서 Spark 날짜 구문 또는 Python 날짜 코드를 사용하여 표현된 사용하려는 타임스탬프 형식 패턴을 입력합니다.

5. (선택 사항) 변환 탭에서 선택한 열을 변환하는 대신 새 열을 만들고 새 열의 이름을 입력하여 원본을 유지할 수 있습니다.

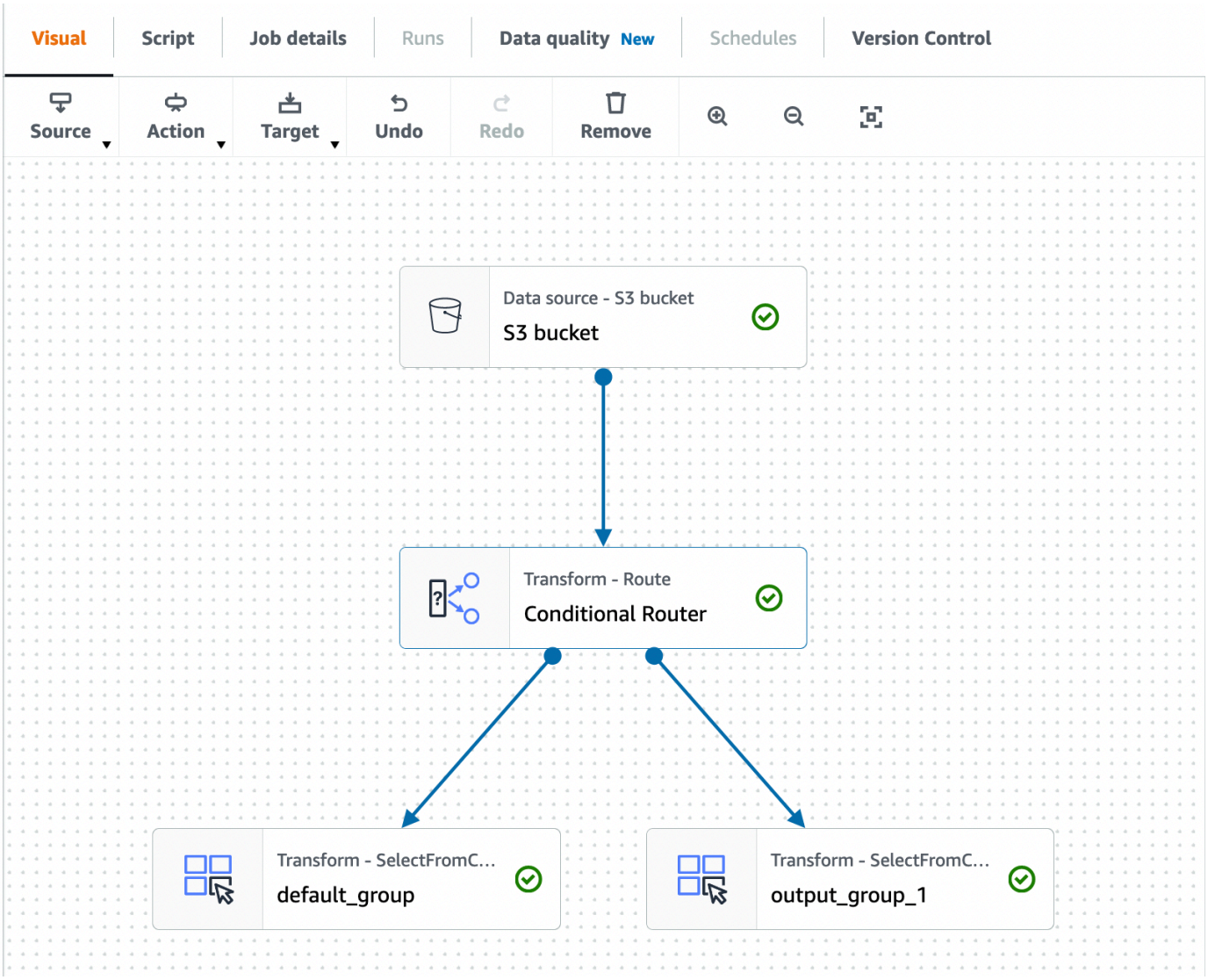
조건부 라우터 변환 생성

조건부 라우터 변환을 사용하면 수신 데이터에 여러 조건을 적용할 수 있습니다. 수신 데이터의 각 행은 그룹 필터 조건을 기준으로 평가되고 해당 그룹으로 처리됩니다. 행이 둘 이상의 그룹 필터 조건을 충족하는 경우 변환은 행을 여러 그룹에 전달합니다. 조건을 충족하지 않는 행은 삭제되거나 기본 출력 그룹으로 라우팅될 수 있습니다.

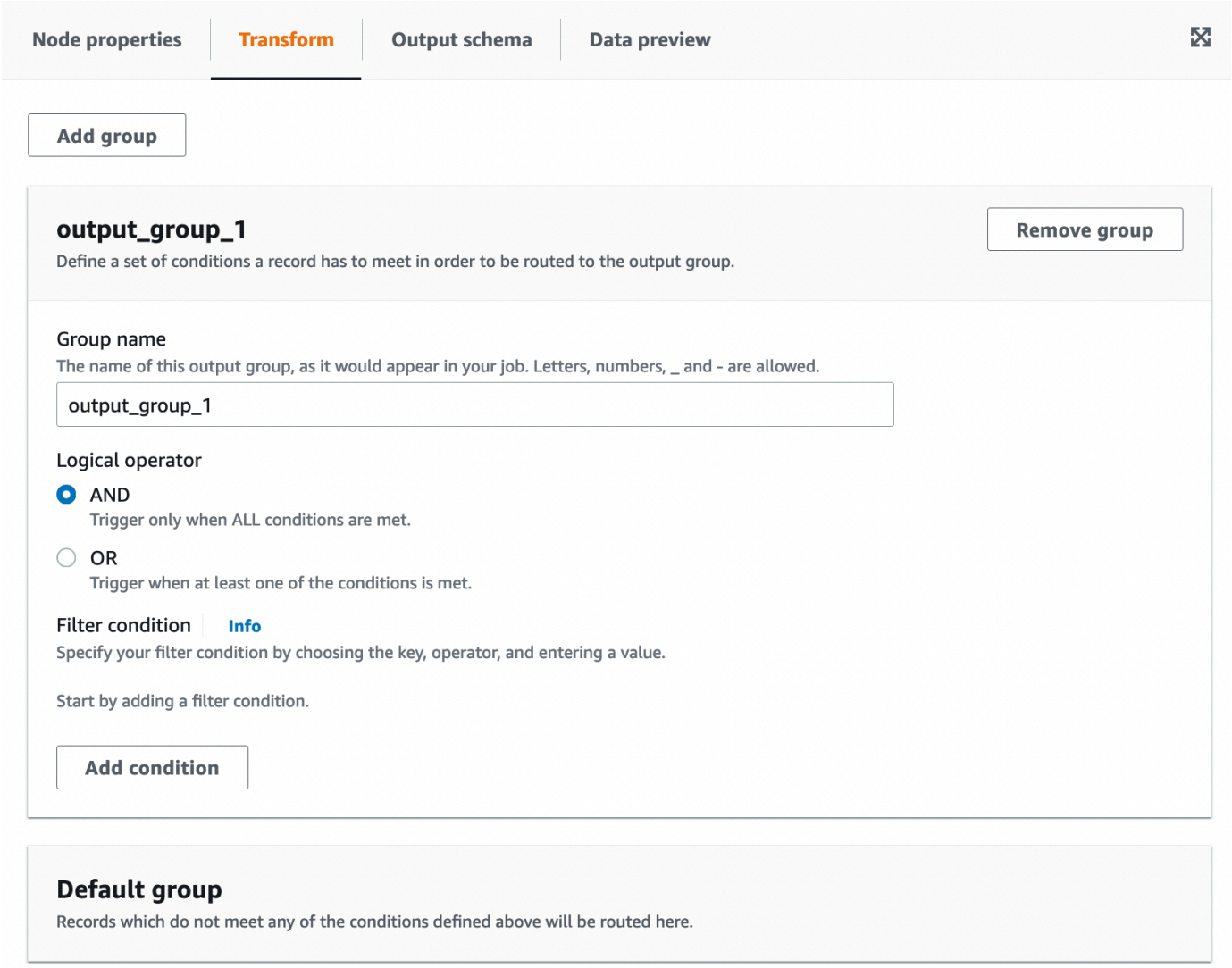
이 변환은 필터 변환과 유사하지만 여러 조건에서 동일한 입력 데이터를 테스트하려는 사용자에게 유용합니다.

조건부 라우터 변환을 추가하려면:

1. 조건부 라우터 변환을 수행할 노드를 선택합니다. 소스 노드일 수도 있고 다른 변환일 수도 있습니다.
2. 작업을 선택한 다음 검색 창을 사용하여 '조건부 라우터'를 찾아 선택합니다. 조건부 라우터 변환이 두 개의 출력 노드와 함께 추가됩니다. 한 출력 노드인 '기본 그룹'에는 다른 출력 노드에 정의된 조건을 전혀 충족하지 않는 레코드가 포함되어 있습니다. 기본 그룹은 편집할 수 없습니다.

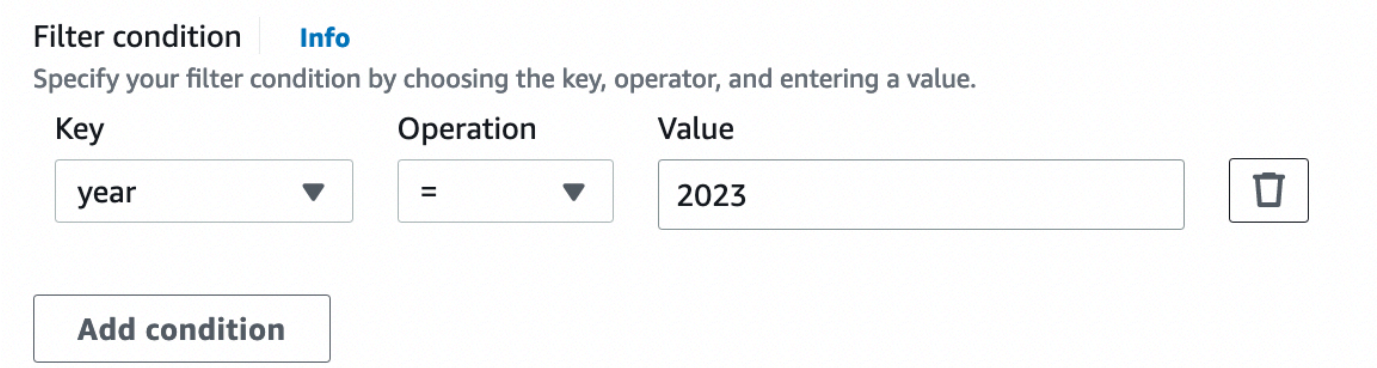


그룹 추가를 선택하여 출력 그룹을 더 추가할 수 있습니다. 각 출력 그룹에 대해 그룹 이름을 지정하고 필터 조건과 논리 연산자를 추가할 수 있습니다.



3. 그룹의 새 이름을 입력하여 출력 그룹 이름을 바꿉니다. AWS Glue Studio는 자동으로 사용자에 대한 그룹 이름을 지정합니다(예: 'output_group_1').
4. 논리 연산자(AND, OR)를 선택하고 키, 연산 및 값을 지정하여 필터 조건을 추가합니다. 논리 연산자를 사용하면 둘 이상의 필터 조건을 구현하고 지정한 각 필터 조건에 대해 논리 연산자를 수행할 수 있습니다.

   키를 지정할 때 스키마에서 사용 가능한 키를 선택할 수 있습니다. 그런 다음 선택한 키 유형에 따라 사용할 수 있는 연산을 선택할 수 있습니다. 예를 들어 키 유형이 '문자열'인 경우 선택할 수 있는 연산은 '일치'입니다.



5. 값 필드에 값을 입력합니다. 조건 추가를 선택하여 추가 필터 조건을 추가할 수 있습니다. 필터 조건을 제거하려면 휴지통 아이콘을 선택합니다.

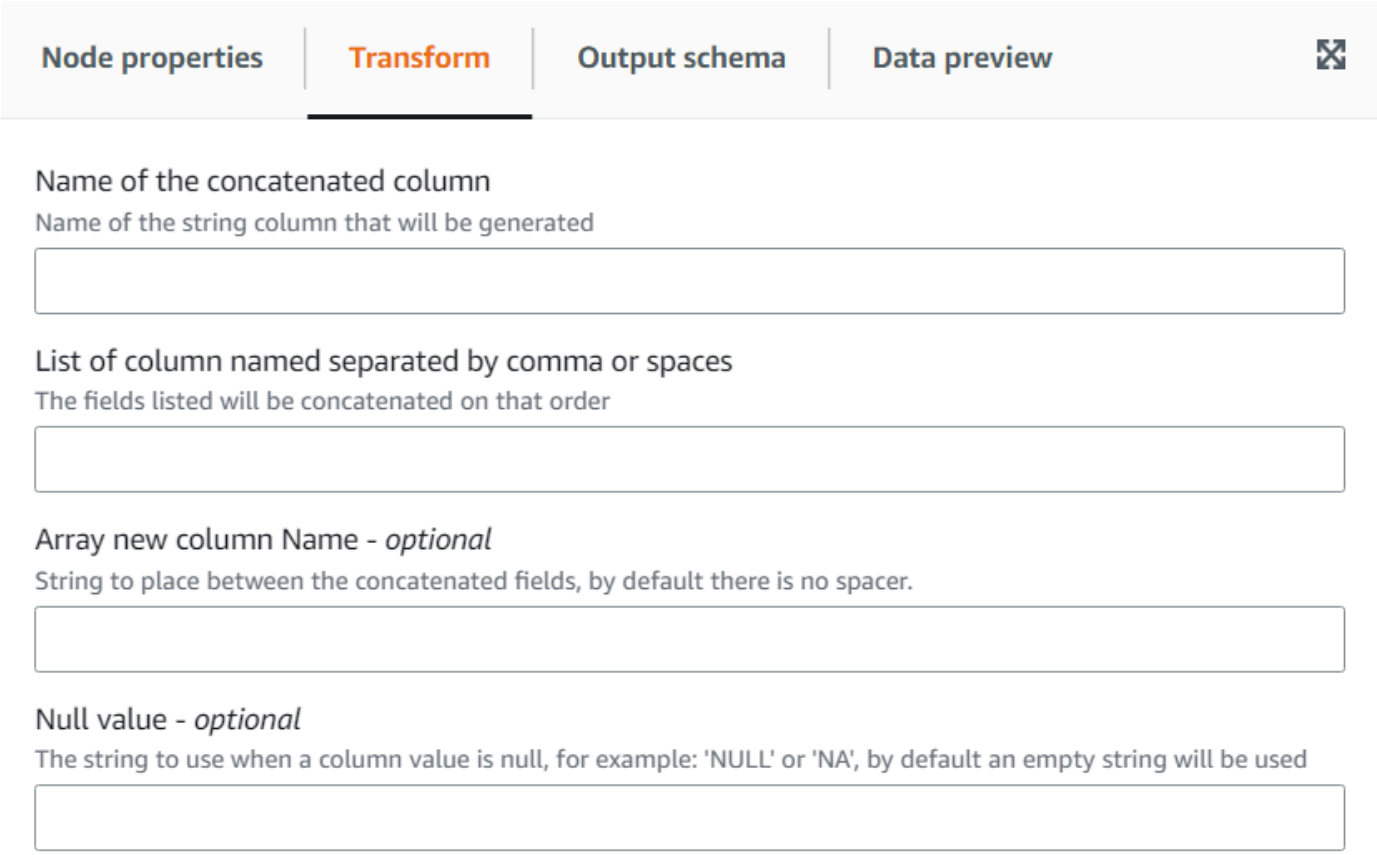
열 연결 변환을 사용하여 열 추가

연결 변환을 사용하면 선택적 스페이서가 있는 다른 열의 값을 사용하여 새 문자열 열을 구축할 수 있습니다. 예를 들어 '-'를 스페이서로 사용해 'year', 'month', 'day'가 순서대로 연결된 항목으로 연결된 열 'date'를 정의하면 다음과 같은 결과가 나옵니다.

day	개월	년	date
01	01	2020	2020-01-01
02	01	2020	2020-01-02
03	01	2020	2020-01-03
04	01	2020	2020-01-04

연결 변환을 추가하려면:

1. 리소스 패널을 엽니다. 그런 다음 열 연결을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 연결된 문자열 및 연결할 열을 보관할 열의 이름을 입력합니다. 드롭다운에서 열을 선택하는 순서대로 사용됩니다.



4. 스페이서 - 선택 사항 - 연결된 필드 사이에 배치할 문자열을 입력합니다. 기본적으로 스페이서는 없습니다.
5. Null 값 - 선택 사항 - 열 값이 null일 때 사용할 문자열을 입력합니다. 기본적으로 열의 값이 'NULL' 또는 'NA'인 경우에는 빈 문자열이 사용됩니다.

문자열 분할 변환을 사용하여 문자열 열 구분

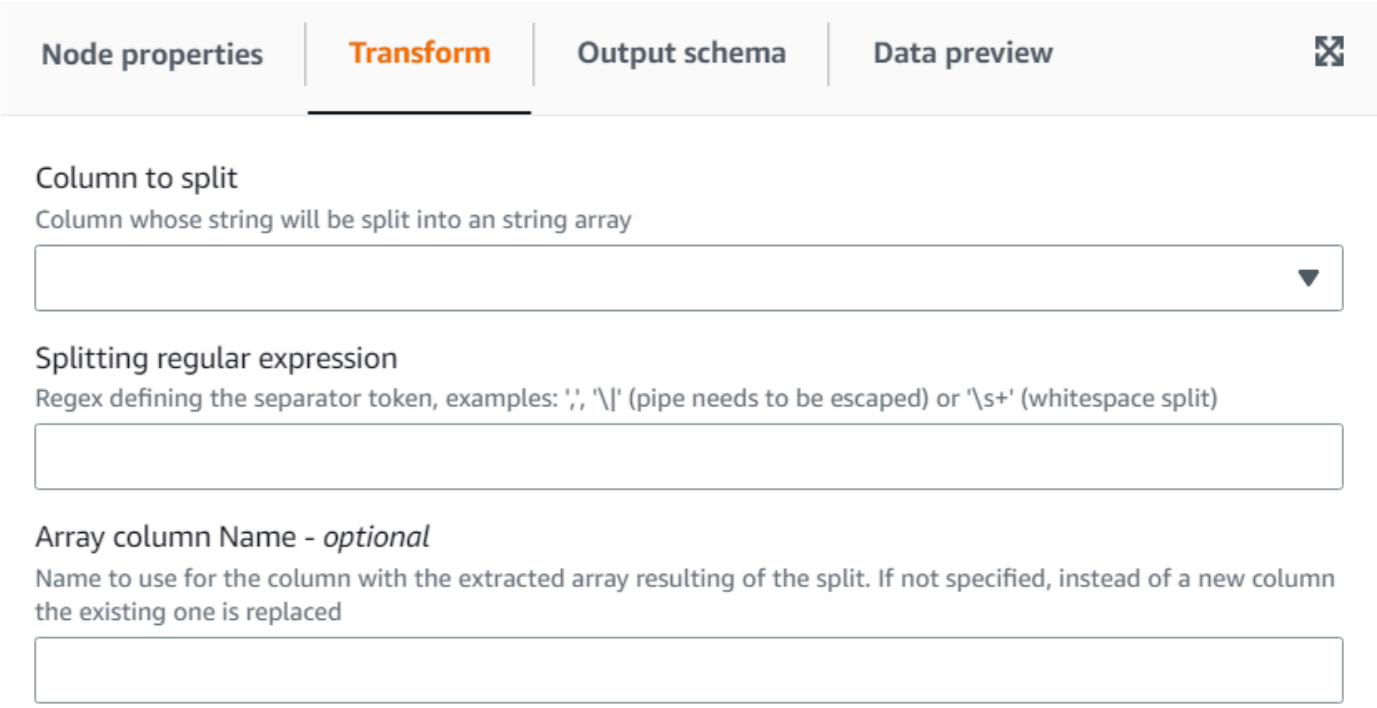
문자열 분할 변환을 사용하면 정규식을 사용하여 문자열을 토큰 배열로 분할해 분할 방식을 정의할 수 있습니다. 그런 다음 각 토큰의 의미를 미리 알고 있다고 가정하고 열을 배열 유형으로 유지하거나 배열에서 열로 변환을 이 다음에 적용하여 배열 값을 상위 수준의 필드로 추출할 수 있습니다. 또한 토큰 순서가 무관한 경우(예: 카테고리 세트) 분해 변환을 사용하여 각 값에 대해 별도의 행을 생성할 수 있습니다.

예를 들어 쉼표를 패턴으로 사용하여 'categories' 열을 분할해 'categorories_arr' 열을 추가할 수 있습니다.

product_id	categories	categories_arr
1	sports,winter	[sports, winter]
2	garden,tools	[garden, tools]
3	videogames	[videogames]
4	game,boardgame,social	[game, boardgame, social]

문자열 분할 변환을 추가하려면:

1. 리소스 패널을 열고 문자열 분할을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 분할할 열을 선택하고 문자열을 분할하는 데 사용할 패턴을 입력합니다. 일반 표현식처럼 특별한 의미가 있어서 이스케이프 처리해야 하는 경우가 아니라면 대부분의 경우 문자만 입력하면 됩니다. 문자 앞에 백슬래시를 추가하여 이스케이프 처리해야 하는 문자는 `\.[]{}()<>*+-=!?^$|`입니다. 예를 들어 점('.')으로 구분하려면 `\.`를 입력해야 합니다. 하지만 쉼표는 특별한 의미가 없으므로 `,` 그대로 지정할 수 있습니다.



4. (선택 사항) 원래 문자열 열을 유지하려는 경우 새 배열 열의 이름을 입력할 수 있습니다. 이렇게 하면 원래 문자열 열과 토큰화된 새 배열 열을 모두 유지할 수 있습니다.

배열을 열로 변환을 사용하여 배열 요소를 상위 수준 열로 추출

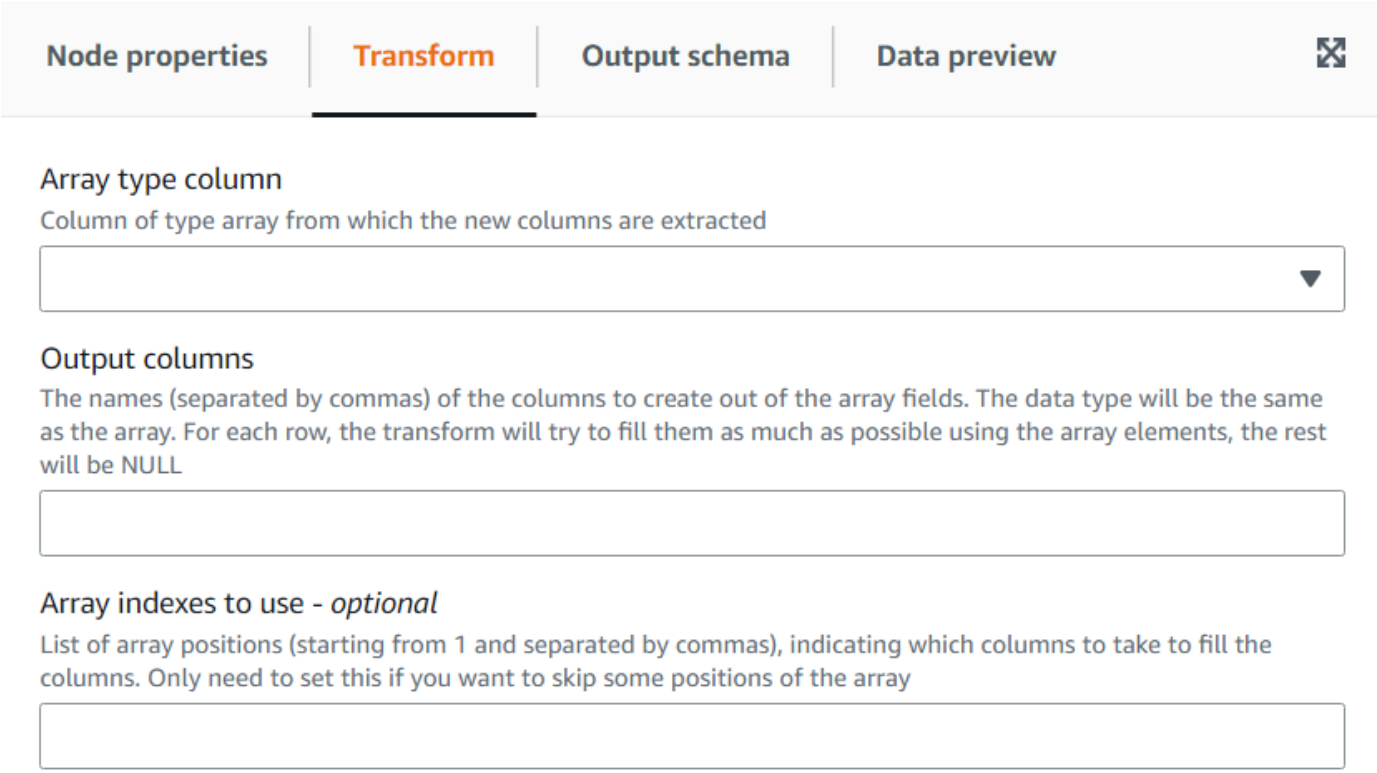
배열을 열로 변환을 사용하면 배열 유형의 열에 있는 일부 또는 모든 요소를 새 열로 추출할 수 있습니다. 배열에 추출하기에 충분한 값이 있는 경우 변환은 새 열을 최대한 많이 채우며, 선택적으로 지정된 위치의 요소를 가져올 수도 있습니다.

예를 들어 ip v4 서브넷에서 '문자열 분할' 변환을 적용한 결과에 해당하는 배열 열 'subnet'이 있는 경우 첫 번째 위치와 네 번째 위치를 새 열 'first_octect' 및 'forth_octect'로 추출할 수 있습니다. 이 예제에서 변환 출력은 다음과 같습니다(마지막 두 행의 배열이 예상보다 짧음).

서브넷	first_octect	fourth_octect
[54, 240, 197, 238]	54	238
[192, 168, 0, 1]	192	1
[192, 168]	192	
[]		

배열을 열로 변환을 추가하려면:

1. 리소스 패널을 열고 배열을 열로를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 추출할 배열 열을 선택하고 추출된 토큰의 새 열 목록을 입력합니다.



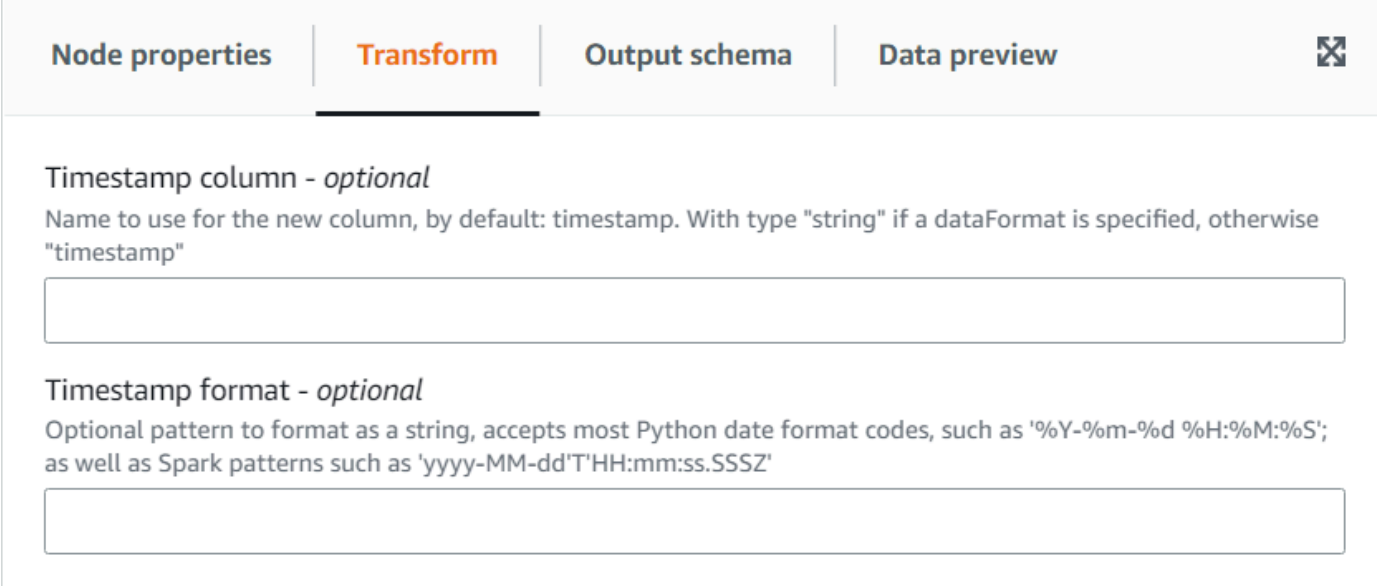
4. (선택 사항) 열에 할당하기 위해 배열 토큰을 사용하지 않으려면 가져올 인덱스를 지정할 수 있습니다. 이 인덱스는 지정된 동일한 순서대로 열 목록에 할당됩니다. 예를 들어 출력 열이 'column1, column2, column3'이고 인덱스가 4, 1, 3'인 경우 배열의 네 번째 요소는 column1로, 첫 번째 요소는 column2로, 세 번째 요소는 column3으로 이동합니다(배열이 인덱스 수보다 적으면 NULL 값이 설정됨).

현재 타임스탬프 추가 변환 사용

현재 타임스탬프 추가 변환을 사용하면 데이터가 처리된 시간으로 행을 표시할 수 있습니다. 이는 감사 목적이나 데이터 파이프라인에서 지연 시간을 추적하는 데 유용합니다. 이 새 열을 타임스탬프 데이터 형식이나 형식이 지정된 문자열로 추가할 수 있습니다.

현재 타임스탬프 추가 변환을 추가하려면:

1. 리소스 패널을 열고 현재 타임스탬프 추가를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.



3. (선택 사항) 변환 탭에서 새 열의 사용자 지정 이름을 입력하고 형식이 지정된 날짜 문자열로 열을 구성하려면 형식을 입력합니다.

행을 열로 피벗 변환 사용

행을 열로 피벗 변환을 사용하면 선택한 열에서 고유 값을 교체하여 숫자 열을 집계할 수 있으며, 이 열은 새 열이 됩니다(여러 개의 열을 선택한 경우 값이 연결되어 새 열의 이름으로 지정됨). 이렇게 하면 행이 통합되지만 각 고유 값에 대한 부분 집계를 통해 열이 더 많아집니다. 예를 들어 다음과 같이 month 및 country별 매출 데이터 세트가 있습니다(쉽게 설명하기 위해 정렬됨).

년	개월	country	amount
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64

년	개월	country	amount
2020	Feb	uk	67
2020	Feb	de	4
2020	Feb	de	7
2020	Feb	us	6
2020	Feb	us	12
2020	Jan	us	90

금액 및 국가를 집계 열로 사용하여 피벗하면 원래 국가 열에서 새 열이 생성됩니다. 아래 표에는 국가 열 대신 de, uk 및 us에 대한 새 열이 있습니다.

년	개월	de	uk	us
2020	Jan	42	32	64
2020	Jan	11	67	18
2021	Jan			90

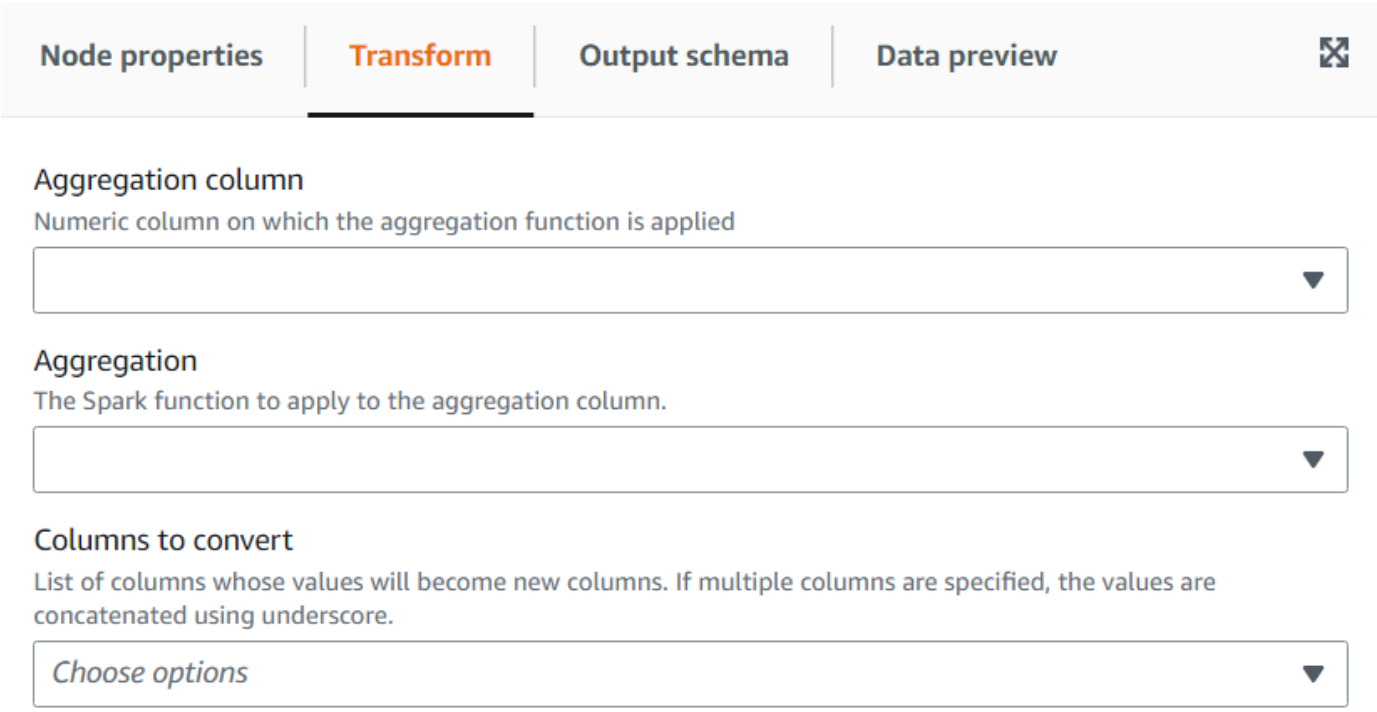
대신 month와 country를 모두 피벗하려는 경우 해당 열 값의 모든 조합에 대한 열이 생성됩니다.

년	Jan_de	Jan_uk	Jan_us	Feb_de	Feb_uk	Feb_us
2020	42	32	64	11	67	18
2021			90			

행을 열로 피벗 변환을 추가하려면:

1. 리소스 패널을 열고 행을 열로 피벗을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.

2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 새 열의 값을 생성하기 위해 집계될 숫자 열, 적용할 집계 함수, 고유한 값을 새 열로 변환할 열을 선택합니다.



열을 행으로 피벗 취소 변환 사용

피벗 취소 변환을 사용하면 열을 새 열의 값으로 변환하여 각 고유 값에 대한 행을 생성할 수 있습니다. 피벗과 반대이지만 원래 열로 결합을 분할하거나 집계된 동일한 값의 행을 구분할 수 없다는 점에서 차이가 있습니다(나중에 분할 변환을 사용하여 분할 가능). 예를 들어 다음과 같은 테이블이 있습니다.

년	개월	de	uk	us
2020	Jan	42	32	64
2020	Feb	11	67	18
2021	Jan			90

'amount' 값을 사용해 'de', 'uk', 'us' 열을 'country' 열로 피벗 취소하고 다음과 같은 결과를 얻을 수 있습니다(설명을 위해 여기에서는 정렬됨).

년	개월	country	amount
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Feb	uk	67
2020	Feb	de	11
2020	Feb	us	18
2021	Jan	us	90

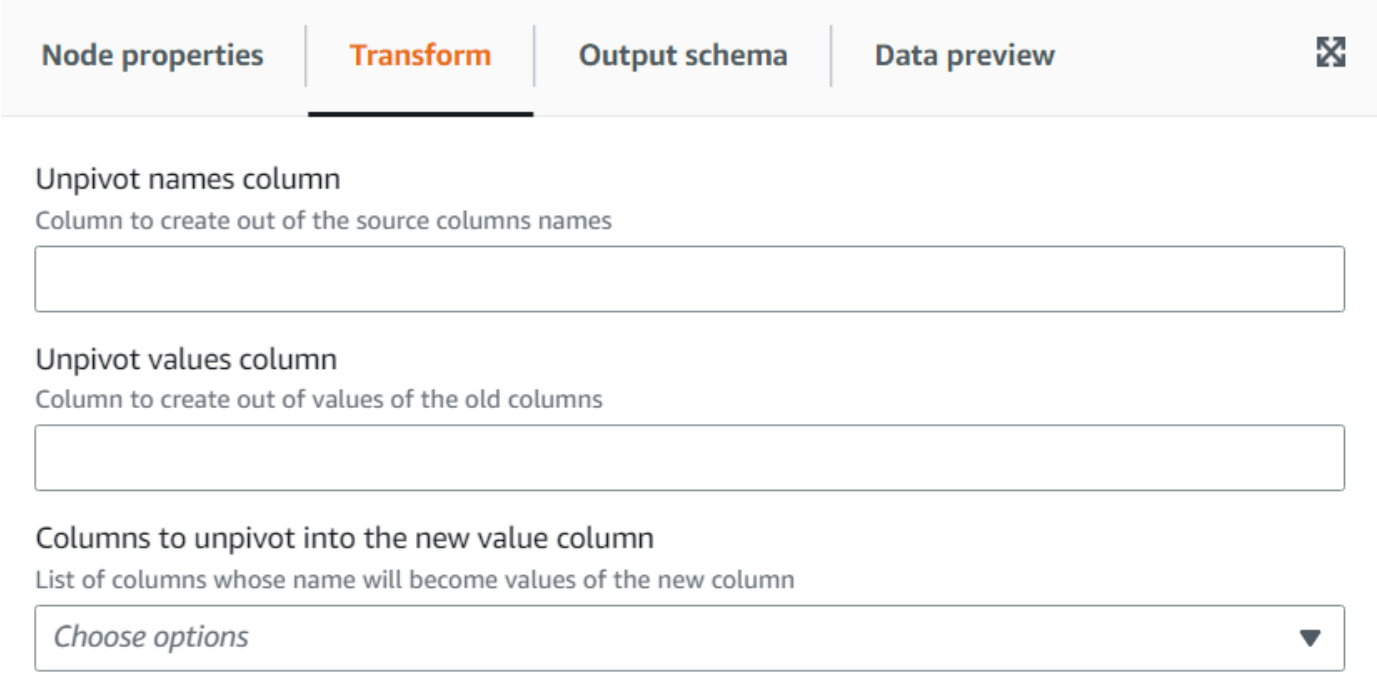
값이 NULL인 열(2021년 1월의 'de' 및 'uk')은 기본적으로 생성되지 않습니다. 이 옵션을 활성화하여 다음을 얻을 수 있습니다.

년	개월	country	amount
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Feb	uk	67
2020	Feb	de	11
2020	Feb	us	18
2021	Jan	us	90
2021	Jan	de	

년	개월	country	amount
2021	Jan	uk	

열을 행으로 피벗 취소 변환을 추가하려면:

1. 리소스 패널을 열고 열을 행으로 피벗 취소를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 피벗 취소할 열의 이름과 값을 보관하기 위해 생성할 새 열을 입력합니다.

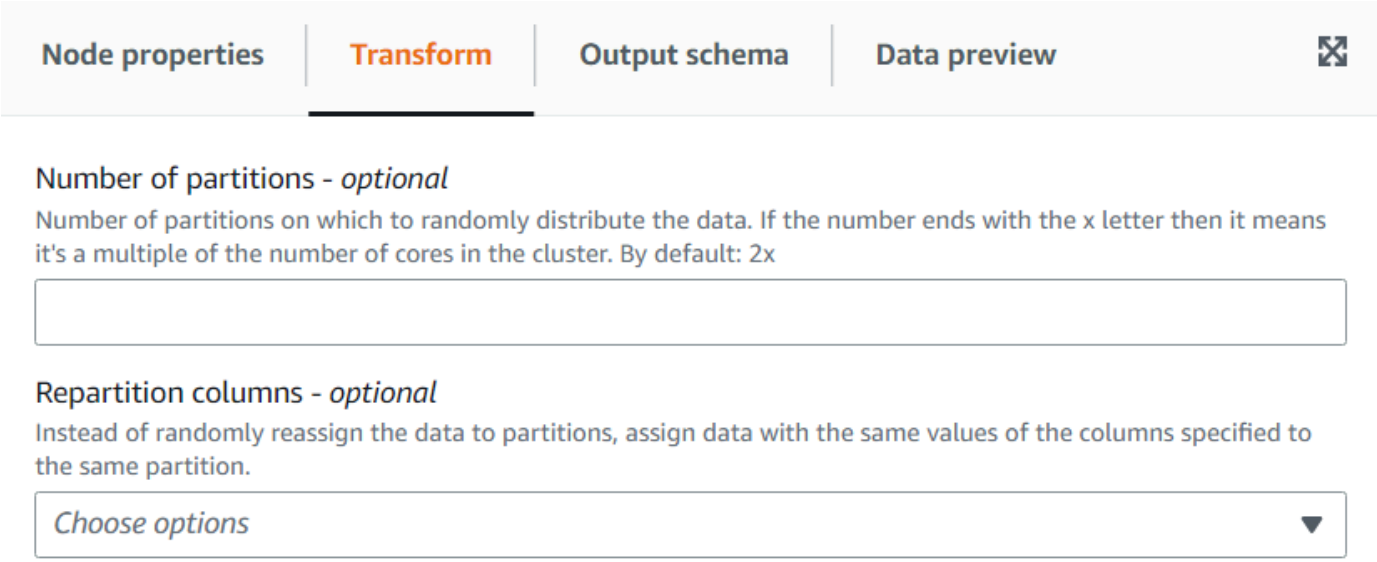


처리 균형 자동 조절 변환을 사용하여 런타임 최적화

처리 균형 자동 조절 변환은 더 나은 성능을 위해 작업자 사이에서 데이터를 재배포합니다. 이는 데이터가 불균형하거나 소스에서 가져온 데이터로 인해 충분한 병렬 처리가 불가능한 경우에 유용합니다. 소스가 gzip으로 압축되었거나 JDBC인 경우에 이러한 상황이 흔히 나타납니다. 데이터 재배포에도 어느 정도의 성능 비용이 발생하므로, 데이터의 균형이 이미 조절된 상태인 경우 최적화해도 보상 효과를 얻지 못할 수도 있습니다. 이 변환은 Apache Spark 재파티셔닝을 사용하여 클러스터 용량에 최적화된 여러 파티션 사이에서 데이터를 임의로 재할당합니다. 고급 사용자의 경우 여러 파티션을 수동으로 입

력할 수 있습니다. 또한 지정된 열을 기준으로 데이터를 재구성하여 파티셔닝된 테이블의 쓰기를 최적화하는 데 사용할 수 있습니다. 그러면 출력 파일의 통합 기능이 더욱 강화됩니다.

1. 리소스 패널을 열고 처리 균형 자동 조절을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. (선택 사항) 변환 탭에서 파티션 수를 입력할 수 있습니다. 일반적으로 이 값은 시스템에서 결정하도록 하는 것이 좋지만, 이 값을 제어해야 하는 경우 승수를 조정하거나 특정 값을 입력할 수 있습니다. 열을 기준으로 파티셔닝된 데이터를 저장하려는 경우 리파티션 열과 동일한 열을 선택할 수 있습니다. 이렇게 하면 각 파티션에서 파일 수를 최소화하고 파티션당 파일 수가 많아지지 않도록 방지할 수 있습니다. 파일 수가 많아지면 해당 데이터를 쿼리하는 도구의 성능이 저하될 수 있습니다.



파생 열 변환을 사용하여 다른 열 결합

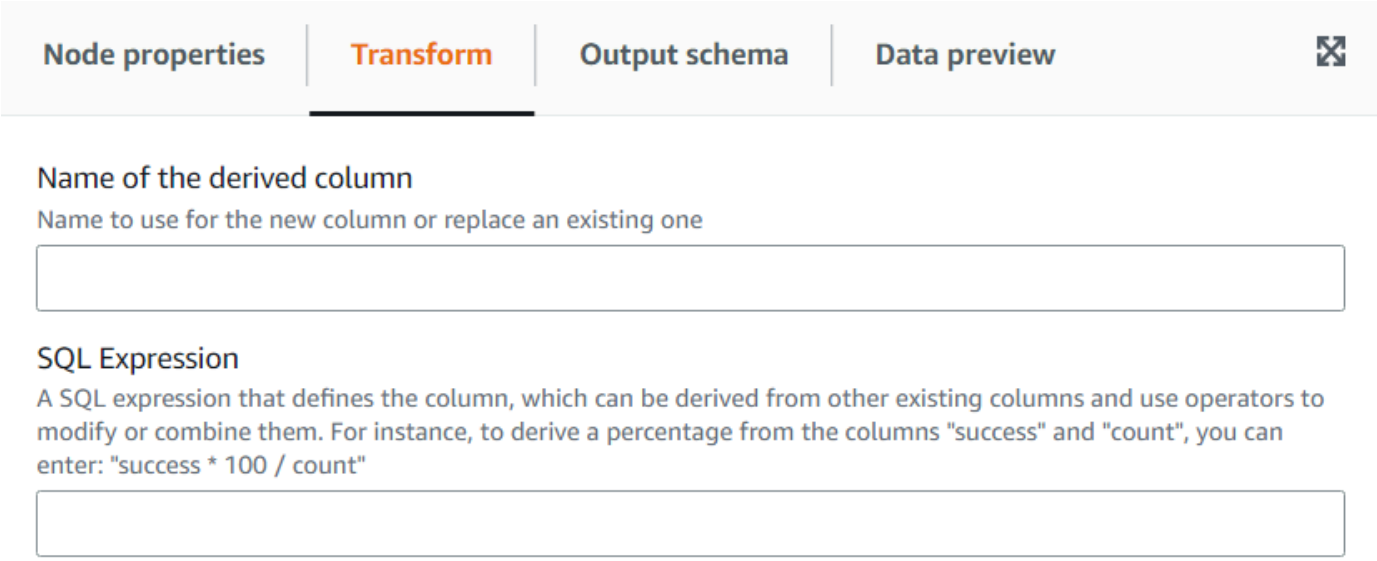
파생 열 변환을 사용하면 상수 및 리터럴뿐만 아니라 데이터의 다른 열을 사용할 수 있는 수학 공식 또는 SQL 표현식을 기반으로 새 열을 정의할 수 있습니다. 예를 들어 'success' 및 'count' 열에서 'percentage' 열을 파생하려면 SQL 표현식 'success * 100 / count || '%"를 입력하면 됩니다.

결과 예제:

success	count	percentage
14	100	14%
6	20	3%
3	40	7.5%

파생 열 변환을 추가하려면:

1. 리소스 패널을 열고 파생 열을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 열 이름 및 해당 콘텐츠에 대한 표현식을 입력합니다.



조회 변환을 사용하여 카탈로그 테이블에서 일치하는 데이터 추가

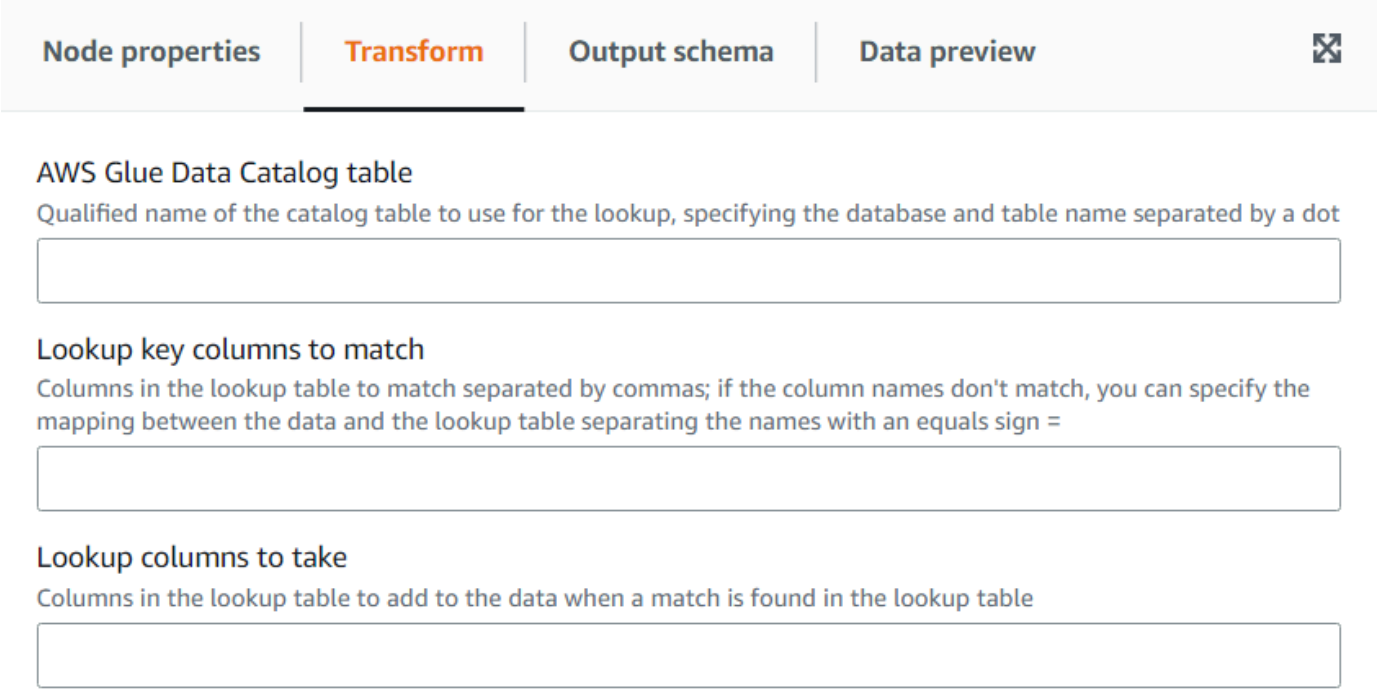
조회 변환을 사용하면 키가 데이터에 정의된 조회 열과 일치하는 경우 정의된 카탈로그 테이블의 열을 추가할 수 있습니다. 이는 조건 일치 열을 사용하여 데이터와 조회 테이블 사이에서 왼쪽 외부 조인을 수행하는 방법과 같습니다.

조회 변환을 추가하려면:

1. 리소스 패널을 열고 조회를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 조회를 수행하는 데 사용할 완전히 정규화된 카탈로그 테이블 이름을 입력합니다. 예를 들어 데이터베이스가 'mydb'이고 테이블이 'mytable'인 경우 'mydb.mytable'을 입력합니다. 그런 다음 조회 키가 구성된 경우 조회 테이블에서 일치 항목을 찾을 기준을 입력합니다. 쉼표로 구분된 키 열 목록을 입력합니다. 키 열 중 하나 이상의 이름에서 같은 이름이 없는 경우 일치 매핑을 정의해야 합니다.

   예를 들어 데이터 열이 'user_id' 및 'region'이고 사용자 테이블에서 해당 열의 이름이 'id' 및 'region'인 경우 일치시킬 열 필드에 'user_id=id, region'을 입력합니다. region=region을 입력할 수도 있지만 동일하므로 이 작업은 필요하지 않습니다.
4. 마지막으로, 조회 테이블에서 일치하는 행에서 가져올 열을 입력하여 데이터에 통합합니다. 일치하는 항목이 없으면 해당 열은 NULL로 설정됩니다.

> **Note**
>
> 조회 변환 아래에서 효율성을 높이기 위해 왼쪽 조인을 사용합니다. 조회 테이블에 복합 키가 있는 경우 하나만 일치하도록 일치하는 열을 모든 키 열과 매칭하도록 설정합니다. 그렇지 않으면 여러 조회 행이 매칭되어 각 일치 항목에 대해 추가 행이 추가됩니다.



배열 또는 맵을 행으로 분해 변환 사용

분해 변환을 사용하면 중첩된 구조에서 조작하기 쉬운 개별 행으로 값을 추출할 수 있습니다. 배열의 경우 이 변환은 행의 다른 열에 대한 값을 복제하여 배열의 각 값에 대해 행을 생성합니다. 맵의 경우 이 변환은 열 및 행의 다른 열을 키와 값으로 사용해 각 항목에 대한 행을 생성합니다.

예를 들어 다음 데이터 세트에는 값이 여러 개인 'category' 배열 열이 있습니다.

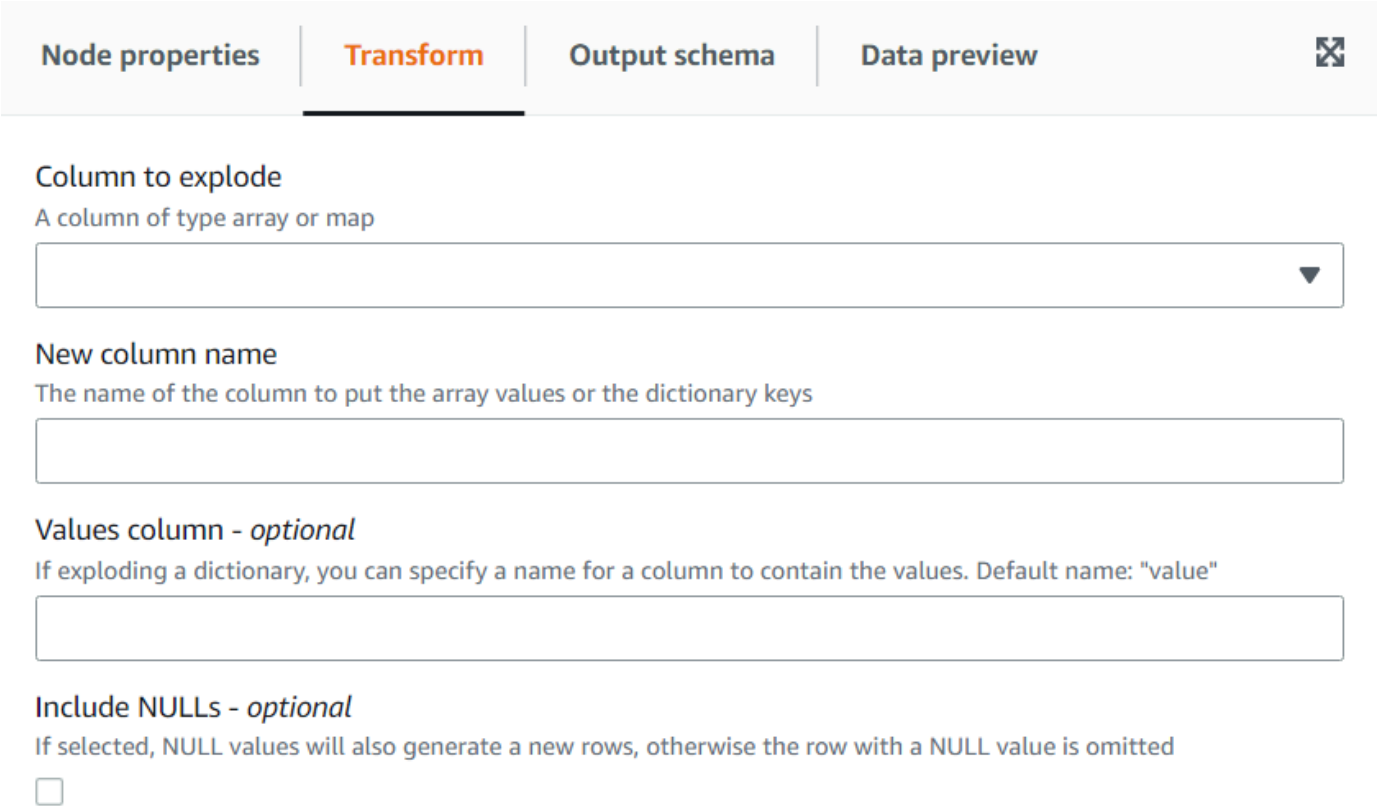
product_id	category
1	[sports, winter]
2	[garden, tools]
3	[videogames]
4	[game, boardgame, social]
5	[]

'category' 열을 같은 이름의 열로 분해하면 해당 열이 재정의됩니다. 다음 결과를 얻기 위해 NULL을 포함하도록 선택할 수 있습니다(설명을 위해 정렬됨).

product_id	category
1	sports
1	winter
2	garden
2	tool
3	videogames
4	게임
4	boardgame
4	social
5	

배열 또는 맵을 행으로 분해 변환을 추가하려면:

1. 리소스 패널을 열고 분해 또는 맵을 행으로 분해를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. (선택 사항) 노드 속성 탭에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 분해할 열을 선택합니다(배열 또는 맵 유형이어야 함). 그런 다음 배열 항목에 대한 열 이름 또는 맵을 분해하는 경우 키와 값에 대한 열 이름을 입력합니다.
4. (선택 사항) 변환 탭에서 분해할 열이 NULL이거나 구조가 비어 있는 경우 기본적으로 분해된 데이터 세트에서 해당 열은 생략됩니다. 행을 유지하고 싶으면(새 열을 NULL로 표시) 'NULL 포함'을 선택합니다.

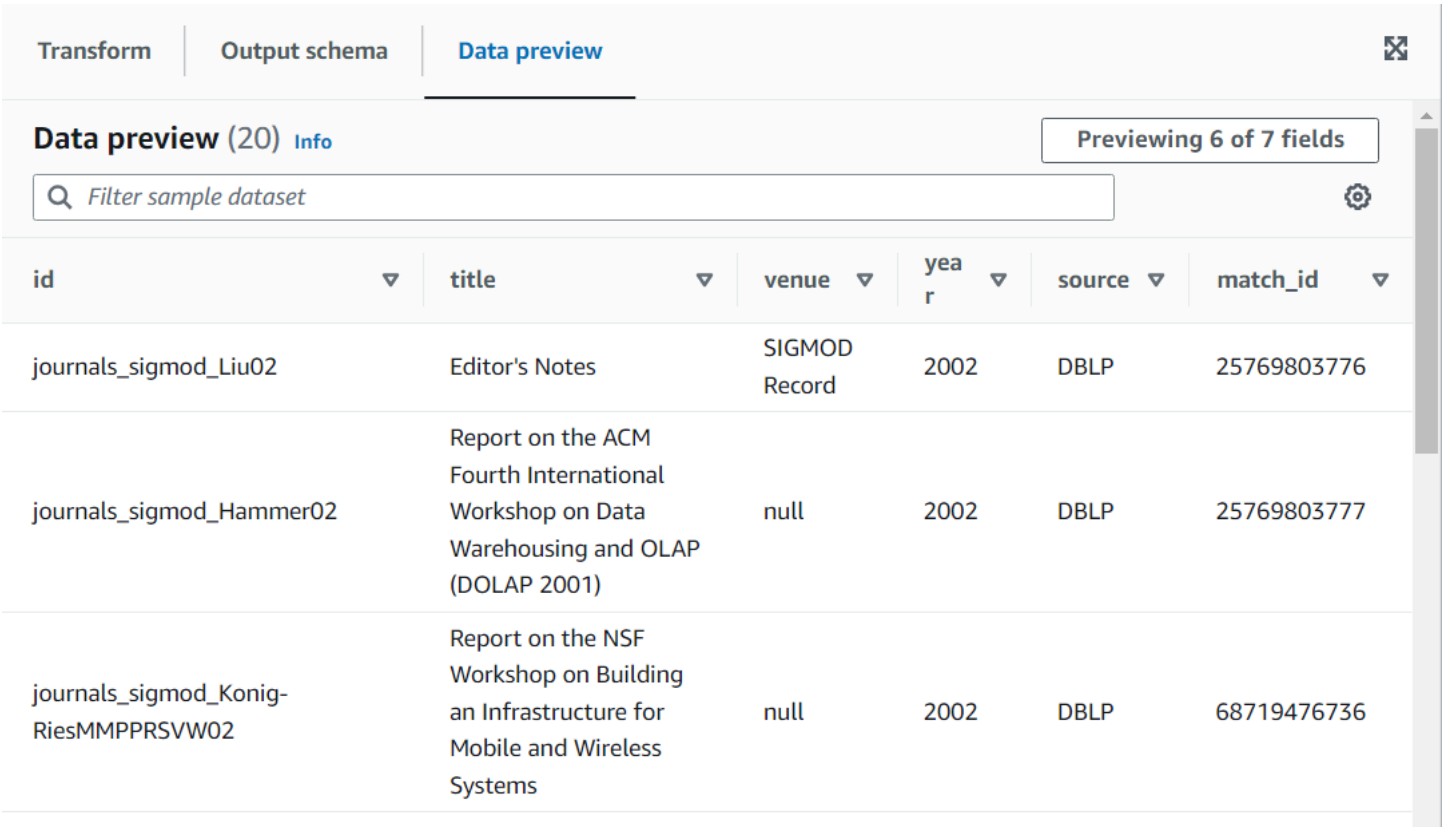


레코드 일치 변환을 사용하여 기존 데이터 분류 변환을 간접적으로 호출

이 변환은 기존 레코드 일치 기계 학습 데이터 분류 변환을 간접적으로 호출합니다.

변환은 레이블을 기반으로 훈련된 모델을 기준으로 현재 데이터를 평가합니다. 알고리즘 훈련에 따라 동등한 것으로 간주되는 항목 그룹에 각 행을 할당하기 위해 'match_id' 열이 추가됩니다. 자세한 내용은 Lake Formation FindMatches를 사용하는 레코드 일치를 참조하세요.

> **Note**
>
> 시각적 작업에서 사용하는 AWS Glue의 버전은 레코드 일치 변환을 생성하기 위해 AWS Glue에서 사용하는 버전과 일치해야 합니다.



작업 다이어그램에 레코드 일치 변환 노드를 추가하려면

1. 리소스 패널을 열고 정규식 일치를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. 노드 속성 패널에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 기계 학습 변환 페이지로부터 가져온 ID를 입력합니다.

4. (선택 사항) 변환 탭에서 신뢰도 점수를 추가하는 옵션을 확인할 수 있습니다. 추가 컴퓨팅 용량을 소모하여 모델은 각 일치에 대한 신뢰도 점수를 추가 열로 추정합니다.

null 행 제거

이 변환은 데이터 세트에서 모든 열이 null인 행을 제거합니다. 또한 빈 필드를 포함하도록 이 기준을 확장하여 하나 이상의 열이 비어 있지 않은 행을 유지할 수 있습니다.

작업 다이어그램에 null 행 제거 변환 노드를 추가하려면

1. 리소스 패널을 열고 null 행 제거를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. 노드 속성 패널에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. (선택 사항) 행이 null이 아닐 뿐만 아니라 비어 있지 않도록 하려면 변환 탭에서 확장 옵션을 선택합니다. 이렇게 하면 이 변환의 목적을 위해 빈 문자열, 배열 또는 맵이 null로 간주됩니다.

JSON 데이터를 포함하는 문자열 열 구문 분석

이 변환은 JSON 데이터를 포함하는 문자열 열을 구문 분석하고 JSON이 객체인지 또는 배열인지에 따라 각각 해당 문자열 열을 구문 또는 배열 열로 변환합니다. 선택적으로 구문 분석된 열과 원래 열을 모두 유지할 수 있습니다.

선택적 샘플링을 통해 JSON 스키마를 제공하거나 추론할 수 있습니다(JSON 객체의 경우).

작업 다이어그램에 JSON 열 구문 분석 변환 노드를 추가하려면

1. 리소스 패널을 열고 JSON 열 구문 분석을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. 노드 속성 패널에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 JSON 문자열이 포함된 열을 선택합니다.
4. (선택 사항) 변환 탭에서 SQL 구문을 사용하여 JSON 데이터가 따르는 스키마를 입력합니다(예: 객체의 경우 'field1 STRING, field2 INT', 배열의 경우 'ARRAY<STRING>').

   배열의 경우 스키마가 필요하지만 객체의 경우 스키마가 지정되지 않았다면 데이터를 사용하여 추론됩니다. 스키마 추론에 따른 영향(특히 대규모 데이터 세트의 경우)을 줄이려면 스키마를 추론하는 데 사용할 샘플 비율을 입력하여 전체 데이터를 두 번 읽지 않도록 합니다. 값이 1보다 작으면

해당 비율의 무작위 샘플이 스키마를 추론하는 데 사용됩니다. 데이터를 신뢰할 수 있고 객체가 행 사이에서 일관된 경우 0.1과 같은 작은 비율을 사용하면 성능을 개선할 수 있습니다.

5. (선택 사항) 원래 문자열 열과 구문 분석된 열을 모두 유지하려는 경우 변환 탭에 새 열 이름을 입력할 수 있습니다.

JSON 경로 추출

이 변환은 JSON 문자열 열에서 새 열을 추출합니다. 이 변환은 몇 개의 데이터 요소만 필요하고 전체 JSON 콘텐츠를 테이블 스키마로 가져오지 않으려는 경우에 유용합니다.

작업 다이어그램에 JSON 경로 추출 변환 노드를 추가하려면

1. 리소스 패널을 열고 JSON 경로 추출을 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.
2. 노드 속성 패널에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 JSON 문자열이 포함된 열을 선택합니다. 쉼표로 구분된 하나 이상의 JSON 경로 식을 입력합니다. 각 표현식은 JSON 배열 또는 객체에서 값을 추출하는 방법을 참조합니다. 예를 들어 JSON 열에 속성이 'prop_1' 및 'prop2'인 객체가 포함된 경우 이름을 'prop_1, prop_2'로 지정하여 모두 추출할 수 있습니다.

   예를 들어 JSON 필드에 JSON `{"a. a": 1}`에서 속성을 추출하기 위한 특수 문자가 있는 경우 `$['a. a']` 경로를 사용할 수 있습니다. 단, 쉼표는 경로를 구분하는 기호로 예약되었으므로, 예외입니다. 그런 다음 각 경로에 대해 해당 열 이름을 쉼표로 구분하여 입력합니다.
4. (선택 사항) 추출 후에 변환 탭에서 JSON 열을 삭제하도록 선택할 수 있습니다. 필요한 부분을 추출한 후 나머지 JSON 데이터가 필요하지 않은 경우에 유용합니다.

정규식을 사용하여 문자열 조각 추출

이 변환은 정규식을 사용하여 문자열 조각을 추출하고 문자열 조각에서 새 열을 생성하거나 정규식 그룹을 사용하는 경우 여러 열을 생성할 수 있습니다.

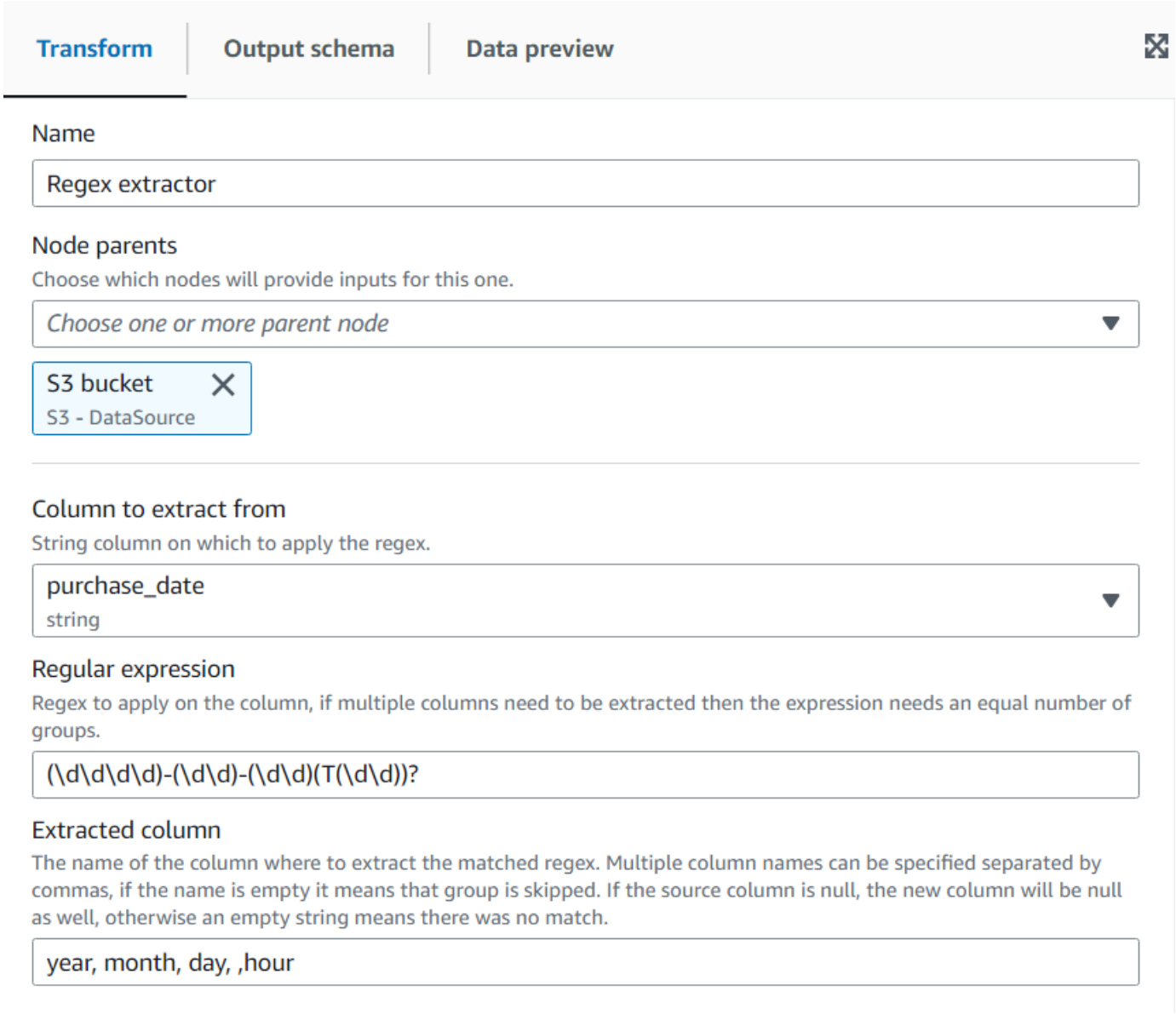
작업 다이어그램에 정규식 추출기 변환 노드를 추가하려면

1. 리소스 패널을 열고 정규식 추출기를 선택하여 작업 다이어그램에 새 변환을 추가합니다. 노드를 추가할 때 선택한 노드가 상위 노드가 됩니다.

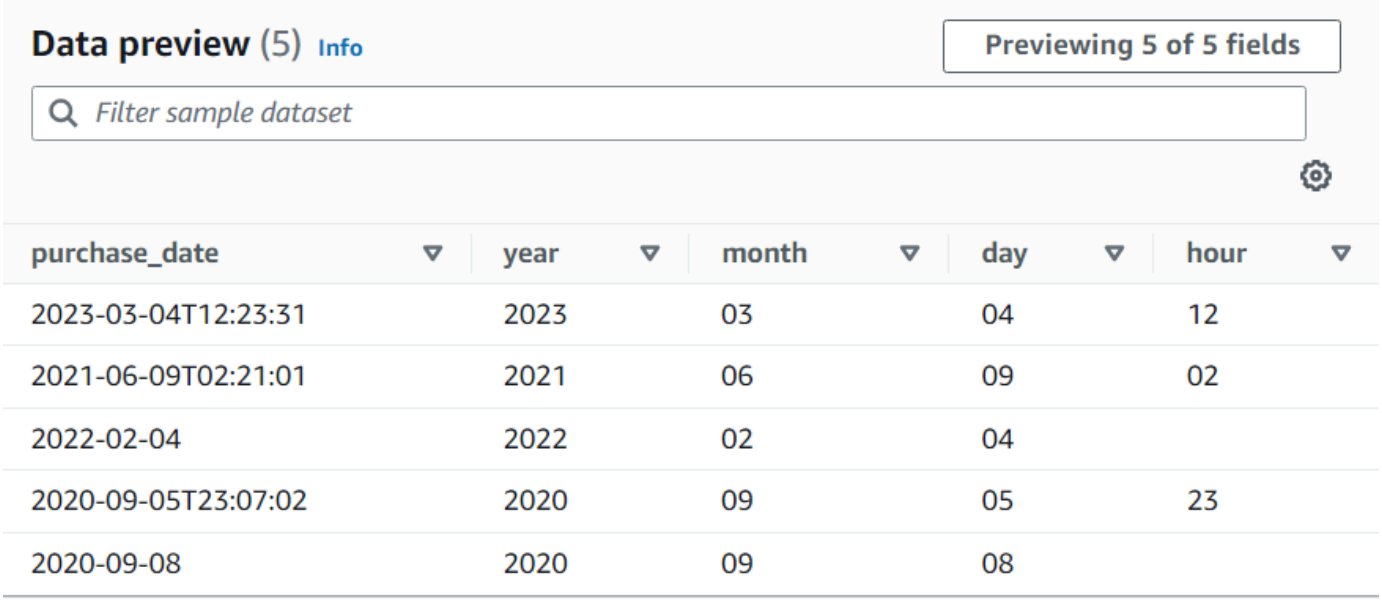
2. 노드 속성 패널에서 작업 다이어그램에 노드 이름을 입력할 수 있습니다. 노드 상위 항목이 아직 선택되지 않은 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.
3. 변환 탭에서 정규식과 정규식을 적용해야 하는 열을 입력합니다. 그런 다음 일치하는 문자열을 저장할 새 열의 이름을 입력합니다. 소스 열이 null인 경우에만 새 열이 null이 되고, 정규식이 일치하지 않으면 열이 비어 있게 됩니다.

   정규식이 그룹을 사용하는 경우 쉼표로 구분된 해당 열 이름이 있지만 열 이름을 비워 두면 그룹을 건너뛸 수 있습니다.

   예를 들어 ISO의 긴 날짜 형식과 ISO의 짧은 날짜 형식을 모두 사용하는 문자열이 포함된 'purchase_date' 열이 있는 경우 가능하면 연도, 월, 일, 시간을 추출하려고 합니다. 시간 그룹은 선택 사항이지만, 시간 그룹을 사용할 수 없는 행에서는 정규식이 일치하지 않으므로 추출된 모든 그룹이 빈 문자열이 됩니다. 이 경우 그룹에서 시간을 선택 사항으로 지정하지 않고 내부 항목을 사용하려고 하므로 이름을 비우고 추출되지 않도록 합니다(이 그룹에는 T 문자가 포함됨).


Transform
Output schema
Data preview
Name
Regex extractor
Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node
S3 bucket
S3 - DataSource
Column to extract from
String column on which to apply the regex.
purchase_date
string
Regular expression
Regex to apply on the column, if multiple columns need to be extracted then the expression needs an equal number of groups.
(\d\d\d\d)-(\d\d)-(\d\d)(T(\d\d))?
Extracted column
The name of the column where to extract the matched regex. Multiple column names can be specified separated by commas, if the name is empty it means that group is skipped. If the source column is null, the new column will be null as well, otherwise an empty string means there was no match.
year, month, day, ,hour

데이터 미리 보기의 결과:



purchase_date	year	month	day	hour
2023-03-04T12:23:31	2023	03	04	12
2021-06-09T02:21:01	2021	06	09	02
2022-02-04	2022	02	04	
2020-09-05T23:07:02	2020	09	05	23
2020-09-08	2020	09	08	

사용자 정의 변환 생성

데이터에 대해 더 복잡한 변환을 수행해야 하거나 데이터 속성 키를 데이터 집합에 추가하려는 경우 작업 다이어그램에 [사용자 정의 코드(Custom code)] 변환을 추가할 수 있습니다. 사용자 정의 코드 노드를 사용하면 변환을 수행하는 스크립트를 입력할 수 있습니다.

사용자 정의 코드를 사용하는 경우 스키마 편집기를 사용하여 사용자 정의 코드를 통해 출력에 대한 변경 사항을 표시해야 합니다. 스키마를 편집할 때 다음 작업을 수행할 수 있습니다.

- 데이터 속성 키 추가 또는 제거
- 데이터 속성 키의 데이터 유형 변경
- 데이터 속성 키의 이름 변경
- 중첩 속성 키 재구성

출력을 대상 위치로 보내기 전에 사용자 정의 변환 노드의 결과에서 단일 `DynamicFrame`을 선택하려면 SelectFromCollection 변환을 사용해야 합니다.

다음 태스크를 사용하여 작업 다이어그램에 사용자 정의 변환 노드를 추가합니다.

작업 다이어그램에 사용자 정의 코드 변환 노드 추가

작업 다이어그램에 사용자 정의 변환 노드를 추가하려면

1. (선택 사항) 리소스 패널을 열고 사용자 지정 변환을 선택하여 작업 다이어그램에 사용자 지정 변환을 추가합니다.
2. [노드 속성(Node properties)] 탭에서 작업 다이어그램에 노드 이름을 입력합니다. 노드 상위 항목이 아직 선택되지 않았거나 사용자 정의 변환에 대해 여러 입력을 원하는 경우 [노드 상위 항목(Node parents)] 목록에서 변환의 입력 소스로 사용할 노드를 선택합니다.

사용자 정의 변환 노드에 대한 코드 입력

입력 필드에 코드를 입력하거나 복사할 수 있습니다. 작업에서는 이 코드를 사용하여 데이터 변환을 수행합니다. Python 또는 Scala에서 코드 조각을 제공할 수 있습니다. 코드는 입력으로 하나 이상의 `DynamicFrames`를 취해야 하며 `DynamicFrames` 컬렉션을 반환합니다.

사용자 정의 변환 노드에 대한 스크립트를 입력하려면

1. 작업 다이어그램에서 사용자 정의 변환 노드를 선택한 상태에서 [변환(Transform)] 탭을 선택합니다.
2. 머리글 [코드 블록(Code block)] 아래의 텍스트 입력 필드에 변환 코드를 붙여넣거나 입력합니다. 사용하는 코드는 [작업 세부 정보(Job details)] 탭에서 작업에 대해 지정된 언어와 일치해야 합니다.

   코드에서 입력 노드를 참조할 때 AWS Glue Studio는 작업 다이어그램 노드에서 반환된 `DynamicFrames`의 이름을 생성 순서에 따라 순차적으로 지정합니다. 코드에 다음 이름 지정 방법 중 하나를 선택합니다.

   - 클래식 코드 생성 - 함수 이름을 사용하여 작업 다이어그램에서 노드를 참조합니다.
     - 데이터 원본 노드: `DataSource0`, `DataSource1`, `DataSource2` 등.
     - 변환 노드: `Transform0`, `Transform1`, `Transform2` 등.
   - 새 코드 생성 - 노드의 노드 속성(Node properties) 탭에서 지정된 이름에 '_node1', '_node2' 등을 추가하여 사용합니다. 예: `S3bucket_node1`, `ApplyMapping_node2`, `S3bucket_node2`, `MyCustomNodeName_node1`.

   새 코드 생성기에 대한 자세한 내용은 스크립트 코드 생성 섹션을 참조하세요.

다음 예는 코드 상자에 입력할 코드의 포맷을 보여줍니다.

Python

다음 예제에서는 처음 수신한 DynamicFrame을 가져와 DataFrame으로 변환하여 기본 필터 방식을 적용한 다음(1,000개 이상의 투표가 있는 레코드만 유지) 반환하기 전에 다시 DynamicFrame으로 변환합니다.

```
def FilterHighVoteCounts (glueContext, dfc) -> DynamicFrameCollection:
    df = dfc.select(list(dfc.keys())[0]).toDF()
    df_filtered = df.filter(df["vote_count"] > 1000)
    dyf_filtered = DynamicFrame.fromDF(df_filtered, glueContext, "filter_votes")
    return(DynamicFrameCollection({"CustomTransform0": dyf_filtered}, glueContext))
```

Scala

다음 예제에서는 처음 수신한 DynamicFrame을 가져와 DataFrame으로 변환하여 기본 필터 방식을 적용한 다음(1,000개 이상의 투표가 있는 레코드만 유지) 반환하기 전에 다시 DynamicFrame으로 변환합니다.

```
object FilterHighVoteCounts {
  def execute(glueContext : GlueContext, input : Seq[DynamicFrame]) :
 Seq[DynamicFrame] = {
    val frame = input(0).toDF()
    val filtered = DynamicFrame(frame.filter(frame("vote_count") > 1000),
 glueContext)
    Seq(filtered)
  }
}
```

사용자 정의 변환 노드에서 스키마 편집

사용자 지정 변환 노드를 사용하는 경우 AWS Glue Studio는 변환을 통해 생성되는 출력 스키마를 자동으로 유추할 수 없습니다. 스키마 편집기를 사용하여 사용자 정의 변환 코드에 의해 구현된 스키마 변경 사항을 설명합니다.

사용자 정의 코드 노드에는 사용자 정의 코드에 대한 입력으로 DynamicFrame을 제공하는 상위 노드가 여러 개 있을 수 있습니다. 사용자 정의 코드 노드는 DynamicFrames의 컬렉션을 반환합니다. 입력으로 사용되는 각 DynamicFrame에는 연결된 스키마가 있습니다. 사용자 정의 코드 노드에서 반환된 각 DynamicFrame을 설명하는 스키마를 추가해야 합니다.

> **Note**
>
> 사용자 지정 변환에서 사용자 고유의 스키마를 설정하면 AWS Glue Studio에서는 이전 노드의 스키마를 상속하지 않습니다. 스키마를 업데이트하려면 사용자 지정(Custom transform) 변환 노드를 선택한 다음 데이터 미리 보기(Data preview) 탭을 선택합니다. 미리 보기가 생성되면 '미리 보기 스키마 사용(Use Preview Schema)'을 선택합니다. 그러면 스키마가 미리 보기 데이터를 사용하는 스키마로 대체됩니다.

사용자 정의 변환 노드에 대한 출력 스키마를 편집하려면

1. 작업 다이어그램에서 사용자 정의 변환 노드를 선택한 상태에서 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택합니다.
2. [편집(Edit)]을 선택하여 스키마를 변경합니다.

   배열 또는 객체와 같은 중첩 데이터 속성 키가 있는 경우 각 스키마 패널의 오른쪽 상단에 있는 [행 확장(Expand-Rows)] 아이콘 ( ) 을 선택하여 하위 데이터 속성 키 목록을 확장할 수 있습니다. 이 아이콘을 선택하면 하위 속성 키 목록을 축소하도록 선택할 수 있는 [행 축소(Collapse-Rows)] 아이콘 ( ) 으로 변경됩니다.
3. 페이지 오른쪽 섹션에서 다음 작업을 사용하여 스키마를 수정합니다.

   - 속성 키의 이름을 바꾸려면 속성 키의 [키(Key)] 텍스트 상자에 커서를 놓고 새 이름을 입력합니다.
   - 속성 키의 데이터 유형을 변경하려면 목록을 사용하여 속성 키의 새 데이터 유형을 선택합니다.
   - 스키마에 새 최상위 속성 키를 추가하려면 [취소(Cancel)] 버튼 왼쪽에 있는 [오버플로(Overflow)] ( ) 아이콘을 선택한 다음 [루트 키 추가(Add root key)]를 선택합니다.
   - 스키마에 하위 속성 키를 추가하려면 상위 키와 연결된 [키 추가(Add-Key)] 아이콘 을 선택합니다. 하위 키의 이름을 입력하고 데이터 유형을 선택합니다.

- 스키마에서 속성 키를 제거하려면 키 이름의 맨 오른쪽에 있는 [제거(Remove)] 아이콘
  ( )
  을 선택합니다.

4. 사용자 정의 변환 코드가 여러 DynamicFrames를 사용하는 경우 출력 스키마를 더 추가할 수 있습니다.

   - 비어 있는 새 스키마를 추가하려면 [오버플로(Overflow)]
     ( )
     아이콘을 선택한 다음 [출력 스키마 추가(Add output schema)]를 선택합니다.
   - 기존 스키마를 새 출력 스키마로 복사하려면 복사하려는 스키마
     가 스키마 선택기에 표시되는지 확인합니다. [오버플로(Overflow)]
     ( )
     아이콘을 선택한 다음 [복제(Duplicate)]를 선택합니다.

   출력 스키마를 제거하려면 복사하려는 스키마가 스키마 선택기에 표시되는지 확인합니다. [오버플로(Overflow)]
   ( )
   아이콘을 선택한 다음 [삭제(Delete)]를 선택합니다.

5. 새 스키마에 새 루트 키를 추가하거나 복제된 키를 편집합니다.
6. 출력 스키마를 수정할 때 [적용(Apply)] 버튼을 선택하여 변경 사항을 저장하고 스키마 편집기를 종료합니다.

   변경 사항을 저장하지 않으려면 [취소(Cancel)] 버튼을 선택합니다.

사용자 정의 변환 출력 구성

사용자 정의 코드 변환은 결과 집합에 DynamicFrame이 하나만 있더라도 DynamicFrames의 컬렉션을 반환합니다.

사용자 정의 변환 노드에서 출력을 처리하려면

1. 사용자 정의 변환 노드가 상위 노드로 있는 SelectFromCollection 변환 노드를 추가합니다. 이 변환을 업데이트하여 사용하려는 데이터 집합을 나타냅니다. 자세한 정보는 SelectFromCollection을 사용하여 유지할 데이터 집합 선택을 참조하세요.
2. 사용자 정의 변환 노드에서 생성된 추가 DynamicFrames를 사용하려면 작업 다이어그램에 SelectFromCollection 변환을 더 추가합니다.

항공 데이터 집합을 여러 데이터 집합으로 분할하기 위해 사용자 정의 변환 노드를 추가하지만 비행 날짜 또는 항공편 번호와 같은 각 출력 스키마에서 일부 식별 속성 키를 복제하는 시나리오를 고려하세요. 사용자 정의 변환 노드를 상위 항목로 사용하여 각 출력 스키마에 대해 SelectFromCollection 변환 노드를 추가합니다.

3. (선택 사항) 그런 다음 각 SelectFromCollection 변환 노드를 작업의 다른 노드에 대한 입력으로 사용하거나 데이터 대상 노드의 부모로 사용할 수 있습니다.

사용자 지정 시각적 변환으로 데이터 변환

사용자 지정 시각적 변환을 사용하면 변환을 생성하여 AWS Glue Studio 작업에 사용할 수 있습니다. 사용자 지정 시각적 변환을 사용하면 코딩에 익숙하지 않은 ETL 개발자도 AWS Glue Studio 인터페이스를 사용하여 점점 늘어나는 변환 라이브러리를 검색하고 사용할 수 있습니다.

사용자 지정 시각적 변환을 생성한 다음 Amazon S3에 업로드하여 AWS Glue Studio의 시각적 편집기를 통해 이러한 작업을 수행하는 데 사용할 수 있습니다.

주제

- 사용자 지정 시각적 변환 시작하기
- 단계 1. JSON 구성 파일 생성
- 단계 2. 변환 로직 구현
- 단계 3. AWS Glue Studio에서 사용자 지정 시각적 변환을 검증하고 문제를 해결합니다.
- 4단계. 필요에 따라 사용자 지정 시각적 변환 업데이트
- 5단계. AWS Glue Studio에서 사용자 지정 시각적 변환 사용
- 사용 예제:
- 사용자 지정 시각적 스크립트의 예
- 비디오

사용자 지정 시각적 변환 시작하기

사용자 지정 시각적 변환을 생성하려면 다음 단계를 따라야 합니다.

- 단계 1. JSON 구성 파일 생성
- 단계 2. 변환 로직 구현
- 단계 3. 사용자 지정 시각적 변환 검증

- 4단계. 필요에 따라 사용자 지정 시각적 변환 업데이트
- 5단계. AWS Glue Studio에서 사용자 지정 시각적 변환 사용

Amazon S3 버킷을 설정하여 시작하고 1단계로 진행합니다. JSON 구성 파일을 생성합니다.

사전 조건

고객이 제공한 변환은 고객 AWS 계정 내에 있습니다. 해당 계정은 변환을 소유하므로 변환 내용을 보거나(검색 및 사용) 편집하거나 삭제할 수 있는 모든 권한을 가집니다.

AWS Glue Studio에서 사용자 지정 변환을 사용하려면 다음 두 파일을 생성하여 해당 AWS 계정의 Amazon S3 자산 버킷에 업로드해야 합니다.

- Python 파일 - 변환 함수를 포함합니다.
- JSON 파일 - 변환을 설명합니다. 이 파일은 변환을 정의하는 데 필요한 구성 파일이라고도 합니다.

파일을 함께 연결하려면 두 파일에 동일한 이름을 사용하세요. 예:

- myTransform.json
- myTransform.py

선택적으로 아이콘이 포함된 SVG 파일을 제공하여 사용자 지정 시각적 변환에 사용자 지정 아이콘을 제공할 수 있습니다. 파일을 함께 페어링하려면 아이콘에서 동일한 이름을 사용합니다.

- myTransform.svg

AWS Glue Studio에서는 각 파일 이름을 사용하여 자동으로 일치시킵니다. 기존 모듈의 파일 이름은 같을 수 없습니다.

변환 파일 이름에 대한 권장 규칙

AWS Glue Studio에서는 파일을 작업 스크립트에 모듈(예: `import myTransform`)로 가져옵니다. 따라서 파일 이름은 Python 변수 이름(식별자)에 설정된 것과 동일한 이름 지정 규칙을 따라야 합니다. 특히 문자 또는 밑줄로 시작하고 그 뒤에는 문자, 숫자 및/또는 밑줄로만 구성되어야 합니다.

> **Note**
>
> 예기치 못한 런타임 문제를 방지하려면 변환 파일 이름이 기존에 로드된 Python 모듈(예: `sys`, `array`, `copy` 등)과 충돌하지 않는지 확인하세요.

Amazon S3 버킷 설정

생성된 변환은 Amazon S3에 저장되며 사용자 AWS 계정에서 소유합니다. 모든 작업 스크립트가 현재 저장되어 있는 Amazon S3 자산 폴더(예: `s3://aws-glue-assets-<accountid>-<region>/transforms`)에 파일(json 및 py)을 업로드하면 새로운 사용자 지정 시각적 변환이 자동으로 생성됩니다. 사용자 지정 아이콘을 사용하는 경우 아이콘도 업로드합니다. 기본적으로 AWS Glue Studio에서는 동일한 S3 버킷의 /transforms 폴더에서 모든 .json 파일을 읽습니다.

단계 1. JSON 구성 파일 생성

JSON 구성 파일은 사용자 지정 시각적 변환을 정의하고 설명하는 데 필요합니다. 구성 파일의 스키마는 다음과 같습니다.

JSON 파일 구조

필드

- `name: string` - (필수) 변환을 식별하는 데 사용되는 변환 시스템 이름입니다. Python 변수 이름(식별자)에 설정된 것과 동일한 이름 지정 규칙을 따릅니다. 특히 문자 또는 밑줄로 시작하고 그 뒤에는 문자, 숫자 및/또는 밑줄로만 구성되어야 합니다.
- `displayName: string` - (선택 사항) AWS Glue Studio 시각적 작업 편집기에 표시되는 변환의 이름입니다. `displayName`을 지정하지 않은 경우 `name`이 AWS Glue Studio에서 변환의 이름으로 사용됩니다.
- `description: string` - (선택 사항) 변환 설명이 AWS Glue Studio에 표시되고 검색할 수 있습니다.
- `functionName: string` - (필수) Python 함수 이름은 Python 스크립트에서 호출할 함수를 식별하는 데 사용됩니다.
- `path: string` - (선택 사항) Python 소스 파일의 전체 Amazon S3 경로입니다. 지정하지 않은 경우 AWS Glue에서는 파일 이름 일치를 사용하여 .json 파일과 .py 파일을 함께 연결합니다. 예를 들어 `myTransform.json` JSON 파일은 동일한 Amazon S3 위치에 있는 Python 파일 `myTransform.py`와 연결됩니다.

- `parameters: Array of TransformParameter object` - (선택 사항) AWS Glue Studio 시각적 편집기에서 파라미터를 구성할 때 표시할 파라미터 목록입니다.

TransformParameter fields

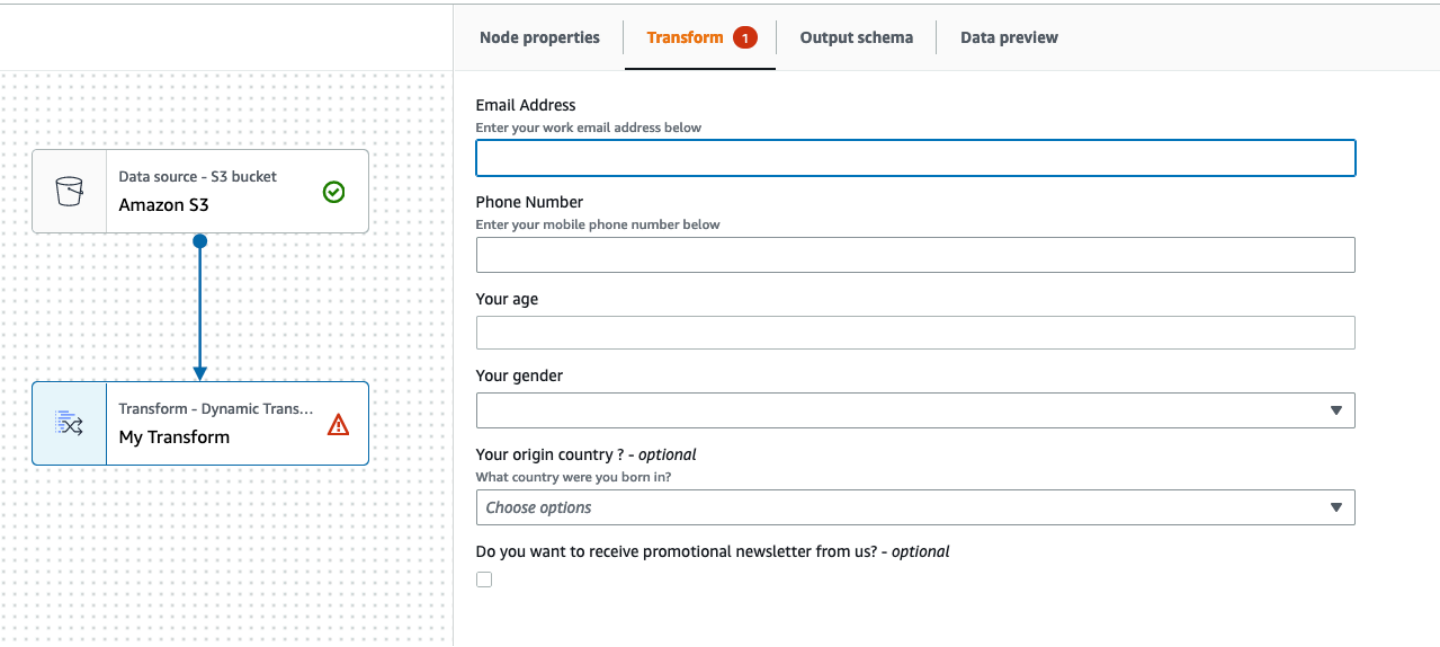
- `name: string` - (필수) Python 함수에 작업 스크립트에서 명명된 인수로 전달될 파라미터 이름입니다. Python 변수 이름(식별자)에 설정된 것과 동일한 이름 지정 규칙을 따릅니다. 특히 문자 또는 밑줄로 시작하고 그 뒤에는 문자, 숫자 및/또는 밑줄로만 구성되어야 합니다.
- `displayName: string` - (선택 사항) AWS Glue Studio 시각적 작업 편집기에 표시되는 변환의 이름입니다. `displayName`을 지정하지 않은 경우 `name`이 AWS Glue Studio에서 변환의 이름으로 사용됩니다.
- `type: string` - (필수) 일반적인 Python 데이터 유형을 허용하는 파라미터 유형입니다. 유효한 값: 'str' | 'int' | 'float' | 'list' | 'bool'.
- `isOptional: boolean` - (선택 사항) 파라미터가 선택 사항인지 여부를 결정합니다. 기본적으로 모든 파라미터는 필수입니다.
- `description: string` - (선택 사항) 사용자가 변환 파라미터를 구성하는 데 도움이 되는 설명이 AWS Glue Studio에 표시됩니다.
- `validationType: string` - (선택 사항) 이 파라미터의 유효성을 검사하는 방법을 정의합니다. 현재는 정규 표현식만 지원합니다. 기본적으로 검증 유형은 `RegularExpression`으로 설정됩니다.
- `validationRule: string` - (선택 사항) `validationType`을 `RegularExpression`으로 설정한 경우에 제출하기 전에 양식 입력을 검증하는 데 사용되는 정규 표현식입니다. 정규 표현식 구문은 RegExp Ecmascript 사양과 호환되어야 합니다.
- `validationMessage: string` - (선택 사항) 검증에 실패한 경우에 표시할 메시지입니다.
- `listOptions: An array of TransformParameterListOption object`, `string` 또는 문자열 값 'column' - (선택 사항) 선택 또는 다중 선택 UI 제어에 표시할 옵션입니다. 쉼표로 구분된 값 목록 또는 강력한 형식의 `TransformParameterListOption` JSON 객체 유형을 허용합니다. 또한 문자열 값 'column'을 지정하여 상위 노드 스키마의 열 목록을 동적으로 채울 수 있습니다.
- `listType: string` - (선택 사항) type = 'list'에 대한 옵션 유형을 정의합니다. 유효한 값: 'str' | 'int' | 'float' | 'list' | 'bool'. 일반적인 Python 데이터 유형을 허용하는 파라미터 유형입니다.

TransformParameterListOption 필드

- `value: string | int | float | bool` - (필수) 옵션 값입니다.
- `label: string` - (선택 사항) 선택 드롭다운에 표시되는 옵션 레이블입니다.

AWS Glue Studio의 파라미터 변환

.json 파일에 `isOptional`로 표시되지 않는 한 기본적으로 파라미터가 필요합니다. AWS Glue Studio에서는 파라미터가 Transform(변환) 탭에 표시됩니다. 이 예에서는 이메일 주소, 전화번호, 나이, 성별, 출신 국가와 같은 사용자 정의 파라미터를 보여줍니다.

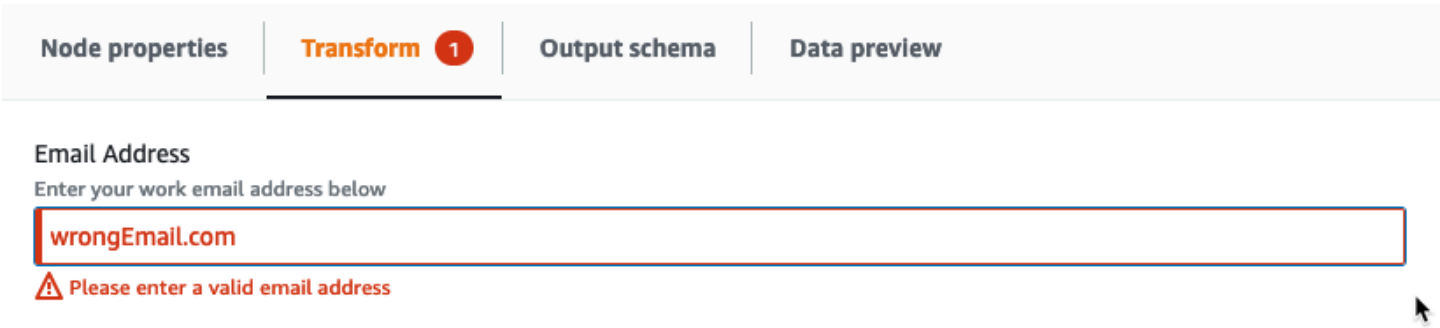


`validationMessage`에서 `validationRule` 파라미터를 지정하고 검증 메시지를 지정하여 json 파일에서 정규 표현식을 통해 AWS Glue Studio에서 일부 검증을 적용할 수 있습니다.

```
    "validationRule": "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
    "validationMessage": "Please enter a valid US number"
```

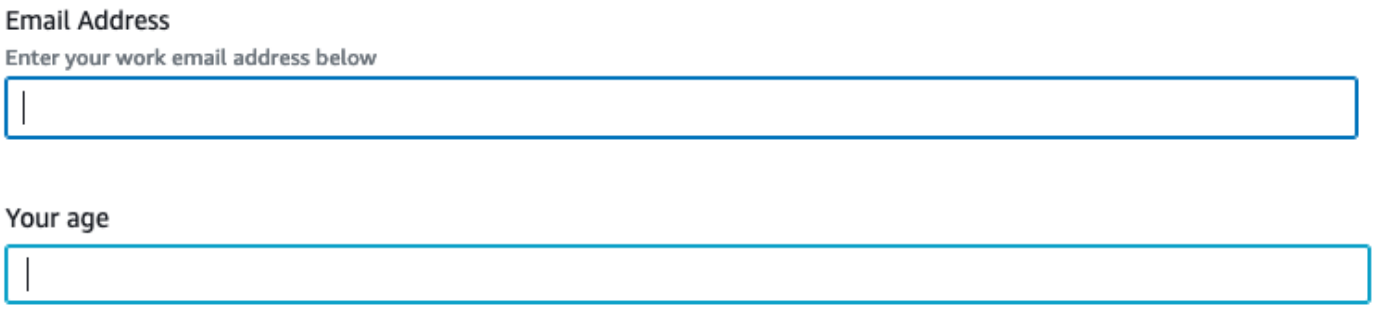
> **Note**
>
> 검증은 브라우저에서 수행되므로 정규 표현식 구문은 RegExp Ecmascript 사양과 호환되어야 합니다. Python 구문은 이러한 정규 표현식에 지원되지 않습니다.

검증을 추가하면 사용자가 잘못된 사용자 입력으로 작업을 저장하는 것을 방지할 수 있습니다. AWS Glue Studio에서는 예제에 표시된 대로 검증 메시지를 표시합니다.

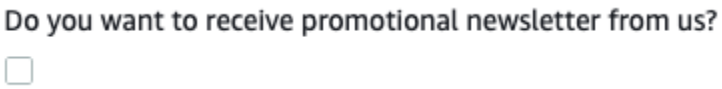


파라미터는 파라미터 구성에 따라 AWS Glue Studio에 표시됩니다.

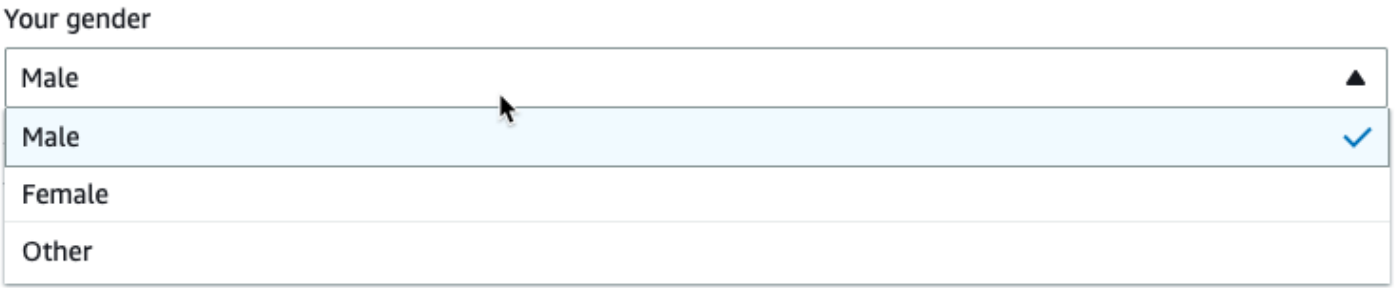
- `type`이 `str`, `int` 또는 `float`일 때 텍스트 입력 필드가 표시됩니다. 예를 들어 스크린샷은 'Email Address' 및 'Your age' 파라미터의 입력 필드를 보여줍니다.



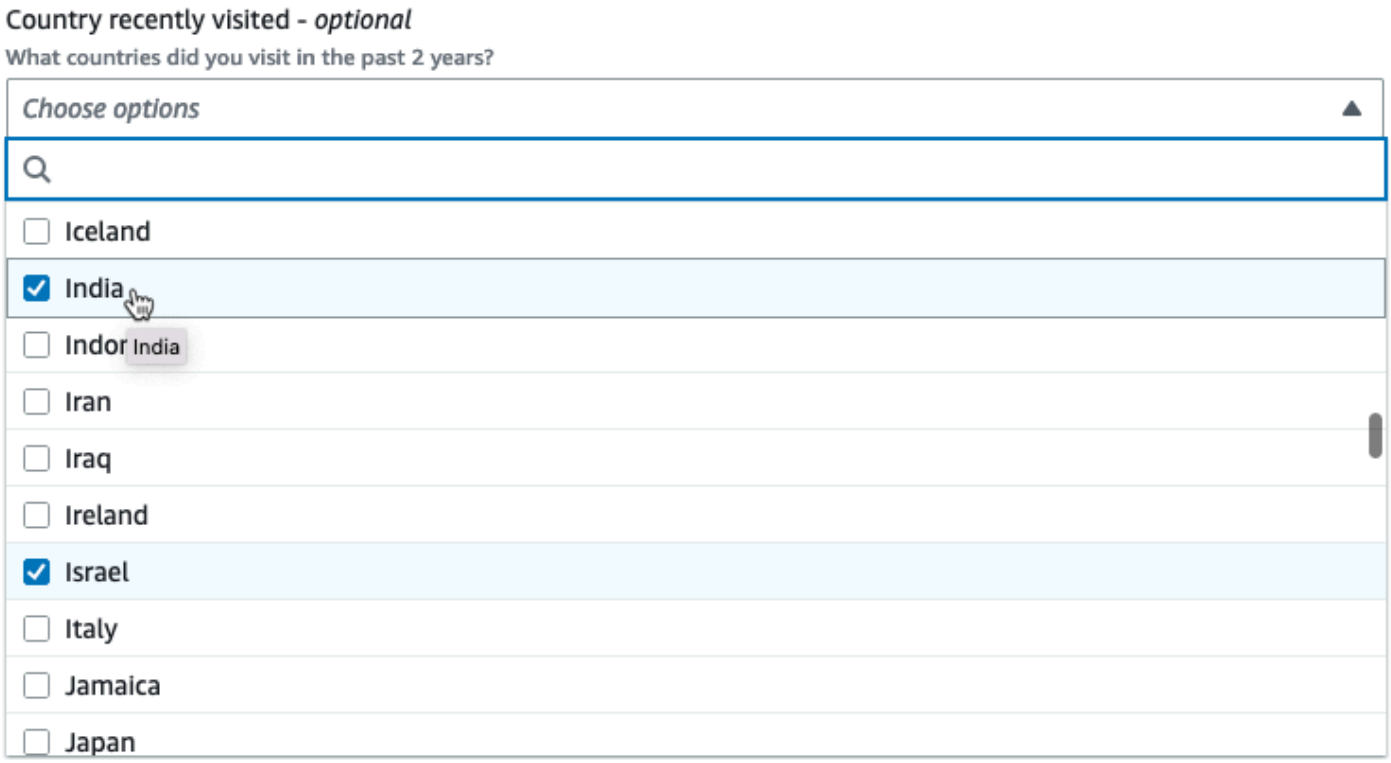
- `type`이 `bool`이면 확인란이 표시됩니다.



- `type`이 `str`이고 `listOptions`가 제공되면 단일 선택 목록이 표시됩니다.



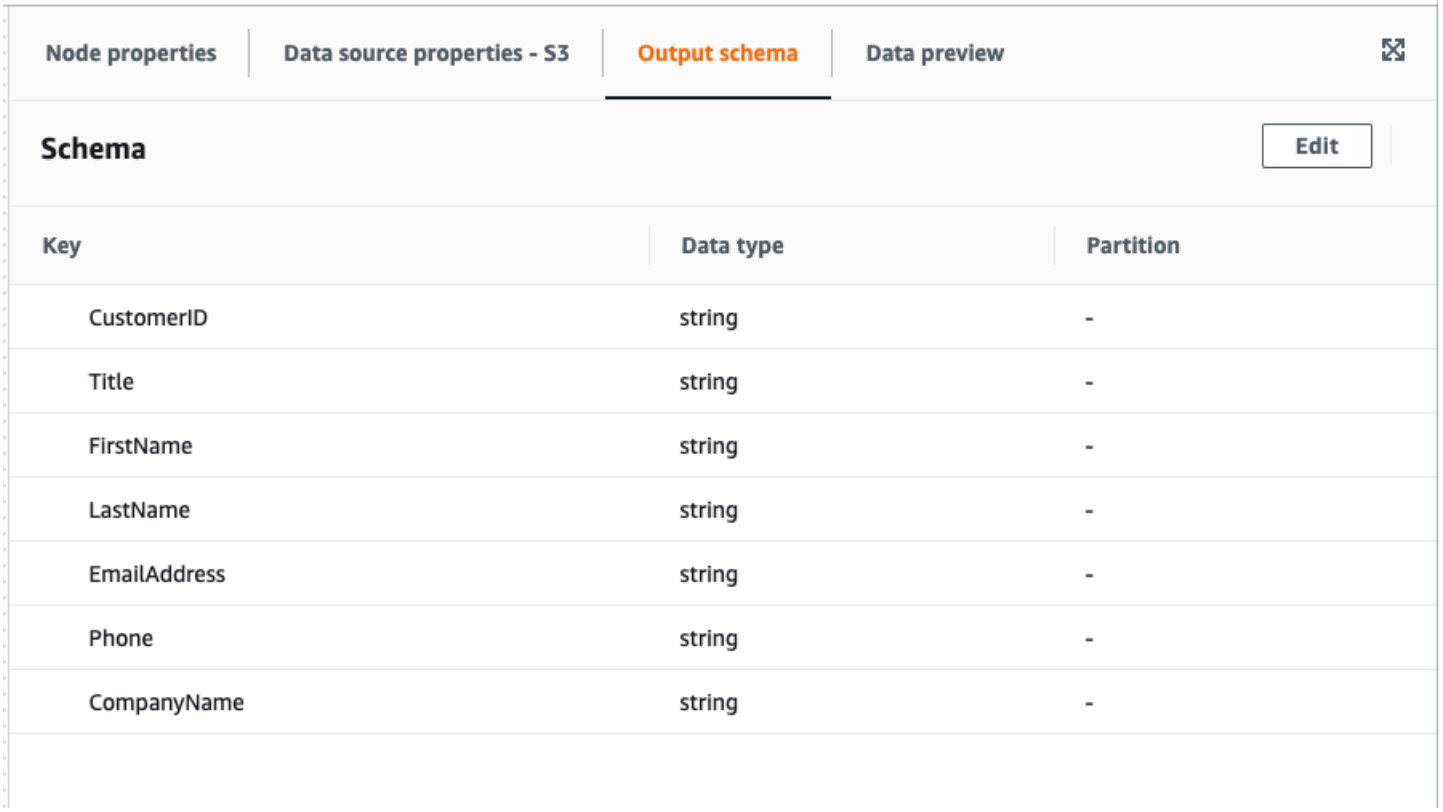
- `type`이 `list` 및 `listOptions`이고 `listType`이 제공되면 다중 선택 목록이 표시됩니다.



열 선택기를 파라미터로 표시

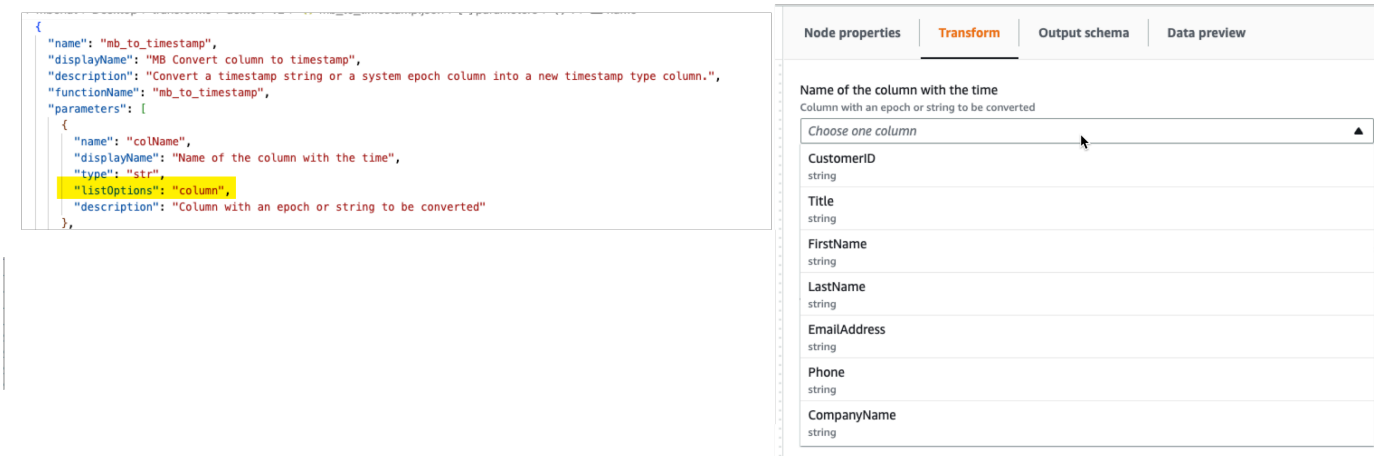
구성 시 사용자가 스키마에서 열을 선택해야 하는 경우 열 선택기를 표시할 수 있습니다. 이 경우 사용자가 열 이름을 입력하지 않아도 됩니다. `listOptions` 필드를 'column'으로 설정하면 AWS Glue Studio에서는 상위 노드 출력 스키마를 기반으로 열 선택기를 표시합니다. AWS Glue Studio에서는 단일 또는 다중 열 선택기를 표시할 수 있습니다.

다음 예제에서는 스키마를 사용합니다.



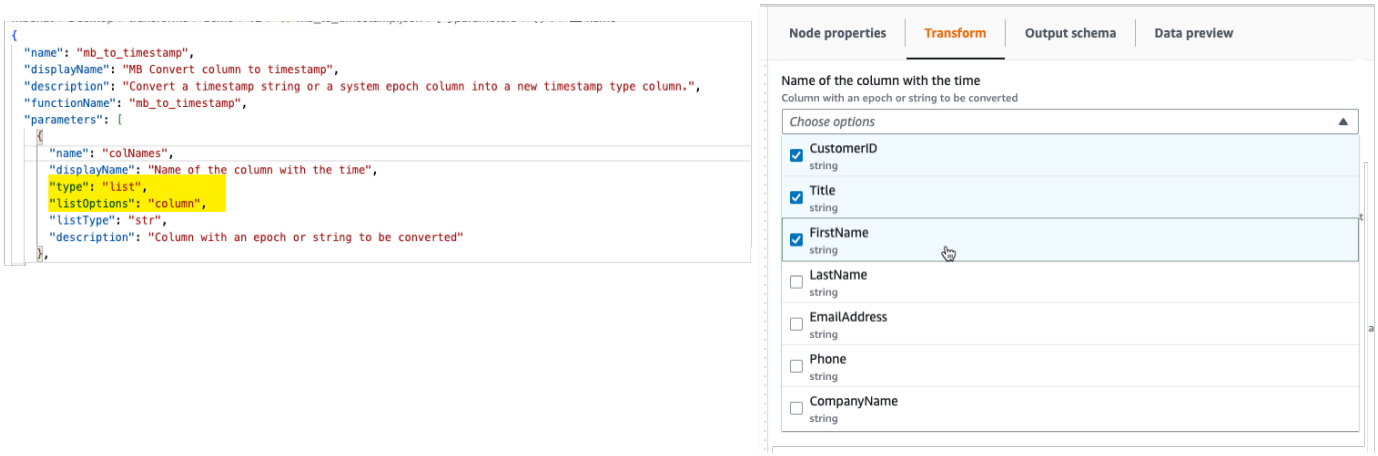
단일 열을 표시하도록 사용자 지정 시각적 변환 파라미터를 정의하려면:

1. JSON 파일에서 `parameters` 객체의 `listOptions` 값을 'column'으로 설정합니다. 이렇게 하면 사용자가 AWS Glue Studio의 선택 목록에서 열을 선택할 수 있습니다.



2. 파라미터를 다음과 같이 정의하여 여러 열을 선택할 수도 있습니다.

   - `listOptions: "column"`
   - `type: "list"`



단계 2. 변환 로직 구현

> **Note**
>
> 사용자 지정 시각적 변환은 Python 스크립트만 지원합니다. Scala는 지원되지 않습니다.

.json 구성 파일에 정의된 함수를 구현하는 코드를 추가하려면 Python 파일을 .json 파일과 동일한 위치에 이름은 동일하지만 ".py" 확장자로 저장하는 것이 좋습니다. AWS Glue Studio에서는 .json 파일과 .py 파일을 자동으로 연결하므로 구성 파일에 Python 파일의 경로를 지정할 필요가 없습니다.

Python 파일에서 명명된 파라미터를 구성하여 선언된 함수를 추가하고 `DynamicFrame`에서 사용하기 위해 함수를 등록합니다. 다음은 Python 파일의 예제입니다.

```
from awsglue import DynamicFrame

# self refers to the DynamicFrame to transform,
# the parameter names must match the ones defined in the config
# if it's optional, need to provide a default value
def myTransform(self, email, phone, age=None, gender="",
                      country="", promotion=False):
   resulting_dynf = # do some transformation on self
   return resulting_dynf

DynamicFrame.myTransform = myTransform
```

Python 코드를 가장 빠르게 개발하고 테스트하려면 AWS Glue 노트북을 사용하는 것이 좋습니다. AWS Glue Studio에서 노트북 시작하기를 참조하세요.

변환 로직을 구현하는 방법을 설명하기 위해 아래 예제의 사용자 지정 시각적 변환은 들어오는 데이터를 필터링하여 특정 미국 주와 관련된 데이터만 유지하는 변환입니다. .json 파일에는 `functionName`에 대한 파라미터(`custom_filter_state`)와 두 개의 인수(유형이 “str”인 “state” 및 “colName”)가 포함되어 있습니다.

예제 구성 .json 파일은 다음과 같습니다.

```
{
"name": "custom_filter_state",
"displayName": "Filter State",
"description": "A simple example to filter the data to keep only the state indicated.",
"functionName": "custom_filter_state",
"parameters": [
   {
    "name": "colName",
    "displayName": "Column name",
    "type": "str",
    "description": "Name of the column in the data that holds the state postal code"
   },
   {
    "name": "state",
    "displayName": "State postal code",
    "type": "str",
    "description": "The postal code of the state whole rows to keep"
   }
  ]
}
```

Python에서 컴패니언 스크립트를 구현하려면

1. AWS Glue 노트북을 시작하고 시작할 세션에 제공된 초기 셀을 실행합니다. 초기 셀을 실행하면 필요한 기본 구성 요소가 생성됩니다.
2. 예제에 설명된 대로 필터링을 수행하는 함수를 생성하고 `DynamicFrame`에 등록합니다. 아래 코드를 복사하여 AWS Glue 노트북의 셀에 붙여 넣습니다.

   ```
   from awsglue import DynamicFrame
   ```

```
def custom_filter_state(self, colName, state):
    return self.filter(lambda row: row[colName] == state)

DynamicFrame.custom_filter_state = custom_filter_state
```

3. 샘플 데이터를 생성하거나 로드하여 동일한 셀 또는 새 셀에서 코드를 테스트합니다. 새 셀에 샘플 데이터를 추가하는 경우 셀을 실행하는 것을 잊지 마세요. 예:

```
# A few of rows of sample data to test
data_sample = [
    {"state": "CA", "count": 4},
    {"state": "NY", "count": 2},
    {"state": "WA", "count": 3}
]
df1 = glueContext.sparkSession.sparkContext.parallelize(data_sample).toDF()
dynf1 = DynamicFrame.fromDF(df1, glueContext, None)
```

4. 다른 인수를 사용하여 “custom_filter_state”를 검증하려면 테스트하세요.

```
[14]: dynf1.custom_filter_state("state", "NY").show()
      {"count": 2, "state": "NY"}
```

5. 여러 테스트를 실행한 후 확장명이 .py인 코드를 저장하고 .json 파일 이름을 미러링하는 이름으로 .py 파일의 이름을 지정합니다. .py 파일과 .json 파일은 동일한 변환 폴더에 있어야 합니다.

   다음 코드를 복사하여 파일에 붙여넣고 .py 파일 확장명으로 이름을 바꿉니다.

```
from awsglue import DynamicFrame

def custom_filter_state(self, colName, state):
    return self.filter(lambda row: row[colName] == state)

DynamicFrame.custom_filter_state = custom_filter_state
```

6. AWS Glue Studio에서 시각적 작업을 열고 사용 가능한 Transforms(변환) 목록에서 선택하여 작업에 변환을 추가합니다.

Python 스크립트 코드에서 이 변환을 다시 사용하려면 “참조된 파일 경로” 아래의 작업에 있는 .py 파일에 Amazon S3 경로를 추가하고 스크립트에서 Python 파일 이름(확장자 제외)을 파일 상단에 추가하여 가져옵니다. 예를 들면 다음과 같습니다. `import` <name of the file (without the extension)>

단계 3. AWS Glue Studio에서 사용자 지정 시각적 변환을 검증하고 문제를 해결합니다.

AWS Glue Studio에서는 사용자 지정 시각적 변환이 AWS Glue Studio에 로드되기 전에 JSON 구성 파일을 검증합니다. 검증에는 다음이 포함됩니다.

- 필수 필드의 존재 여부
- JSON 형식 검증
- 올바르지 않거나 유효하지 않은 파라미터
- .py 파일과 .json 파일이 동일한 Amazon S3 경로에 모두 존재하는지 여부
- .py와 .json의 파일 이름 일치

검증이 성공하면 변환이 시각적 편집기의 사용 가능한 Actions(작업) 목록에 나열됩니다. 사용자 지정 아이콘이 제공된 경우 작업 옆에 표시되어야 합니다.

검증에 실패할 경우 AWS Glue Studio에서는 사용자 지정 시각적 변환을 로드하지 않습니다.

4단계. 필요에 따라 사용자 지정 시각적 변환 업데이트

변환이 해당 json 정의를 따르는 한 변환 스크립트를 생성하고 사용한 이후에 업데이트할 수 있습니다.

- DynamicFrame에 할당할 때 사용되는 이름은 json `functionName`과 일치해야 합니다.
- 함수 인수는 단계 1. JSON 구성 파일 생성에 설명된 대로 json 파일에 정의되어야 합니다.
- 작업이 Python 파일에 직접 의존하므로 Python 파일의 Amazon S3 경로는 변경할 수 없습니다.

> **Note**
>
> 업데이트가 필요한 경우 스크립트와 .json 파일이 지속적으로 업데이트되고 모든 시각적 작업이 새 변환과 함께 올바르게 다시 저장되었는지 확인하세요. 업데이트 후 시각적 작업을 저장하지 않으면 업데이트가 적용 및 검증되지 않습니다. Python 스크립트 파일의 이름이 바뀌거나 .json 파일 옆에 배치되지 않은 경우 .json 파일에 전체 경로를 지정해야 합니다.

사용자 지정 아이콘

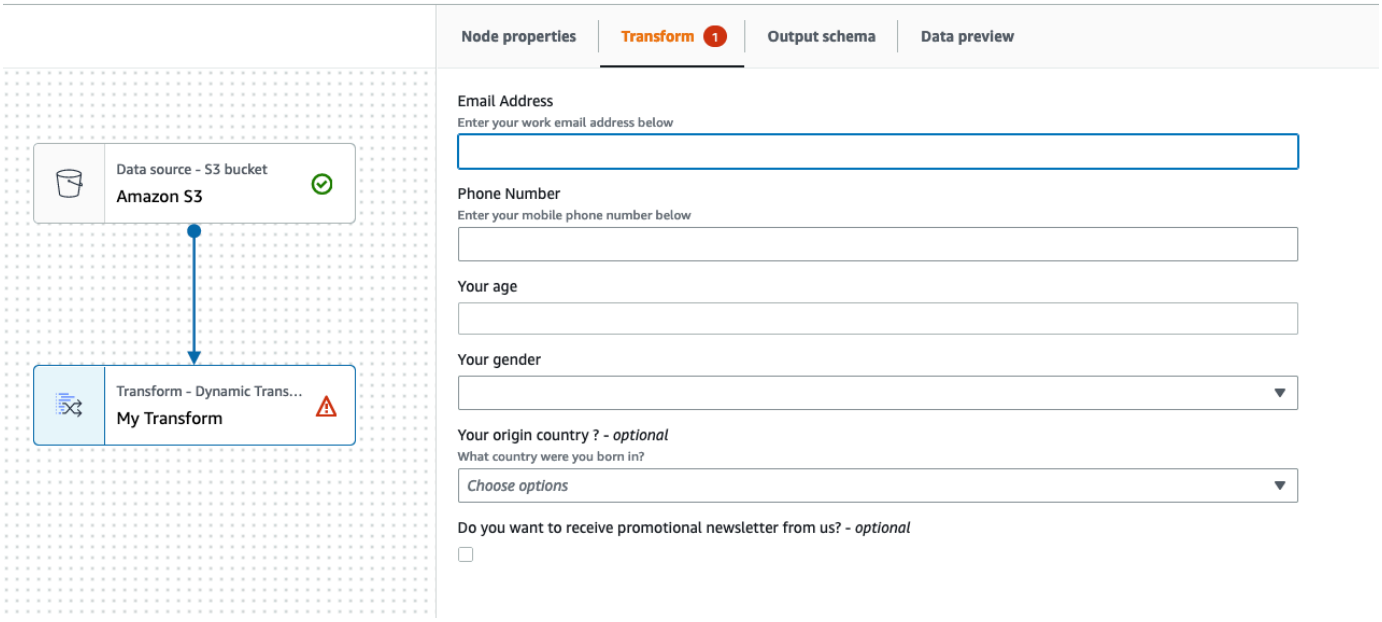
워크플로의 일부에서 작업의 기본 아이콘이 시각적으로 구분되지 않는다고 판단되면 the section called “ 사용자 지정 시각적 변환 시작하기 ”에 설명된 대로 사용자 지정 아이콘을 제공할 수 있습니다. Amazon S3에 호스팅된 해당 SVG를 업데이트하여 아이콘을 업데이트할 수 있습니다.

최상의 결과를 얻으려면 Cloudscape Design System의 지침에 따라 32x32px로 표시되도록 이미지를 설계합니다. Cloudscape 지침에 대한 자세한 내용은 Cloudscape 설명서를 참조하세요.

5단계. AWS Glue Studio에서 사용자 지정 시각적 변환 사용

AWS Glue Studio에서 사용자 지정 시각적 변환을 사용하려면 구성 및 소스 파일을 업로드한 다음 Action(작업) 메뉴에서 변환을 선택합니다. 값 또는 입력이 필요한 파라미터는 Transform(변환) 탭에서 사용할 수 있습니다.

1. 두 파일(Python 소스 파일 및 JSON 구성 파일)을 작업 스크립트가 저장되는 Amazon S3 자산 폴더에 업로드합니다. 기본적으로 AWS Glue에서는 동일한 Amazon S3 버킷의 /transforms 폴더에서 모든 .json 파일을 가져옵니다.
2. Action(작업) 메뉴에서 사용자 지정 시각적 변환을 선택합니다. .json 구성 파일에 지정된 변환 `displayName` 또는 이름으로 이름이 지정됩니다.
3. 구성 파일에 구성된 모든 파라미터의 값을 입력합니다.



사용 예제:

다음은 .json 구성 파일에 있는 모든 가능한 파라미터의 예입니다.

```
{
  "name": "MyTransform",
  "displayName": "My Transform",
  "description": "This transform description will be displayed in UI",
  "functionName": "myTransform",
  "parameters": [
      {
      "name": "email",
      "displayName": "Email Address",
      "type": "str",
      "description": "Enter your work email address below",
      "validationType": "RegularExpression",
      "validationRule": "^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*(\\.\\w{2,3})+$",
      "validationMessage": "Please enter a valid email address"
    },
    {
      "name": "phone",
      "displayName": "Phone Number",
      "type": "str",
      "description": "Enter your mobile phone number below",
      "validationRule": "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
      "validationMessage": "Please enter a valid US number"
    },
    {
      "name": "age",
      "displayName": "Your age",
      "type": "int",
      "isOptional": true
    },
    {
      "name": "gender",
      "displayName": "Your gender",
      "type": "str",
      "listOptions": [
            {"label": "Male", "value": "male"},
            {"label": "Female", "value": "female"},
            {"label": "Other", "value": "other"}
        ],
      "isOptional": true
    },
```

```
    {
      "name": "country",
      "displayName": "Your origin country ?",
      "type": "list",
      "listOptions": "Afghanistan,Albania,Algeria,American
 Samoa,Andorra,Angola,Anguilla,Antarctica,Antigua and
 Barbuda,Argentina,Armenia,Aruba,Australia,Austria,Azerbaijan,Bahamas,Bahrain,Bangladesh,Barbad
 and Herzegovina,Botswana,Bouvet Island,Brazil,British Indian Ocean Territory,Brunei
 Darussalam,Bulgaria,Burkina Faso,Burundi,Cambodia,Cameroon,Canada,Cape
 Verde,Cayman Islands,Central African Republic,Chad,Chile,China,Christmas
 Island,Cocos (Keeling Islands),Colombia,Comoros,Congo,Cook Islands,Costa
 Rica,Cote D'Ivoire (Ivory Coast),Croatia (Hrvatska,Cuba,Cyprus,Czech
 Republic,Denmark,Djibouti,Dominica,Dominican Republic,East Timor,Ecuador,Egypt,El
 Salvador,Equatorial Guinea,Eritrea,Estonia,Ethiopia,Falkland Islands (Malvinas),Faroe
 Islands,Fiji,Finland,France,France,Metropolitan,French Guiana,French Polynesia,French
 Southern
 Territories,Gabon,Gambia,Georgia,Germany,Ghana,Gibraltar,Greece,Greenland,Grenada,Guadeloupe,G
Bissau,Guyana,Haiti,Heard and McDonald Islands,Honduras,Hong
 Kong,Hungary,Iceland,India,Indonesia,Iran,Iraq,Ireland,Israel,Italy,Jamaica,Japan,Jordan,Kazak
 (North),Korea
 (South),Kuwait,Kyrgyzstan,Laos,Latvia,Lebanon,Lesotho,Liberia,Libya,Liechtenstein,Lithuania,Lu
 Islands,Martinique,Mauritania,Mauritius,Mayotte,Mexico,Micronesia,Moldova,Monaco,Mongolia,Mont
 Antilles,New Caledonia,New Zealand,Nicaragua,Niger,Nigeria,Niue,Norfolk
 Island,Northern Mariana Islands,Norway,Oman,Pakistan,Palau,Panama,Papua
 New Guinea,Paraguay,Peru,Philippines,Pitcairn,Poland,Portugal,Puerto
 Rico,Qatar,Reunion,Romania,Russian Federation,Rwanda,Saint Kitts and Nevis,Saint
 Lucia,Saint Vincent and The Grenadines,Samoa,San Marino,Sao Tome and Principe,Saudi
 Arabia,Senegal,Seychelles,Sierra Leone,Singapore,Slovak Republic,Slovenia,Solomon
 Islands,Somalia,South Africa,S. Georgia and S. Sandwich Isls.,Spain,Sri
 Lanka,St. Helena,St. Pierre and Miquelon,Sudan,Suriname,Svalbard and Jan Mayen
 Islands,Swaziland,Sweden,Switzerland,Syria,Tajikistan,Tanzania,Thailand,Togo,Tokelau,Tonga,Tri
 and Tobago,Tunisia,Turkey,Turkmenistan,Turks and Caicos
 Islands,Tuvalu,Uganda,Ukraine,United Arab Emirates,United Kingdom
 (Britain / UK),United States of America (USA),US Minor Outlying
 Islands,Uruguay,Uzbekistan,Vanuatu,Vatican City State (Holy See),Venezuela,Viet
 Nam,Virgin Islands (British),Virgin Islands (US),Wallis and Futuna Islands,Western
 Sahara,Yemen,Yugoslavia,Zaire,Zambia,Zimbabwe",
      "description": "What country were you born in?",
      "listType": "str",
      "isOptional": true
    },
    {
      "name": "promotion",
      "displayName": "Do you want to receive promotional newsletter from us?",
```

```
      "type": "bool",
      "isOptional": true
    }
  ]
}
```

사용자 지정 시각적 스크립트의 예

다음 예에서는 동일한 변환을 수행합니다. 하지만 두 번째 예제(SparkSQL)가 가장 깔끔하고 가장 효율적이며, 그 다음은 Pandas UDF이며, 첫 번째 예제의 하위 수준 매핑이 마지막입니다. 다음 예제는 두 열을 합산하는 간단한 변환의 전체 예입니다.

```
from awsglue import DynamicFrame

# You can have other auxiliary variables, functions or classes on this file, it won't
 affect the runtime
def record_sum(rec, col1, col2, resultCol):
    rec[resultCol] = rec[col1] + rec[col2]
    return rec


# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    # The mapping will alter the columns order, which could be important
    fields = [field.name for field in self.schema()]
    if resultCol not in fields:
        # If it's a new column put it at the end
        fields.append(resultCol)
    return self.map(lambda record: record_sum(record, col1, col2,
 resultCol)).select_fields(paths=fields)


# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

다음 예제는 SparkSQL API를 활용하는 동일한 변환입니다.

```
from awsglue import DynamicFrame

# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    df = self.toDF()
    return DynamicFrame.fromDF(
        df.withColumn(resultCol, df[col1] + df[col2]) # This is the conversion logic
        , self.glue_ctx, self.name)


# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

다음 예제에서는 동일한 변환을 사용하지만 pandas UDF를 사용하므로 일반 UDF를 사용하는 것보다 더 효율적입니다. pandas UDF 작성에 대한 자세한 내용은 Apache Spark SQL 설명서를 참조하세요.

```
from awsglue import DynamicFrame
import pandas as pd
from pyspark.sql.functions import pandas_udf

# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    @pandas_udf("integer")  # We need to declare the type of the result column
    def add_columns(value1: pd.Series, value2: pd.Series) # pd.Series:
        return value1 + value2

    df = self.toDF()
    return DynamicFrame.fromDF(
        df.withColumn(resultCol, add_columns(col1, col2)) # This is the conversion
 logic
        , self.glue_ctx, self.name)

# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

비디오

다음 동영상은 시각적 사용자 지정 변환을 소개하고 사용 방법을 보여줍니다.

AWS Glue Studio에서 데이터 레이크 프레임워크 사용

개요

오픈 소스 데이터 레이크 프레임워크는 Amazon S3에 빌드된 데이터 레이크에 저장된 파일의 증분 데이터 처리를 간소화합니다. AWS Glue 3.0 이상에서는 다음과 같은 오픈 소스 데이터 레이크 스토리지 프레임워크를 지원합니다.

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

AWS Glue 4.0을 기준으로 AWS Glue에서는 Amazon S3에 저장된 데이터를 트랜잭션 형태로 일관성 있게 읽고 쓸 수 있도록 이러한 프레임워크에 대한 기본 지원을 제공합니다. AWS Glue 작업에 이러한 프레임워크를 사용하기 위해 별도의 커넥터를 설치하거나 추가 구성 단계를 완료할 필요가 없습니다.

데이터 레이크 프레임워크는 Spark 스크립트 편집기 작업을 통해 AWS Glue Studio 내에서 소스 또는 대상으로 사용될 수 있습니다. Apache Hudi, Apache Iceberg, Delta Lake를 사용하는 방법에 대한 자세한 내용은 AWS Glue ETL 작업에 데이터 레이크 프레임워크 사용을 참조하세요.

AWS Glue 스트리밍 소스에서 오픈 테이블 형식 생성

AWS Glue 스트리밍 ETL 작업은 스트리밍 소스의 데이터를 지속적으로 소비하고, 전송 중인 데이터를 정리 및 변환하여 몇 초 만에 분석에 사용할 수 있도록 합니다.

AWS 요구 사항을 지원하는 다양한 서비스를 제공합니다. Database Migration Service와 같은 AWS 데이터베이스 복제 서비스는 일반적으로 데이터 레이크의 스토리지 계층을 호스팅하는 Amazon S3로 원본 시스템의 데이터를 복제할 수 있습니다. 온라인 소스 애플리케이션을 지원하는 관계형 데이터베이스 관리 시스템(RDBMS)에서 업데이트를 적용하는 것은 간단하지만 데이터 레이크에 이 CDC 프로세스를 적용하기는 어렵습니다. 오픈 소스 데이터 관리 프레임워크는 증분 데이터 처리 및 데이터 파이프라인 개발을 간소화하는 데 유용한 옵션입니다.

자세한 내용은 다음을 참조하세요.

- AWS Glue 스트리밍을 사용하여 Apache Hudi 기반의 실시간에 가까운 트랜잭션 데이터 레이크를 만들 수 있습니다

- Build a real-time GDPR-aligned Apache Iceberg data lake

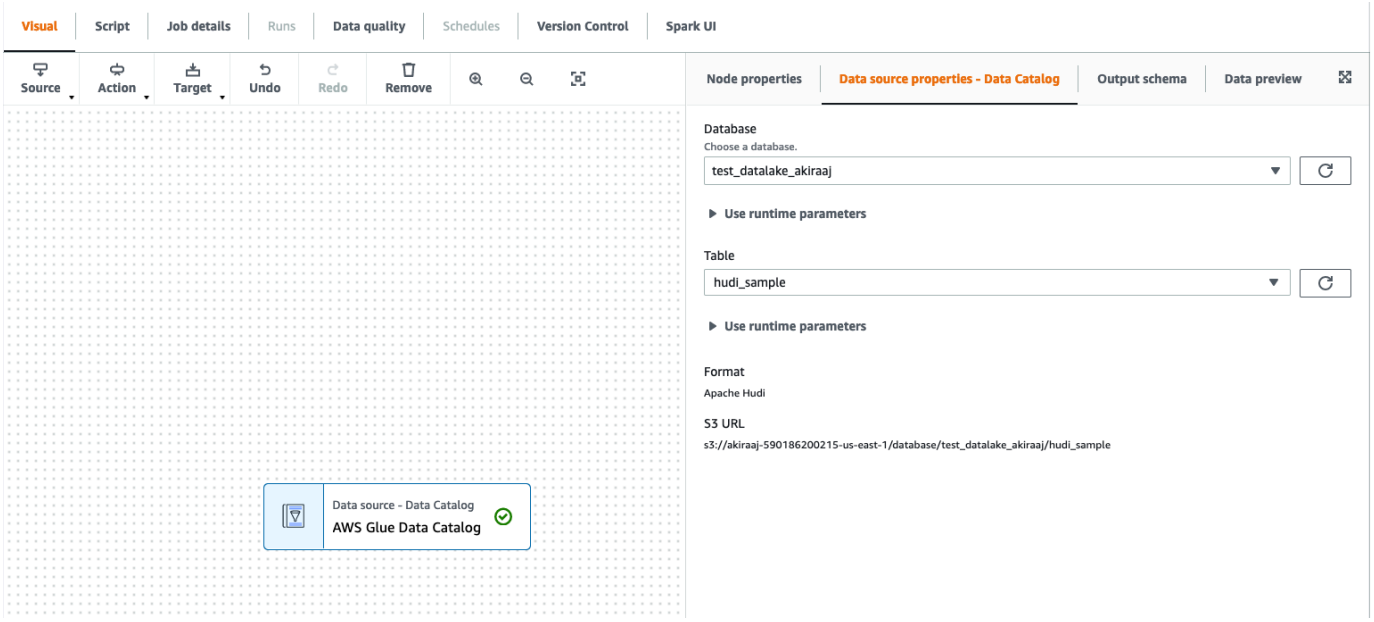
AWS Glue Studio에서 Hudi 프레임워크 사용

작업을 생성하거나 편집할 때 AWS Glue Studio는 사용 중인 AWS Glue의 버전에 따라 해당하는 Hudi 라이브러리를 자동으로 추가합니다. 자세한 내용은 AWS Glue에서 Hudi 프레임워크 사용을 참조하세요.

데이터 카탈로그 데이터 소스에서 Apache Hudi 프레임워크 사용

작업에 Hudi 데이터 소스 형식을 추가하려면

1. 소스 메뉴에서 AWS Glue Studio 데이터 카탈로그를 선택합니다.
2. 데이터 소스 속성 탭에서 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식 유형을 Apache Hudi와 Amazon S3 URL로 표시합니다.



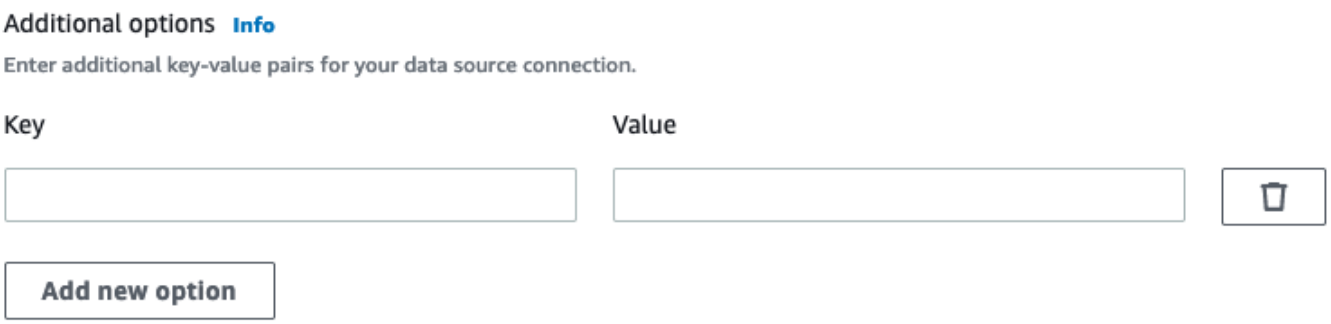
Amazon S3 데이터 소스에서 Hudi 프레임워크 사용

1. 소스 메뉴에서 Amazon S3를 선택합니다.
2. Amazon S3 소스 유형으로 데이터 카탈로그 테이블을 선택한 경우 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식을 Apache Hudi와 Amazon S3 URL로 표시합니다.

4. Amazon S3 소스 유형으로 Amazon S3 위치를 선택한 경우 Amazon S3 찾아보기를 클릭하여 Amazon S3 URL을 선택합니다.
5. 데이터 형식에서 Apache Hudi를 선택합니다.

> **Note**
>
> AWS Glue Studio가 선택한 Amazon S3 폴더 또는 파일에서 스키마를 유추할 수 없는 경우 추가 옵션을 선택하여 새 폴더 또는 파일을 선택합니다.
> 추가 옵션의 스키마 추론 아래에서 다음 옵션을 선택합니다.
>
> - AWS Glue Studio에서 통해 샘플 파일을 자동으로 선택하도록 하겠습니다. AWS Glue Studio는 스키마를 추론할 수 있도록 Amazon S3 위치에서 샘플 파일을 선택합니다. 자동 샘플링된 파일 필드에서 자동으로 선택된 파일을 볼 수 있습니다.
> - Amazon S3에서 샘플 파일을 선택합니다. Amazon S3 찾아보기를 클릭하여 사용할 Amazon S3 파일을 선택합니다.

6. 스키마 추론을 클릭합니다. 그런 다음 출력 스키마 탭을 클릭하여 출력 스키마를 볼 수 있습니다.
7. 키-값 페어를 입력하려면 추가 옵션을 선택합니다.



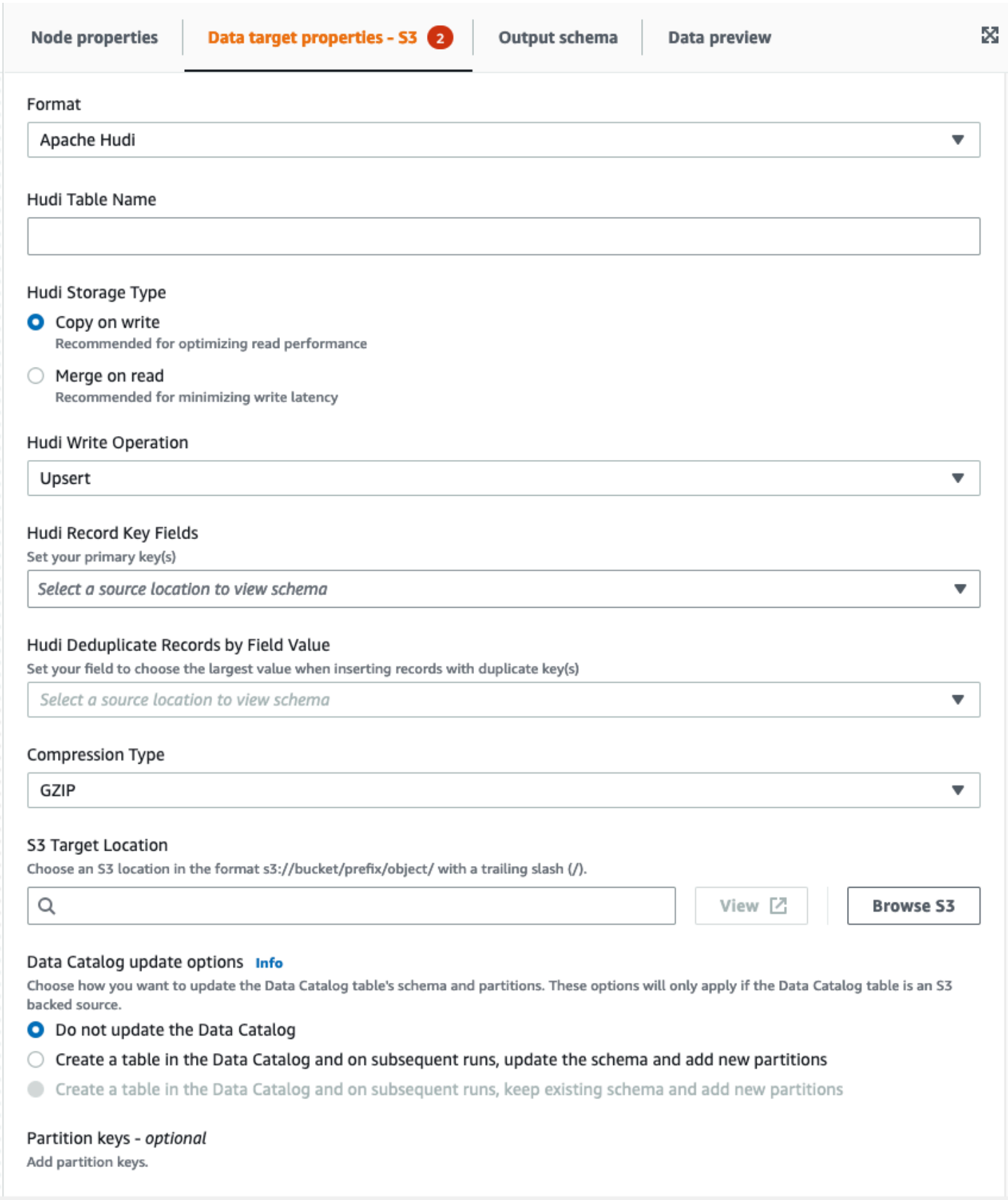
데이터 대상에서 Apache Hudi 프레임워크 사용

데이터 카탈로그 데이터 대상에서 Apache Hudi 프레임워크 사용

1. 대상 메뉴에서 AWS Glue Studio 데이터 카탈로그를 선택합니다.
2. 데이터 소스 속성 탭에서 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식 유형을 Apache Hudi와 Amazon S3 URL로 표시합니다.

Amazon S3 데이터 대상에서 Apache Hudi 프레임워크 사용

값을 입력하거나 사용 가능한 옵션 중에서 선택하여 Apache Hudi 형식을 구성합니다. Apache Hudi에 대한 자세한 내용은 Apache Hudi 설명서를 참조하세요.


Node properties
Data target properties - S3 2
Output schema
Data preview
Format
Apache Hudi
Hudi Table Name
Hudi Storage Type
Copy on write
Recommended for optimizing read performance
Merge on read
Recommended for minimizing write latency
Hudi Write Operation
Upsert
Hudi Record Key Fields
Set your primary key(s)
Select a source location to view schema
Hudi Deduplicate Records by Field Value
Set your field to choose the largest value when inserting records with duplicate key(s)
Select a source location to view schema
Compression Type
GZIP
S3 Target Location
Choose an S3 location in the format s3://bucket/prefix/object/ with a trailing slash (/).
View
Browse S3
Data Catalog update options Info
Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog table is an S3 backed source.
Do not update the Data Catalog
Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions
Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions
Partition keys - optional
Add partition keys.

- Hudi 테이블 이름 - HUDI 테이블의 이름입니다.
- Hudi 스토리지 유형 - 다음 두 가지 옵션 중에서 선택합니다.
  - 쓸 때 복사 - 읽기 성능을 최적화하기 위해 권장됩니다. Hudi 스토리지의 기본 유형입니다. 업데이트할 때마다 쓰기 중에 새 버전의 파일이 생성됩니다.
  - 읽을 때 병합 - 쓰기 지연 시간을 최소화하기 위해 권장됩니다. 업데이트는 행 기반 delta 파일에 기록되며 새 버전의 열 형식 파일을 작성할 때 필요에 따라 압축됩니다.
- Hudi 쓰기 작업 - 다음 옵션 중에서 선택합니다.
  - Upsert - 인덱스를 조회하여 입력 레코드를 먼저 삽입 또는 업데이트로 태깅하는 기본 작업입니다. 기존 데이터를 업데이트하는 경우에 권장됩니다.
  - 삽입 - 레코드를 삽입하지만 기존 레코드를 확인하지 않으므로 중복이 발생할 수 있습니다.
  - 대량 삽입 - 레코드를 삽입하며 대량의 데이터에 권장됩니다.
- Hudi 레코드 키 필드 - 검색 창을 사용하여 기본 레코드 키를 검색하고 선택합니다. Hudi의 레코드는 한 쌍의 레코드 키와 해당 레코드가 속한 파티션 경로인 프라이머리 키로 식별됩니다.
- Hudi 사전 결합 필드 - 실제 쓰기 전에 사전 결합에 사용되는 필드입니다. 두 레코드에 동일한 키 값이 있는 경우 AWS Glue Studio는 사전 결합 필드에 대해 가장 큰 값을 가진 레코드를 선택합니다. 증분 값(예: updated_at)이 속한 필드를 설정합니다.
- 압축 유형 - 압축 유형 옵션(비압축, GZIP, LZO 또는 Snappy) 중 하나를 선택합니다.
- Amazon S3 대상 위치 - S3 찾아보기를 클릭하여 Amazon S3 대상 위치를 선택합니다.
- 데이터 카탈로그 업데이트 옵션 - 다음 옵션 중에 선택합니다.
  - [데이터 카탈로그 업데이트 안 함(Do not update the Data Catalog)]: (기본값) 스키마가 변경되거나 새 파티션이 추가된 경우에도 작업에서 데이터 카탈로그를 업데이트하지 않으려면 이 옵션을 선택합니다.
  - 데이터 카탈로그에 테이블 생성, 후속 실행 시 스키마 업데이트 및 새 파티션 추가: 이 옵션을 선택하면 작업이 처음 실행될 때 데이터 카탈로그에 테이블이 생성됩니다. 후속 작업 실행 시 스키마가 변경되거나 새 파티션이 추가되면 작업이 데이터 카탈로그 테이블을 업데이트합니다.

    또한 데이터 카탈로그에서 데이터베이스를 선택하고 테이블 이름을 입력해야 합니다.
  - [데이터 카탈로그에 테이블 생성, 기존 스키마 유지 및 새 파티션 추가(Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions)]: 이 옵션을 선택하면 작업이 처음 실행될 때 데이터 카탈로그에 테이블이 생성됩니다. 후속 작업 실행 시 작업은 새 파티션을 추가하기 위해서만 데이터 카탈로그 테이블을 업데이트합니다.

    또한 데이터 카탈로그에서 데이터베이스를 선택하고 테이블 이름을 입력해야 합니다.

- [파티션 키(Partition keys)]: 출력에서 파티션 키로 사용할 열을 선택합니다. 파티션 키를 더 추가하려면 [파티션 키 추가(Add a partition key)]를 선택합니다.
- 추가 옵션 - 필요에 따라 키-값 페어를 입력합니다.

AWS Glue Studio를 통해 코드 생성

작업이 저장될 때 Hudi 소스 또는 대상이 감지되면 다음 작업 파라미터가 작업에 추가됩니다.

- `--datalake-formats` - 시각적 작업에서 감지된 데이터 레이크 형식의 개별 목록('형식'을 선택하여 직접적으로 또는 데이터 레이크가 지원하는 카탈로그 테이블을 선택하여 간접적으로)
- `--conf` - `--datalake-formats`의 값을 기반으로 생성됩니다. 예를 들어, `--datalake-formats`의 값이 'hudi'인 경우 AWS Glue에 이 파라미터에 대한 `spark.serializer=org.apache.spark.serializer.KryoSerializer –conf spark.sql.hive.convertMetastoreParquet=false` 값이 생성됩니다.

AWS Glue 제공 라이브러리 재정의

AWS Glue에서 지원하지 않는 Hudi 버전을 사용하려는 경우 자체 Hudi 라이브러리 JAR 파일을 지정하면 됩니다. 자체 JAR 파일을 사용하려면:

- `--extra-jars` 작업 파라미터를 사용합니다. 예: `'--extra-jars': 's3pathtojarfile.jar'`. 자세한 내용을 알아보려면 AWS Glue 작업 파라미터를 참조하세요.
- `--datalake-formats` 작업 파라미터의 값으로 `hudi`를 포함하지 마세요. 빈 문자열을 값으로 입력하면 AWS Glue에서 데이터 레이크 라이브러리를 자동으로 제공하지 않습니다. 자세한 내용은 AWS Glue에서 Hudi 프레임워크 사용을 참조하세요.

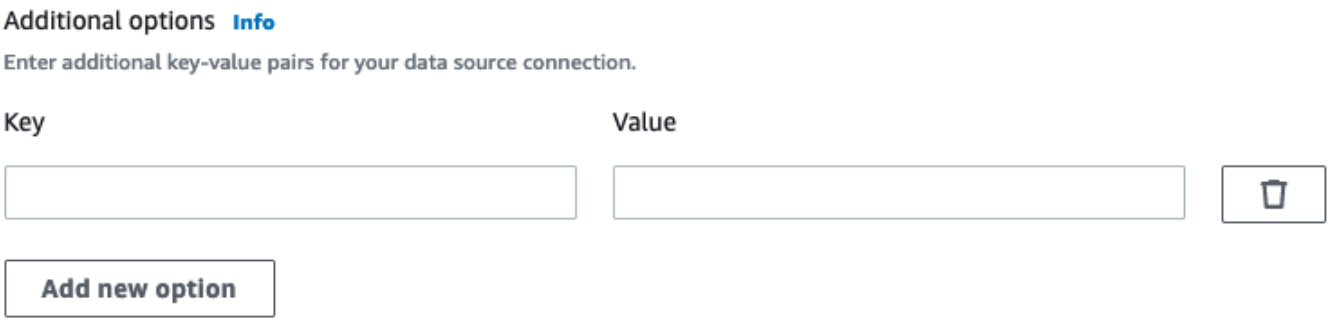
AWS Glue Studio에서 Delta Lake 프레임워크 사용

데이터 소스에서 Delta Lake 프레임워크 사용

Amazon S3 데이터 소스에서 Delta Lake 프레임워크 사용

1. 소스 메뉴에서 Amazon S3를 선택합니다.
2. Amazon S3 소스 유형으로 데이터 카탈로그 테이블을 선택한 경우 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식을 Delta Lake 및 Amazon S3 URL로 표시합니다.

4. 키-값 페어를 입력하려면 추가 옵션을 선택합니다. 예를 들어 키-값 페어는 키: timestampAsOf와 값: 2023-02-24 14:16:18일 수 있습니다.



5. Amazon S3 소스 유형으로 Amazon S3 위치를 선택한 경우 Amazon S3 찾아보기를 클릭하여 Amazon S3 URL을 선택합니다.
6. 데이터 형식에서 델타 레이크를 선택합니다.

> **Note**
>
> AWS Glue Studio가 선택한 Amazon S3 폴더 또는 파일에서 스키마를 유추할 수 없는 경우 추가 옵션을 선택하여 새 폴더 또는 파일을 선택합니다.
> 추가 옵션의 스키마 추론 아래에서 다음 옵션을 선택합니다.
>
> - AWS Glue Studio에서 통해 샘플 파일을 자동으로 선택하도록 하겠습니다. AWS Glue Studio는 스키마를 추론할 수 있도록 Amazon S3 위치에서 샘플 파일을 선택합니다. 자동 샘플링된 파일 필드에서 자동으로 선택된 파일을 볼 수 있습니다.
> - Amazon S3에서 샘플 파일을 선택합니다. Amazon S3 찾아보기를 클릭하여 사용할 Amazon S3 파일을 선택합니다.

7. 스키마 추론을 클릭합니다. 그런 다음 출력 스키마 탭을 클릭하여 출력 스키마를 볼 수 있습니다.

데이터 카탈로그 데이터 소스에서 Delta Lake 프레임워크 사용

1. 소스 메뉴에서 AWS Glue Studio 데이터 카탈로그를 선택합니다.
2. 데이터 소스 속성 탭에서 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식 유형을 Delta Lake 및 Amazon S3 URL로 표시합니다.

> **Note**
>
> Delta Lake 소스가 AWS Glue 데이터 카탈로그 테이블로 등록되지 않은 경우 다음 두 가지 옵션을 사용할 수 있습니다.
>
> 1. Delta Lake 데이터 스토어에 대한 AWS Glue 크롤러를 생성합니다. 자세한 내용은 Delta Lake 데이터 스토어에 대한 구성 옵션을 지정하는 방법을 참조하세요.
> 2. Amazon S3 데이터 소스를 사용하여 Delta Lake 데이터 소스를 선택합니다. Amazon S3 데이터 소스에서 Delta Lake 프레임워크 사용 섹션을 참조하세요.

데이터 대상에서 Delta Lake 형식 사용

데이터 카탈로그 데이터 대상에서 Delta Lake 형식 사용

1. 대상 메뉴에서 AWS Glue Studio 데이터 카탈로그를 선택합니다.
2. 데이터 소스 속성 탭에서 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식 유형을 Delta Lake 및 Amazon S3 URL로 표시합니다.

Amazon S3 데이터 소스에서 Delta Lake 형식 사용

값을 입력하거나 사용 가능한 옵션 중에서 선택하여 Delta Lake 형식을 구성합니다.

- 압축 유형 - 압축 유형 옵션(비압축 또는 Snappy) 중 하나를 선택합니다.
- Amazon S3 대상 위치 - S3 찾아보기를 클릭하여 Amazon S3 대상 위치를 선택합니다.
- 데이터 카탈로그 업데이트 옵션 - Glue Studio 시각적 편집기에서 이 형식에 대한 데이터 카탈로그 업데이트는 지원되지 않습니다.
  - [데이터 카탈로그 업데이트 안 함(Do not update the Data Catalog)]: (기본값) 스키마가 변경되거나 새 파티션이 추가된 경우에도 작업에서 데이터 카탈로그를 업데이트하지 않으려면 이 옵션을 선택합니다.
  - AWS Glue 작업 실행 후 데이터 카탈로그를 업데이트하려면 AWS Glue 크롤러를 실행하거나 일정을 예약합니다. 자세한 내용은 Delta Lake 데이터 스토어에 대한 구성 옵션을 지정하는 방법을 참조하세요.
- 파티션 키 - 출력에서 파티션 키로 사용할 열을 선택합니다. 파티션 키를 더 추가하려면 [파티션 키 추가(Add a partition key)]를 선택합니다.

- 선택 사항으로 추가 옵션을 선택하여 키-값 페어를 입력합니다. 예를 들어 키-값 페어는 키: timestampAsOf와 값: 2023-02-24 14:16:18일 수 있습니다.

AWS Glue Studio에서 Apache Iceberg 프레임워크 사용

데이터 대상에서 Apache Iceberg 프레임워크 사용

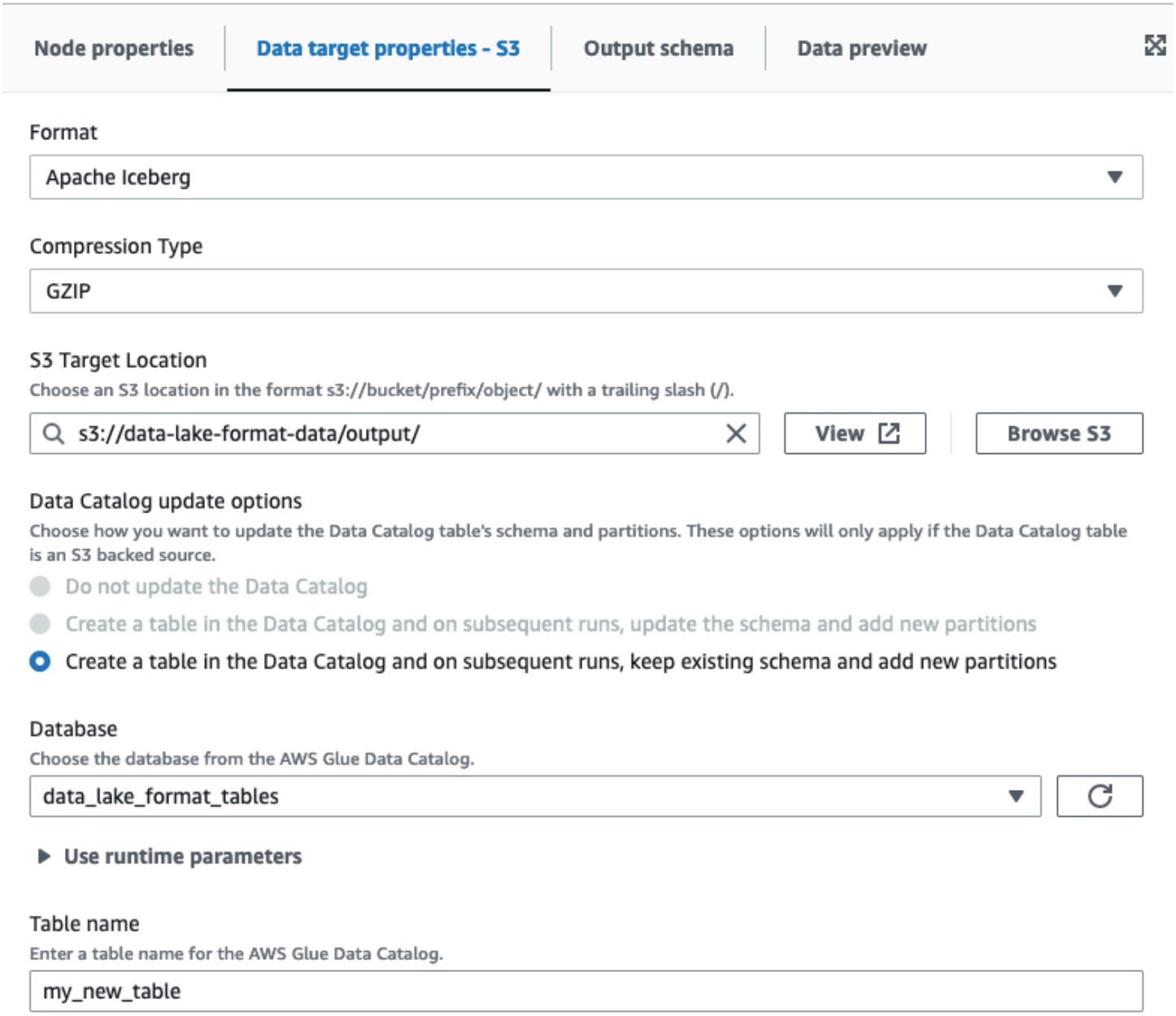
데이터 카탈로그 데이터 대상에서 Apache Iceberg 프레임워크 사용

1. 대상 메뉴에서 AWS Glue Studio 데이터 카탈로그를 선택합니다.
2. 데이터 소스 속성 탭에서 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식 유형을 Apache Iceberg 및 Amazon S3 URL로 표시합니다.

Amazon S3 데이터 대상에서 Apache Iceberg 프레임워크 사용

값을 입력하거나 사용 가능한 옵션 중에서 선택하여 Apache Iceberg 형식을 구성합니다.

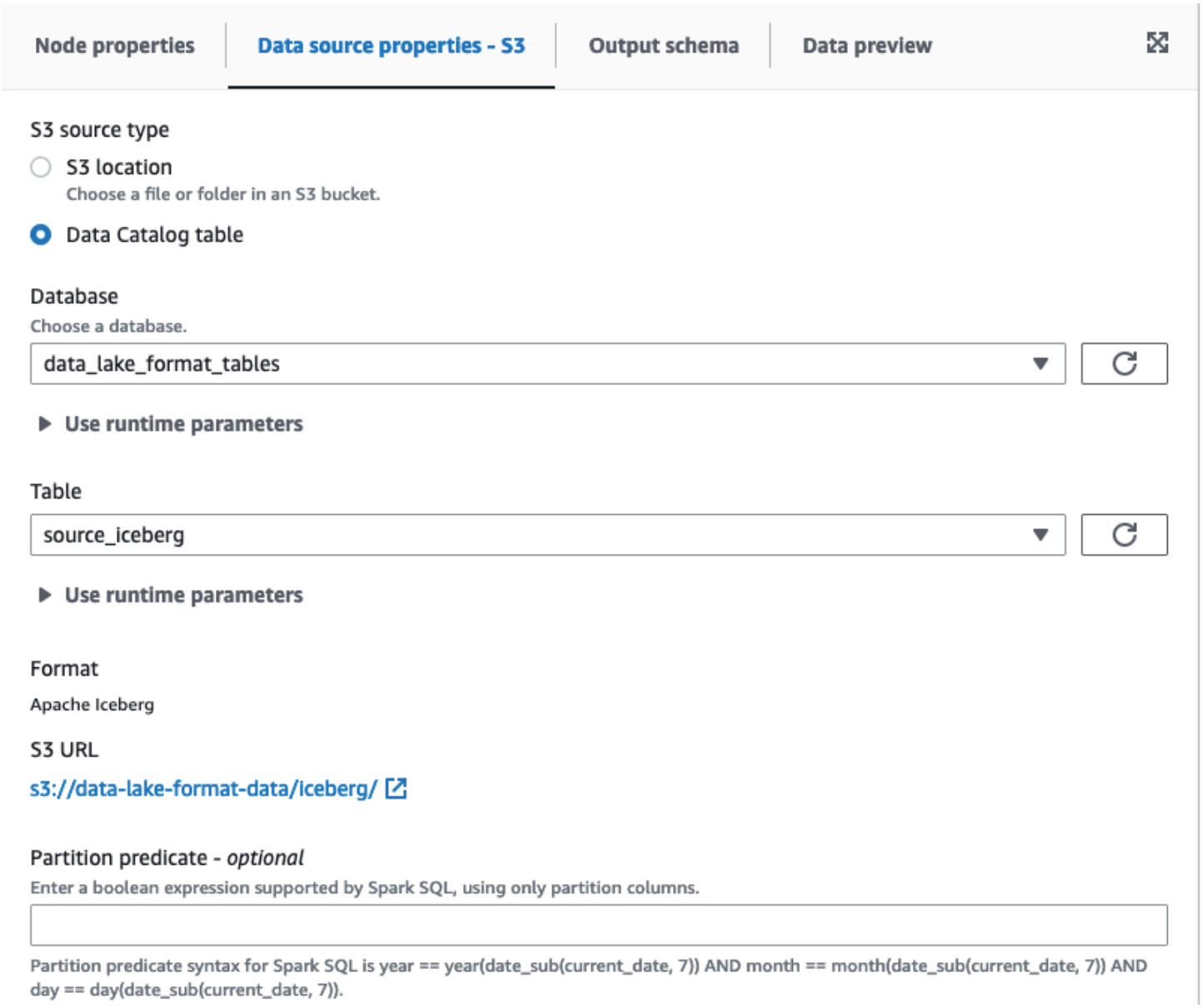
- 형식 - 드롭다운 메뉴에서 Apache Iceberg를 선택합니다.
- Amazon S3 대상 위치 - S3 찾아보기를 클릭하여 Amazon S3 대상 위치를 선택합니다.
- 데이터 카탈로그 업데이트 옵션 - 계속 진행하려면 데이터 카탈로그에 테이블 생성, 후속 실행 시 기존 스키마 유지 및 새 파티션 추가를 선택해야 합니다. AWS Glue를 사용하여 새 Iceberg 테이블을 작성하려면 Data Catalog를 Iceberg 테이블의 카탈로그로 구성해야 합니다. Data Catalog에 등록된 기존 Iceberg 테이블을 업데이트하려면 Data Catalog를 대상으로 선택합니다.
  - 데이터베이스 - Data Catalog에서 데이터베이스를 선택합니다.
  - 테이블 이름 – 사용자의 테이블 이름에 대한 값을 입력합니다. Apache Iceberg 테이블 이름은 모두 소문자여야 합니다. 공백은 허용되지 않으므로 필요한 경우 밑줄을 사용합니다. 예를 들어 'data_lake_format_tables'와 같습니다.



Amazon S3 데이터 소스에서 Apache Iceberg 프레임워크 사용

데이터 카탈로그 데이터 소스에서 Apache Iceberg 프레임워크 사용

1. 소스 메뉴에서 AWS Glue Studio 데이터 카탈로그를 선택합니다.
2. 데이터 소스 속성 탭에서 데이터베이스와 테이블을 선택합니다.
3. AWS Glue Studio는 형식 유형을 Apache Iceberg 및 Amazon S3 URL로 표시합니다.



Amazon S3 데이터 소스에서 Apache Iceberg 프레임워크 사용

Apache Iceberg는 AWS Glue Studio에서 Amazon S3 소스 노드의 데이터 옵션으로 사용할 수 없습니다.

시각적 ETL 작업을 사용하여 데이터 소스에 연결

새 작업을 생성하는 동안 AWS Glue에서 시각적 ETL 작업을 편집할 때 연결을 사용하여 데이터에 연결할 수 있습니다. 커넥터를 사용하여 데이터를 읽는 소스 노드와 데이터 쓰기 위치를 지정하는 대상 노드를 추가하여 이 작업을 수행할 수 있습니다.

주제

- 데이터 소스 노드의 속성 수정하기
- 데이터 원본에 데이터 카탈로그 테이블 사용
- 데이터 원본에 커넥터 사용
- 데이터 원본에 Amazon S3의 파일 사용
- 스트리밍 데이터 원본 사용
- 참조

데이터 소스 노드의 속성 수정하기

데이터 원본 속성을 지정하려면 먼저 작업 다이어그램에서 데이터 원본 노드를 선택합니다. 그런 다음 노드 세부 정보 패널의 오른쪽에서 노드 속성을 구성합니다.

데이터 원본 노드의 속성을 수정하려면

1. 새 작업 또는 저장된 작업의 시각적 편집기로 이동합니다.
2. 작업 다이어그램에서 데이터 원본 노드를 선택합니다.
3. 노드 세부 정보 패널에서 [노드 속성(Node properties)] 탭을 선택하고 다음 정보를 입력합니다.
   - [이름(Name)]: (선택 사항) 작업 다이어그램의 노드와 연결할 이름을 입력합니다. 이 이름은 이 작업에 대한 모든 노드에서 고유해야 합니다.
   - [노드 유형(Node type)]: 노드 유형에 따라 노드에서 수행되는 작업이 결정됩니다. [노드 유형(Node type)]에 대한 옵션 목록에서 [데이터 원본(Data source)] 제목 아래에 나열된 값 중 하나를 선택합니다.
4. [데이터 원본 속성(Data source properties)] 정보를 구성합니다. 자세한 내용은 다음 단원을 참조하세요.
   - 데이터 원본에 데이터 카탈로그 테이블 사용
   - 데이터 원본에 커넥터 사용
   - 데이터 원본에 Amazon S3의 파일 사용
   - 스트리밍 데이터 원본 사용
5. (선택 사항) 노드 속성과 데이터 원본 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터 원본에 대한 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.

6. (선택 사항) 노드 속성과 데이터 원본 속성을 구성한 후 노드 세부 정보 패널에서 [데이터 미리 보기(Data preview)] 탭을 선택하여 데이터 원본에서 데이터 집합을 미리 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. 이 기능 사용과 관련된 비용이 있으며 IAM 역할을 제공하는 즉시 결제가 시작됩니다.

데이터 원본에 데이터 카탈로그 테이블 사용

Amazon S3와 커넥터를 제외한 모든 데이터 원본의 경우 선택한 원본 유형의 테이블이 AWS Glue Data Catalog에 있어야 합니다. AWS Glue는 데이터 카탈로그 테이블을 생성하지 않습니다.

데이터 카탈로그 테이블을 기반으로 데이터 원본 노드를 구성하려면

1. 새 작업 또는 저장된 작업의 시각적 편집기로 이동합니다.
2. 작업 다이어그램에서 데이터 원본 노드를 선택합니다.
3. [데이터 원본 속성(Data source properties)] 탭을 선택한 후 다음 정보를 입력합니다.

   - [S3 소스 유형(S3 source type)]: (Amazon S3 데이터 원본만 해당) 기존 AWS Glue Data Catalog 테이블을 사용하려면 [카탈로그 테이블 선택(Select a Catalog table)] 옵션을 선택합니다.
   - [데이터베이스(Database)]: 데이터 카탈로그에서 이 작업에 사용할 원본 테이블이 포함된 데이터베이스를 선택합니다. 검색 필드를 사용하여 이름으로 데이터베이스를 검색할 수 있습니다.
   - [테이블(Table)]: 목록에서 원본 데이터와 연결된 테이블을 선택합니다. 이 테이블이 AWS Glue Data Catalog에 이미 존재해야 합니다. 검색 필드를 사용하여 이름으로 테이블을 검색할 수 있습니다.
   - 파티션 조건자: (Amazon S3 데이터 원본만 해당) 분할 열만 포함하는 Spark SQL 기반 부울 표현식을 입력합니다. 예: `"(year=='2020' and month=='04')"`
   - [임시 디렉터리(Temporary directory)]: (Amazon Redshift 데이터 원본만 해당) ETL 작업에서 임시 중간 결과를 작성할 수 있는 Amazon S3 작업 디렉터리 위치에 대한 경로를 입력합니다.
   - [클러스터와 연결된 역할(Role associated with the cluster)]: (Amazon Redshift 데이터 원본만 해당) Amazon Redshift 클러스터에 대한 권한이 포함된 ETL 작업에 사용할 역할을 입력합니다. 자세한 내용은 the section called “데이터 원본 및 데이터 대상 권한” 단원을 참조하십시오.

데이터 원본에 커넥터 사용

[노드 유형(Node type)]에 대한 커넥터를 선택하는 경우 사용자 정의 커넥터로 작업 작성의 지침에 따라 데이터 원본 속성 구성을 완료합니다.

데이터 원본에 Amazon S3의 파일 사용

Amazon S3를 데이터 원본으로 선택한 경우 다음 중 하나를 선택할 수 있습니다.

- 데이터 카탈로그 데이터베이스 및 테이블.
- Amazon S3의 버킷, 폴더 또는 파일.

Amazon S3 버킷을 데이터 원본으로 사용하는 경우 AWS Glue는 파일 중 하나에서 또는 샘플 파일로 지정한 파일을 사용하여 지정된 위치에 있는 데이터의 스키마를 탐지합니다. [스키마 추론(Infer schema)] 버튼을 사용할 때 스키마 감지가 발생합니다. Amazon S3 위치 또는 샘플 파일을 변경하는 경우 [스키마 추론(Infer schema)]을 다시 선택하여 새 정보로 스키마 감지를 수행해야 합니다.

Amazon S3의 파일에서 직접 읽는 데이터 원본 노드를 구성하려면

1. 새 작업 또는 저장된 작업의 시각적 편집기로 이동합니다.
2. Amazon S3 소스에 대한 작업 다이어그램에서 데이터 원본 노드를 선택합니다.
3. [데이터 원본 속성(Data source properties)] 탭을 선택한 후 다음 정보를 입력합니다.

   - [S3 소스 유형(S3 source type)]: (Amazon S3 데이터 원본만 해당) [S3 위치(S3 location)] 옵션을 선택합니다.
   - [S3 URL]: 작업에 대한 데이터가 포함된 Amazon S3 버킷, 폴더 또는 파일의 경로를 입력합니다. [S3 찾아보기(Browse S3)]를 선택하여 계정에서 사용할 수 있는 위치에서 경로를 선택할 수 있습니다.
   - 재귀(Recursive): AWS Glue가 S3 위치에 있는 하위 폴더의 파일에서 데이터를 읽도록 하려면 이 옵션을 선택합니다.

     하위 폴더에 분할된 데이터가 포함된 경우 AWS Glue는 폴더 이름에 지정된 파티션 정보를 데이터 카탈로그에 추가하지 않습니다. 예를 들어 Amazon S3의 다음 폴더를 고려합니다.

     ```
     S3://sales/year=2019/month=Jan/day=1
     S3://sales/year=2019/month=Jan/day=2
     ```

     재귀(Recursive)를 선택하고 `sales` 폴더를 S3 위치로 선택하면 AWS Glue는 모든 하위 폴더의 데이터를 읽지만 연도, 월 또는 일의 파티션을 생성하지 않습니다.
   - [데이터 포맷(Data format)]: 데이터가 저장되는 포맷을 선택합니다. JSON, CSV 또는 Parquet를 선택할 수 있습니다. 선택한 값은 소스 파일에서 데이터를 읽는 방법을 AWS Glue 작업에 알려줍니다.

> **Note**
>
> 올바른 데이터 포맷을 선택하지 않을 경우 AWS Glue는 스키마를 올바르게 추론할 수 있지만 작업에서 소스 파일의 데이터를 올바르게 구문 분석할 수 없습니다.

선택한 포맷에 따라 추가 구성 옵션을 입력할 수 있습니다.

- [JSON](JavaScript Object Notation)
  - [JsonPath]: 테이블 스키마 정의에 사용하는 객체를 가리키는 JSON 경로입니다. JSON 경로 표현식은 XPath 표현식이 XML 문서와 함께 사용되는 것과 같은 방식으로 항상 JSON 구조를 참조합니다. JSON 경로의 "루트 멤버 객체"는 객체 또는 배열인 경우에도 항상 `$`라고 합니다. JSON 경로 작성 시 점이나 대괄호를 사용할 수 없습니다.

    JSON 경로에 대한 자세한 내용은 GitHub 웹 사이트의 JsonPath를 참조하세요.
  - [소스 파일의 레코드가 여러 줄에 걸쳐 있을 수 있음(Records in source files can span multiple lines)]: 단일 레코드가 CSV 파일의 여러 줄에 걸쳐 있을 수 있는 경우 이 옵션을 선택합니다.
- [CSV](쉼표로 분리된 값)
  - 구분 기호: 행의 각 열 항목을 구분하는 문자(예: `;` 또는 `,`)를 입력합니다.
  - [이스케이프 문자(Escape character)]: 이스케이프 문자로 사용되는 문자를 입력합니다. 이 문자는 이스케이프 문자 바로 뒤에 오는 문자를 문자 그대로 사용해야 하며 구분 기호로 해석되어서는 안 됨을 나타냅니다.
  - [인용 문자(Quote character)]: 개별 문자열을 단일 값으로 그룹화하는 데 사용되는 문자를 입력합니다. 예를 들어 CSV 파일에 `"This is a single value"`와 같은 값이 있는 경우 [큰따옴표(")(Double quote (""))]를 선택합니다.
  - [소스 파일의 레코드가 여러 줄에 걸쳐 있을 수 있음(Records in source files can span multiple lines)]: 단일 레코드가 CSV 파일의 여러 줄에 걸쳐 있을 수 있는 경우 이 옵션을 선택합니다.
  - [소스 파일의 첫 번째 줄에 열 머리글 포함(First line of source file contains column headers)]: CSV 파일의 첫 번째 행에 데이터 대신 열 머리글이 포함된 경우 이 옵션을 선택합니다.
- [Parquet](Apache Parquet 컬럼 방식 스토리지)

  Parquet 포맷으로 저장된 데이터에 대해 구성할 추가 설정이 없습니다.

- Apache Hudi

  Apache Hudi 포맷으로 저장된 데이터에 대해 구성할 추가 설정이 없습니다.

- Delta Lake

  Delta Lake 포맷으로 저장된 데이터에 대해 구성할 추가 설정이 없습니다.

- Excel

  Excel 포맷으로 저장된 데이터에 대해 구성할 추가 설정이 없습니다.

- [분할 조건자(Partition predicate)]: 데이터 원본에서 읽은 데이터를 분할하려면 분할 열만 포함하는 Spark SQL 기반의 부울 식을 입력합니다. 예: `"(year=='2020' and month=='04')"`
- 고급 옵션(Advanced options): AWS Glue가 특정 파일을 기준으로 데이터 스키마를 탐지하도록 하려면 이 섹션을 확장합니다.
  - 스키마 추론(Schema inference): AWS Glue가 파일을 선택하도록 하는 대신 특정 파일을 사용하려면 S3에서 샘플 파일 선택(Choose a sample file from S3) 옵션을 선택합니다. Excel 소스에는 스키마 추론을 사용할 수 없습니다.
  - [자동 샘플링된 파일(Auto-sampled file)]: 스키마를 추론하는 데 사용할 Amazon S3의 파일 경로를 입력합니다.

  데이터 원본 노드를 편집하고 선택한 샘플 파일을 변경하는 경우 [스키마 다시 로드(Reload schema)]를 사용하여 새 샘플 파일로 스키마를 감지합니다.

4. [스키마 추론(Infer schema)] 버튼을 선택하여 Amazon S3의 소스 파일에서 스키마를 감지합니다. Amazon S3 위치 또는 샘플 파일을 변경하는 경우 [스키마 추론(Infer schema)]을 다시 선택하여 새 정보로 스키마를 추론해야 합니다.

스트리밍 데이터 원본 사용

지속적으로 실행되고 Amazon Kinesis Data Streams, Apache Kafka 및 Amazon Managed Streaming for Apache Kafka(Amazon MSK)의 스트리밍 소스의 데이터 사용하는 스트리밍 추출, 변환, 로드 작업을 생성할 수 있습니다.

스트리밍 데이터 원본의 속성을 구성하려면

1. 새 작업 또는 저장된 작업의 시각적 그래프 편집기로 이동합니다.
2. Kafka 또는 Kinesis Data Streams에 대한 그래프에서 데이터 원본 노드를 선택합니다.
3. [데이터 원본 속성(Data source properties)] 탭을 선택한 후 다음 정보를 입력합니다.

Kinesis

- Kinesis 소스 유형(Kinesis source type): 스트림 세부 정보(Stream details) 옵션을 선택하여 스트리밍 소스에 직접 액세스하거나 데이터 카탈로그 테이블(Data Catalog table)을 선택하여 테이블에 저장된 정보를 대신 사용합니다.

  스트림 세부 정보(Stream details)를 선택한 경우 다음 추가 정보를 지정합니다.

  - 데이터 스트림 위치: 스트림이 현재 사용자와 연결되어 있는지 또는 다른 사용자와 연결되어 있는지 여부를 선택합니다.
  - 리전(Region): 스트림이 있는 AWS 리전을 선택합니다. 이 정보는 데이터 스트림에 액세스하기 위한 ARN을 구성하는 데 사용됩니다.
  - 스트림 ARN(Stream ARN): Kinesis 데이터 스트림의 Amazon 리소스 이름(ARN)을 입력합니다. 스트림이 현재 계정 내에 있는 경우 드롭다운 목록에서 스트림 이름을 선택할 수 있습니다. 검색 필드를 사용하여 이름이나 ARN으로 데이터 스트림을 검색할 수 있습니다.
  - 데이터 포맷(Data format): 목록에서 데이터 스트림에 사용할 포맷을 선택합니다.

    AWS Glue는 스트리밍 데이터에서 스키마를 자동으로 탐지합니다.

  데이터 카탈로그 테이블(Data Catalog table)을 선택한 경우 다음 추가 정보를 지정합니다.

  - 데이터베이스(Database): (선택 사항) AWS Glue 데이터 카탈로그에서 스트리밍 데이터 원본과 연결된 테이블이 포함된 데이터베이스를 선택합니다. 검색 필드를 사용하여 이름으로 데이터베이스를 검색할 수 있습니다.
  - [테이블(Table)]: (선택 사항) 목록에서 원본 데이터와 연결된 테이블을 선택합니다. 이 테이블이 AWS Glue 데이터 카탈로그에 이미 존재해야 합니다. 검색 필드를 사용하여 이름으로 테이블을 검색할 수 있습니다.
  - 스키마 탐지(Detect schema): AWS Glue가 데이터 카탈로그 테이블의 스키마 정보를 사용하는 대신 스트리밍 데이터에서 스키마를 탐지하도록 하려면 이 옵션을 선택합니다. 스트림 세부 정보(Stream details) 옵션을 선택한 경우 이 옵션은 자동으로 사용됩니다.
- 시작 위치(Starting position): 기본값으로 ETL 작업은 가장 오래된 항목(Earliest) 옵션을 사용하므로 스트림에서 사용 가능한 가장 오래된 레코드부터 데이터를 읽습니다. 최신 항목(Latest)을 대신 선택할 수도 있습니다. 이 경우 ETL 작업은 스트림에서 가장 최근 레코드 직후부터 읽기를 시작해야 합니다.
- [기간 크기(Window size)]: 기본적으로 ETL 작업은 100초 기간에 데이터를 처리하고 작성합니다. 이를 통해 데이터를 효율적으로 처리할 수 있으며 예상보다 늦게 도착하는 데이터에

대해 집계를 수행할 수 있습니다. 이 기간 크기를 수정하여 적시성 또는 집계 정확도를 높일 수 있습니다.

AWS Glue 스트리밍 작업은 작업 북마크 대신 체크포인트를 사용하여 읽은 데이터를 추적합니다.

- 연결 옵션(Connection options): 키-값 페어를 추가하여 추가 연결 옵션을 지정하려면 이 섹션을 확장합니다. 여기에서 지정할 수 있는 옵션에 대한 자세한 내용은 AWS Glue 개발자 가이드의 "connectionType": "kinesis"를 참조하세요.

Kafka

- Apache Kafka 소스(Apache Kafka source): 스트림 세부 정보(Stream details) 옵션을 선택하여 스트리밍 소스에 직접 액세스하거나 데이터 카탈로그 테이블(Data Catalog table)을 선택하여 테이블에 저장된 정보를 대신 사용합니다.

  데이터 카탈로그 테이블(Data Catalog table)을 선택한 경우 다음 추가 정보를 지정합니다.

  - 데이터베이스(Database): (선택 사항) AWS Glue 데이터 카탈로그에서 스트리밍 데이터 원본과 연결된 테이블이 포함된 데이터베이스를 선택합니다. 검색 필드를 사용하여 이름으로 데이터베이스를 검색할 수 있습니다.
  - [테이블(Table)]: (선택 사항) 목록에서 원본 데이터와 연결된 테이블을 선택합니다. 이 테이블이 AWS Glue 데이터 카탈로그에 이미 존재해야 합니다. 검색 필드를 사용하여 이름으로 테이블을 검색할 수 있습니다.
  - 스키마 탐지(Detect schema): AWS Glue가 데이터 카탈로그 테이블에 스키마 정보를 저장하는 대신 스트리밍 데이터에서 스키마를 탐지하도록 하려면 이 옵션을 선택합니다. 스트림 세부 정보(Stream details) 옵션을 선택한 경우 이 옵션은 자동으로 사용됩니다.

  스트림 세부 정보(Stream details)를 선택한 경우 다음 추가 정보를 지정합니다.

  - 연결 이름(Connection name): Kafka 데이터 스트림에 대한 액세스 및 인증 정보가 포함된 AWS Glue 연결을 선택합니다. Kafka 스트리밍 데이터 원본과의 연결을 사용해야 합니다. 연결이 없는 경우 AWS Glue 콘솔을 사용하여 Kafka 데이터 스트림에 대한 연결을 생성할 수 있습니다.
  - 주제 이름(Topic name): 읽을 주제의 이름을 입력합니다.
  - 데이터 포맷(Data format): Kafka 이벤트 스트림에서 데이터를 읽을 때 사용할 포맷을 선택합니다.
- 시작 위치(Starting position): 기본값으로 ETL 작업은 가장 오래된 항목(Earliest) 옵션을 사용하므로 스트림에서 사용 가능한 가장 오래된 레코드부터 데이터를 읽습니다. 최신 항목

(Latest)을 대신 선택할 수도 있습니다. 이 경우 ETL 작업은 스트림에서 가장 최근 레코드 직후부터 읽기를 시작해야 합니다.

- [기간 크기(Window size)]: 기본적으로 ETL 작업은 100초 기간에 데이터를 처리하고 작성합니다. 이를 통해 데이터를 효율적으로 처리할 수 있으며 예상보다 늦게 도착하는 데이터에 대해 집계를 수행할 수 있습니다. 이 기간 크기를 수정하여 적시성 또는 집계 정확도를 높일 수 있습니다.

  AWS Glue 스트리밍 작업은 작업 북마크 대신 체크포인트를 사용하여 읽은 데이터를 추적합니다.
- 연결 옵션(Connection options): 키-값 페어를 추가하여 추가 연결 옵션을 지정하려면 이 섹션을 확장합니다. 여기에서 지정할 수 있는 옵션에 대한 자세한 내용은 AWS Glue 개발자 가이드의 "connectionType": "kafka"를 참조하세요.

> **Note**
>
> 데이터 미리 보기는 현재 스트리밍 데이터 원본에 대해 지원되지 않습니다.

참조

모범 사례

- Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue

ETL 프로그래밍

- AWS Glue에서 ETL에 대한 관련 연결 유형 및 옵션
- JDBC connectionType 값
- Amazon Redshift 간 데이터 이동을 위한 고급 옵션

데이터 대상 노드 구성

데이터 대상은 작업이 변환된 데이터를 쓰는 위치입니다.

데이터 대상 옵션 개요

다음이 데이터 대상(데이터 싱크라고도 함)일 수 있습니다.

- [S3] - 작업이 선택한 Amazon S3 위치와 지정한 포맷의 파일에 데이터를 씁니다.

  데이터 대상에 대한 파티션 열을 구성하면 작업이 파티션 키를 기반으로 디렉터리에 Amazon S3에 대한 데이터 집합을 씁니다.
- AWS Glue Data Catalog - 작업에서 데이터 카탈로그의 테이블과 연결된 정보를 사용하여 출력 데이터를 대상 위치에 씁니다.

  수동으로 또는 크롤러를 사용하여 테이블을 생성할 수 있습니다. AWS CloudFormation 템플릿을 사용하여 데이터 카탈로그에서 테이블을 생성할 수도 있습니다.
- 커넥터 - 커넥터는 데이터 스토어와 AWS Glue 간의 통신을 용이하게 하는 코드입니다. 작업은 커넥터 및 연결된 연결을 사용하여 출력 데이터를 대상 위치에 씁니다. AWS Marketplace에서 제공되는 커넥터를 구독하거나 사용자 정의 커넥터를 생성할 수 있습니다. 자세한 내용은 AWS Glue Studio에 커넥터 추가 섹션을 참조하세요.

작업이 Amazon S3 데이터 대상에 쓸 때 데이터 카탈로그를 업데이트하도록 선택할 수 있습니다. 스키마 또는 파티션이 변경될 때 크롤러가 데이터 카탈로그를 업데이트하도록 요구하는 대신 이 옵션을 사용하면 테이블을 쉽게 최신 상태로 유지할 수 있습니다. 이 옵션은 필요에 따라 데이터 카탈로그에 새 테이블을 추가하고 테이블 파티션을 업데이트하며 작업에서 직접 테이블의 스키마를 업데이트하여 분석에 데이터를 사용할 수 있도록 하는 프로세스를 단순화합니다.

데이터 대상 노드 편집

데이터 대상은 작업이 변환된 데이터를 쓰는 위치입니다.

작업 다이어그램에서 데이터 대상 노드를 추가하거나 구성하려면

1. (선택 사항) 대상 노드를 추가해야 하는 경우 시각적 편집기 상단의 도구 모음에서 Target(대상)을 선택한 다음 S3 또는 Glue Data Catalog(Glue 데이터 카탈로그)를 선택합니다.

   - 대상으로 [S3]를 선택하면 작업은 지정한 Amazon S3 위치에 있는 하나 이상의 파일에 데이터 집합을 씁니다.
   - 대상으로 [AWS Glue Data Catalog]를 선택하면 작업은 데이터 카탈로그에서 선택한 테이블에 설명된 위치에 씁니다.

2. 작업 다이어그램에서 데이터 대상 노드를 선택합니다. 노드를 선택하면 페이지 오른쪽에 노드 세부 정보 패널이 나타납니다.
3. [노드 속성(Node properties)] 탭을 선택한 후 다음 정보를 입력합니다.

   - [이름(Name)]: 작업 다이어그램의 노드와 연결할 이름을 입력합니다.
   - [노드 유형(Node type)]: 값이 이미 선택되어 있어야 하지만 필요에 따라 변경할 수 있습니다.
   - [상위 노드(Node parents)]: 상위 노드는 대상 위치에 쓰려는 출력 데이터를 제공하는 작업 다이어그램의 노드입니다. 미리 채워진 작업 다이어그램의 경우 대상 노드에 이미 상위 노드가 선택되어 있어야 합니다. 표시되는 상위 노드가 없으면 목록에서 상위 노드를 선택합니다.

     대상 노드에는 단일 상위 노드가 있습니다.
4. [데이터 대상 속성(Data target properties)] 정보를 구성합니다. 자세한 내용은 다음 단원을 참조하세요.

   - 데이터 대상에 Amazon S3 사용
   - 데이터 대상에 데이터 카탈로그 테이블 사용
   - 데이터 대상에 커넥터 사용
5. (선택 사항) 데이터 대상 노드 속성을 구성한 후 노드 세부 정보 패널에서 [출력 스키마(Output schema)] 탭을 선택하여 데이터에 대해 출력 스키마를 볼 수 있습니다. 작업의 노드에 대해 이 탭을 처음 선택하면 데이터 액세스를 위해 IAM 역할을 제공하라는 메시지가 나타납니다. [작업 세부 정보(Job details)] 탭에서 IAM 역할을 지정하지 않은 경우 여기에 IAM 역할을 입력하라는 메시지가 나타납니다.

데이터 대상에 Amazon S3 사용

Amazon S3와 커넥터를 제외한 모든 데이터 원본의 경우 선택한 원본 유형의 테이블이 AWS Glue Data Catalog에 있어야 합니다. AWS Glue Studio는 데이터 카탈로그 테이블을 생성하지 않습니다.

Amazon S3에 쓰는 데이터 대상 노드를 구성하려면

1. 새 작업 또는 저장된 작업의 시각적 편집기로 이동합니다.
2. 작업 다이어그램에서 데이터 원본 노드를 선택합니다.
3. [데이터 원본 속성(Data source properties)] 탭을 선택한 후 다음 정보를 입력합니다.

   - [포맷(Format)]: 목록에서 포맷을 선택합니다. 데이터 결과에 사용할 수 있는 포맷 유형은 다음과 같습니다.
     - [JSON]: JavaScript Object Notation.

- [CSV]: 쉼표로 분리된 값.
- [Avro]: Apache Avro JSON 바이너리입니다.
- Parquet: 데이터 포맷으로 `DynamicFrames`에 최적화된 사용자 정의 Parquet 라이터 유형입니다. 데이터에 대해 미리 계산된 스키마를 요구하는 대신 스키마를 동적으로 계산하고 수정합니다.
- [ORC]: Apache Optimized Row Columnar(ORC) 포맷입니다.
- Apache Hudi: 증분 데이터 처리 및 데이터 파이프라인 개발을 간소화하는 오픈 소스 데이터 레이크 스토리지 프레임워크입니다.
- Apache Iceberg: SQL 테이블처럼 작동하는 고성능 테이블 형식입니다.
- Delta Lake: ACID 트랜잭션을 수행하고, 메타데이터 처리를 확장하고, 스트리밍 및 배치 데이터 처리를 통합하는 데 도움이 되는 오픈 소스 데이터 레이크 스토리지 프레임워크입니다.
- XML: 확장형 마크업 언어(XML)입니다.
- Tableau Hyper: Tableau의 메모리 내 데이터 엔진 기술입니다.

이러한 포맷 옵션에 대한 자세한 내용은 AWS Glue Developer Guide의 Format Options for ETL Inputs and Outputs in AWS Glue를 참조하세요.

- 압축 유형: `CSV`, `JSON`, `Parquet` 파일 유형을 사용하여 데이터를 선택적으로 압축하도록 선택할 수 있습니다. 기본값은 압축 안 함 또는 [없음(None)]입니다.

파일 형식	압축
JSON/CSV/XML	GZIP, BZIP2, BROTLI, DEFLATE, LZ4, Snappy
PARQUET	Snappy, LZ4, LZO, BROTLI, GZIP
ORC	Snappy, ZLIB, 비압축, LZO
Avro	GZIP, BZIP2, BROTLI, DEFLATE, LZ4, Snappy
Delta Lake	GZIP, BZIP2, BROTLI, DEFLATE, LZ4, Snappy
Apache Hudi	GZIP, LZO, Snappy

파일 형식	압축
Apache Iceberg	GZIP, LZO, Snappy
Tableau 하이퍼	없음

- [S3 대상 위치(S3 Target Location)]: 데이터 출력을 위한 Amazon S3 버킷 및 위치입니다. [S3 찾아보기(Browse S3)] 버튼을 선택하여 액세스 권한이 있는 Amazon S3 버킷을 확인하고 그 중 하나를 대상으로 선택할 수 있습니다.
- 데이터 카탈로그 업데이트 옵션
  - [데이터 카탈로그 업데이트 안 함(Do not update the Data Catalog)]: (기본값) 스키마가 변경되거나 새 파티션이 추가된 경우에도 작업에서 데이터 카탈로그를 업데이트하지 않으려면 이 옵션을 선택합니다.
  - [데이터 카탈로그에 테이블 생성, 후속 실행 시 스키마 업데이트 및 새 파티션 추가(Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions)]: 이 옵션을 선택하면 작업이 처음 실행될 때 데이터 카탈로그에 테이블이 생성됩니다. 후속 작업 실행 시 스키마가 변경되거나 새 파티션이 추가되면 작업이 데이터 카탈로그 테이블을 업데이트합니다.

    또한 데이터 카탈로그에서 데이터베이스를 선택하고 테이블 이름을 입력해야 합니다.
  - [데이터 카탈로그에 테이블 생성, 기존 스키마 유지 및 새 파티션 추가(Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions)]: 이 옵션을 선택하면 작업이 처음 실행될 때 데이터 카탈로그에 테이블이 생성됩니다. 후속 작업 실행 시 작업은 새 파티션을 추가하기 위해서만 데이터 카탈로그 테이블을 업데이트합니다.

    또한 데이터 카탈로그에서 데이터베이스를 선택하고 테이블 이름을 입력해야 합니다.
- 파일 파티셔닝: 출력을 저장할 파티셔닝 유형을 선택합니다.
  - 파일 자동 생성(권장): 생성된 파일 수의 기본값입니다.
  - 다중 파일 출력: 원하는 파일 출력 수를 지정합니다. 최적의 성능을 위해 자동 생성된 기본 파일 수 값을 사용합니다.
- [파티션 키(Partition keys)]: 출력에서 파티션 키로 사용할 열을 선택합니다. 파티션 키를 더 추가하려면 [파티션 키 추가(Add a partition key)]를 선택합니다.

파일 파티셔닝은 Tableau Hyper에 대상 형식으로 지원되지 않습니다.

데이터 대상에 데이터 카탈로그 테이블 사용

Amazon S3과 커넥터를 제외한 모든 데이터 원본의 경우 선택한 대상 유형의 테이블이 AWS Glue Data Catalog에 있어야 합니다. AWS Glue Studio는 데이터 카탈로그 테이블을 생성하지 않습니다.

데이터 카탈로그 테이블을 사용하는 대상에 대한 데이터 속성을 구성하려면

1. 새 작업 또는 저장된 작업의 시각적 편집기로 이동합니다.
2. 작업 다이어그램에서 데이터 대상 노드를 선택합니다.
3. [데이터 대상 속성(Data target properties)] 탭을 선택한 후 다음 정보를 입력합니다.

   - [데이터베이스(Database)]: 목록에서 대상으로 사용할 테이블이 포함된 데이터베이스를 선택합니다. 이 데이터베이스가 데이터 카탈로그에 이미 존재해야 합니다.
   - [테이블(Table)]: 목록에서 출력 데이터의 스키마를 정의하는 테이블을 선택합니다. 이 테이블이 데이터 카탈로그에 이미 존재해야 합니다.

     데이터 카탈로그의 테이블은 열 이름, 데이터 유형 정의, 파티션 정보 및 대상 데이터 집합에 대한 기타 메타데이터로 구성됩니다. 작업은 데이터 카탈로그에서 이 테이블에 설명된 위치에 씁니다.

     데이터 카탈로그에 테이블 생성에 대한 자세한 내용은 AWS Glue Developer Guide의 Defining Tables in the Data Catalog를 참조하세요.
   - 데이터 카탈로그 업데이트 옵션
     - [테이블 정의 변경 안 함(Do not change table definition)]: (기본값) 스키마가 변경되거나 새 파티션이 추가된 경우에도 작업에서 데이터 카탈로그를 업데이트하지 않으려면 이 옵션을 선택합니다.
     - [스키마 업데이트 및 새 파티션 추가(Update schema and add new partitions)]: 이 옵션을 선택하면 스키마가 변경되거나 새 파티션이 추가될 때 작업이 데이터 카탈로그 테이블을 업데이트합니다.
     - [기존 스키마 유지 및 새 파티션 추가(Keep existing schema and add new partitions)]: 이 옵션을 선택하면 작업에서 새 파티션을 추가하기 위해서만 데이터 카탈로그 테이블을 업데이트합니다.
     - [파티션 키(Partition keys)]: 출력에서 파티션 키로 사용할 열을 선택합니다. 파티션 키를 더 추가하려면 [파티션 키 추가(Add a partition key)]를 선택합니다.

데이터 대상에 커넥터 사용

[노드 유형(Node type)]에 대한 커넥터를 선택하는 경우 사용자 정의 커넥터로 작업 작성의 지침에 따라 데이터 대상 속성 구성을 완료합니다.

작업 스크립트 편집 또는 업로드

AWS Glue Studio 시각적 편집기를 사용하여 작업 스크립트를 편집하거나 고유한 스크립트를 업로드합니다.

작업이 AWS Glue Studio로 생성된 경우에만 시각적 편집기를 사용하여 작업 노드를 편집할 수 있습니다. 작업이 AWS Glue 콘솔, API 명령 또는 명령줄 인터페이스(CLI)로 생성된 경우 AWS Glue Studio의 스크립트 편집기를 사용하여 작업 스크립트, 파라미터, 일정을 편집할 수 있습니다. 작업을 스크립트 전용 모드로 변환하여 AWS Glue Studio에서 생성된 작업의 스크립트를 편집할 수도 있습니다.

작업 스크립트를 편집하거나 자체 스크립트를 업로드하려면

1. 새 작업을 생성하는 경우 [작업(Jobs)] 페이지에서 [Spark 스크립트 편집기(Spark script editor)] 옵션을 선택하여 Spark 작업을 생성하거나 [Python 셸 스크립트 편집기(Python Shell script editor)]를 선택하여 Python 셸 작업을 생성합니다. 새 스크립트를 작성하거나 기존 스크립트를 업로드할 수 있습니다. [Spark 스크립트 편집기(Spark script editor)]를 선택하면 Scala 또는 Python 스크립트를 작성하거나 업로드할 수 있습니다. [Python Shell 스크립트 편집기(Python Shell script editor)]를 선택하면 Python 스크립트만 작성하거나 업로드할 수 있습니다.

   새 작업을 만드는 옵션을 선택한 후 나타나는 [옵션(Options)] 섹션에서 시작 스크립트로 시작하거나([표준 문안 코드로 새 스크립트 생성(Create a new script with boilerplate code)]) 작업 스크립트로 사용할 로컬 파일을 업로드할 수 있습니다.

   [Spark 스크립트 편집기(Spark script editor)]를 선택한 경우 Python 또는 Scala 스크립트 파일을 업로드할 수 있습니다. Scala 스크립트는 파일 확장명이 `.scala`여야 합니다. Python 스크립트는 Python 유형의 파일로 인식되어야 합니다. [Python Shell 스크립트 편집기(Python Shell script editor)]를 선택하면 Python 스크립트 파일만 업로드할 수 있습니다.

   선택을 마쳤으면 [생성(Create)]을 선택하여 작업을 생성하고 시각적 편집기를 엽니다.
2. 새 작업 또는 저장된 작업에 대한 시각적 작업 편집기로 이동한 다음 [스크립트(Script)] 탭을 선택합니다.
3. 스크립트 편집기 옵션 중 하나를 사용하여 새 작업을 생성하지 않았고 기존 작업에 대한 스크립트를 편집한 적이 없는 경우 [스크립트(Script)] 탭에 머리글 [스크립트(잠김)(Script (Locked))]가 표시

됩니다. 이는 스크립트 편집기가 읽기 전용 모드임을 의미합니다. [스크립트 편집(Edit script)]을 선택하여 편집할 스크립트의 잠금을 해제합니다.

스크립트를 편집 가능하게 만들기 위해 AWS Glue Studio는 작업을 시각적 작업에서 스크립트 전용 작업으로 변환합니다. 편집을 위해 스크립트를 잠금 해제하면 저장한 후 이 작업에 대해 더 이상 시각적 편집기를 사용할 수 없습니다.

확인 창에서 [확인(Confirm)]을 선택하여 계속하거나 [취소(Cancel)]를 선택하여 작업을 시각적 편집에 사용할 수 있도록 유지합니다.

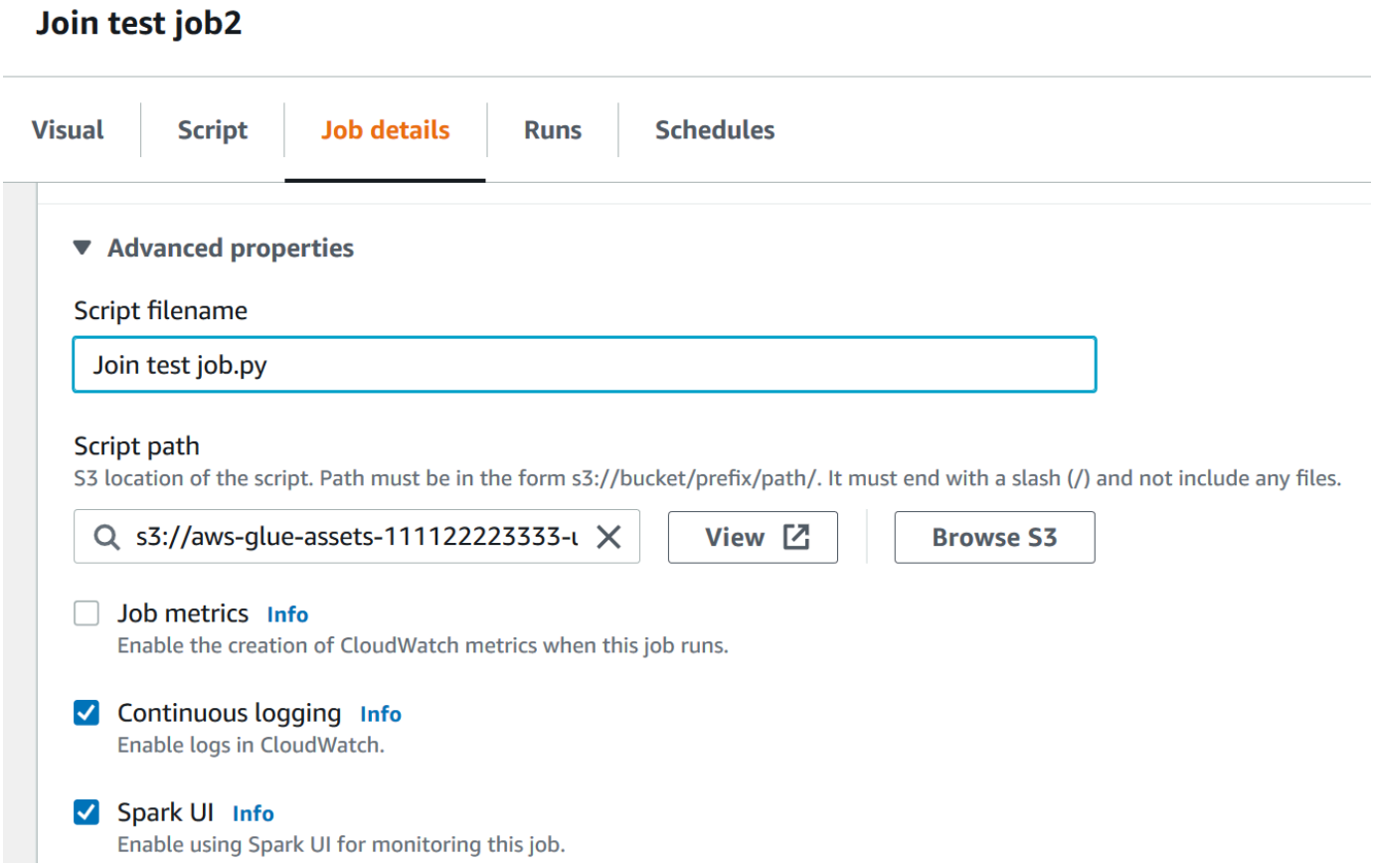
[확인(Confirm)]를 선택하면 [시각적(Visual)] 탭이 더 이상 편집기에 표시되지 않습니다. AWS Glue Studio를 사용하여 스크립트 편집기로 스크립트를 수정하거나, 작업 세부 정보 또는 일정을 수정하거나, 작업 실행을 볼 수 있습니다.

> **Note**
>
> 작업을 저장할 때까지 스크립트 전용 작업으로의 변환은 영구적이지 않습니다. 콘솔 웹 페이지를 새로 고치거나 저장하기 전에 작업을 다고 시각적 편집기에서 다시 열면 시각적 편집기에서 개별 노드를 계속 편집할 수 있습니다.

4. 필요에 따라 스크립트를 편집합니다.

   스크립트 편집을 마치면 [저장(Save)]을 선택하여 작업을 저장하고 시각적 객체에서 스크립트 전용으로 작업을 영구적으로 변환합니다.

5. (선택 사항) AWS Glue Studio 콘솔의 스크립트(Script) 탭에서 다운로드(Download) 버튼을 선택하여 스크립트를 다운로드할 수 있습니다. 이 버튼을 선택하면 새 브라우저 창이 열리고 Amazon S3의 해당 위치에 있는 스크립트가 표시됩니다. 작업의 [작업 세부 정보(Job details)] 탭에 있는 [스크립트 파일 이름(Script filename)] 및 [스크립트 경로(Script path)] 파라미터는 Amazon S3에 있는 스크립트 파일의 이름과 위치를 결정합니다.



작업을 저장할 때 AWS Glue는 이 필드에서 지정한 위치에 작업 스크립트를 저장합니다. Amazon S3 내의 이 위치에서 스크립트 파일을 수정할 경우 AWS Glue Studio는 다음에 작업을 편집할 때 수정된 스크립트를 로드합니다.

AWS Glue Studio에서 Scala 스크립트 생성 및 편집

작업 생성을 위해 스크립트 편집기를 선택하면 기본적으로 작업 프로그래밍 언어가 Python 3로 설정됩니다. 스크립트를 업로드하는 대신 새 스크립트를 작성하도록 선택할 경우 AWS Glue Studio는 Python으로 작성된 표준 문안 텍스트로 새 스크립트를 시작합니다. 대신 Scala 스크립트를 작성하려면 먼저 Scala를 사용하도록 스크립트 편집기를 구성해야 합니다.

> **Note**
>
> Scala를 작업의 프로그래밍 언어로 선택하고 시각적 편집기를 사용하여 작업을 설계하면 생성된 작업 스크립트가 Scala로 작성되고 추가 작업이 필요하지 않습니다.

AWS Glue Studio에서 새 Scala 스크립트를 작성하려면

1. [Spark 스크립트 편집기(Spark script editor)] 옵션을 선택하여 새 작업을 생성합니다.
2. [옵션(Options)]에서 [표준 문안 코드로 새 스크립트 생성(Create a new script with boilerplate code)]을 선택합니다.
3. [작업 세부 정보(Job details)] 탭을 선택하고 [언어(Language)]를 `Scala(Python 3` 대신)로 설정합니다.

> **Note**
>
> 작업을 생성하기 위해 [Spark 스크립트 편집기(Spark script editor)] 옵션을 선택하면 작업의 [유형(Type)] 속성이 `Spark`로 자동 설정됩니다.

4. [스크립트(Script)] 탭을 클릭합니다.
5. Python 표준 문안 텍스트를 제거합니다. 다음 Scala 표준 문안 텍스트로 바꿀 수 있습니다.

```
import com.amazonaws.services.glue.{DynamicRecord, GlueContext}
import org.apache.spark.SparkContext
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job

object MyScript {
  def main(args: Array[String]): Unit = {
    val sc: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(sc)

    }
}
```

6. 편집기에서 Scala 작업 스크립트를 작성합니다. 필요에 따라 `import` 문을 더 추가합니다.

AWS Glue Studio에서 Python 셸 작업 생성 및 편집

작업을 생성하기 위해 Python 셸 스크립트 편집기를 선택하면 기존 Python 스크립트를 업로드하거나 새 스크립트를 작성할 수 있습니다. 새 스크립트를 작성하도록 선택하면 표준 문안 코드가 새 Python 작업 스크립트에 추가됩니다.

새 Python 셸 작업을 생성하려면

AWS Glue Studio에서 작업 시작의 지침을 참조하세요.

Python 셸 작업에 지원되는 작업 속성은 Spark 작업에 지원되는 속성과 동일하지 않습니다. 다음 목록은 [작업 세부 정보(Job details)] 탭에서 Python 셸 작업에 사용 가능한 작업 파라미터의 변경 사항을 설명합니다.

- 작업의 [유형(Type)] 속성은 `Python Shell`로 자동 설정되며 변경할 수 없습니다.
- [언어(Language)] 대신 작업에 대한 [Python 버전(Python version)] 속성이 있습니다. 현재 AWS Glue Studio에서 생성된 Python 셸 작업은 Python 3.6을 사용합니다.
- [Glue 버전(Glue version)] 속성은 Python 셸 작업에 적용되지 않으므로 사용할 수 없습니다.
- [작업자 유형(Worker type)] 및 [작업자 수(Number of workers)] 대신 [데이터 처리 장치(Data processing units)] 속성이 표시됩니다. 이 작업 속성은 작업을 실행할 때 Python 셸에서 사용하는 DPU(데이터 처리 장치) 수를 결정합니다.
- [작업 북마크(Job bookmark)] 속성은 Python 셸 작업에 지원되지 않기 때문에 사용할 수 없습니다.
- [고급 속성(Advanced properties)]에서 Python 셸 작업에는 다음 속성을 사용할 수 없습니다.
  - 작업 지표
  - 연속 로깅
  - [Spark UI] 및]Spark UI 로그 경로(Spark UI logs path)]
  - 머리글 [라이브러리(Libraries)] 아래의 [종속 jar 경로(Dependent jars path)]

작업 다이어그램에서 노드의 상위 노드 변경

노드의 상위 항목을 변경하여 작업 다이어그램 내에서 노드를 이동하거나 노드의 데이터 원본을 변경할 수 있습니다.

상위 노드를 변경하려면

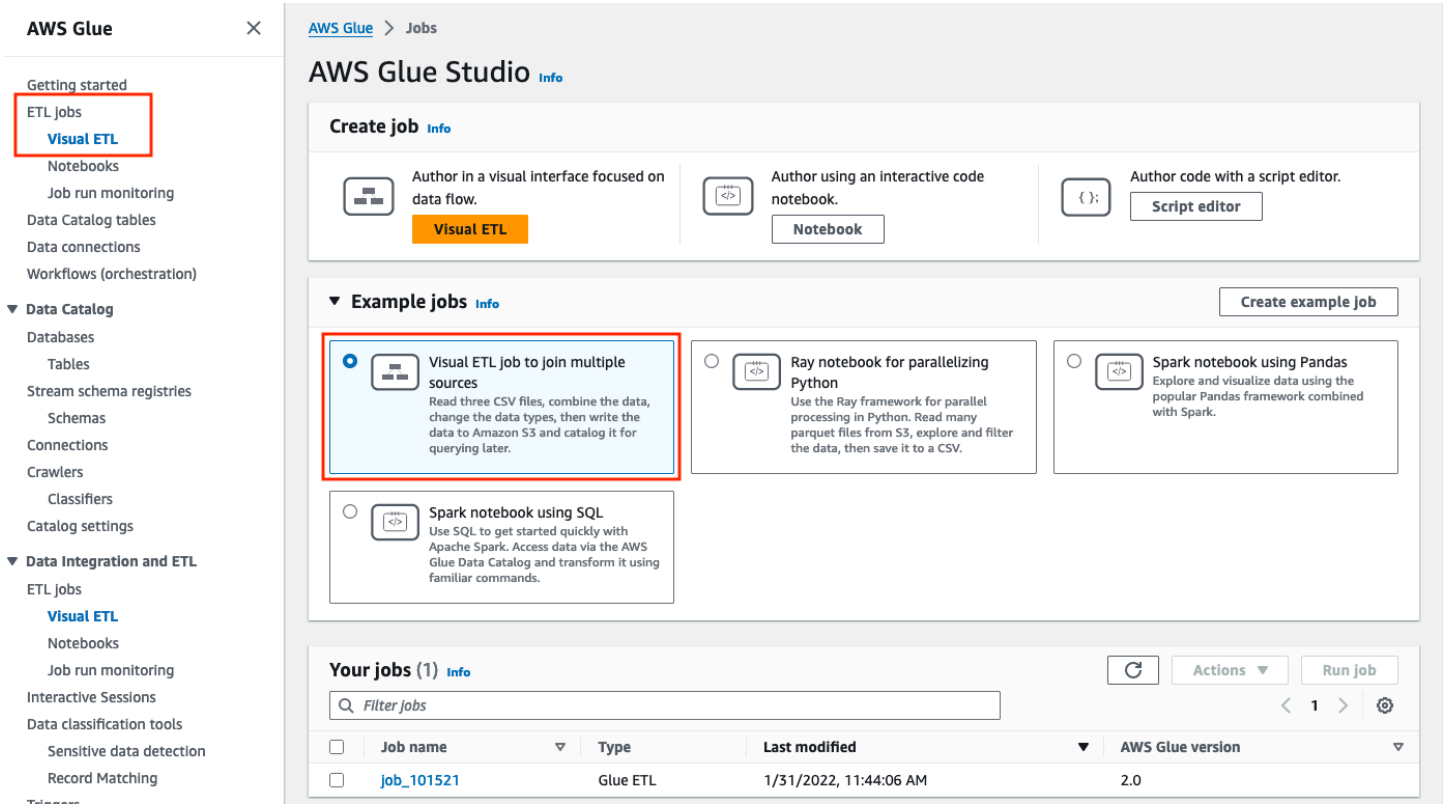
1. 수정할 작업 다이어그램의 노드를 선택합니다.
2. 노드 세부 정보 패널의 [노드 속성] 탭에 있는 [노드 상위 항목(Node parents)] 아래에서 노드의 현재 상위 항목을 제거합니다.
3. 목록에서 새 상위 노드를 선택합니다.
4. 새로 선택한 상위 노드와 일치하도록 필요에 따라 노드의 다른 속성을 수정합니다.

실수로 노드를 수정한 경우 도구 모음의 [실행 취소(Undo)] 버튼을 사용하여 작업을 되돌릴 수 있습니다.

작업 다이어그램에서 노드 삭제

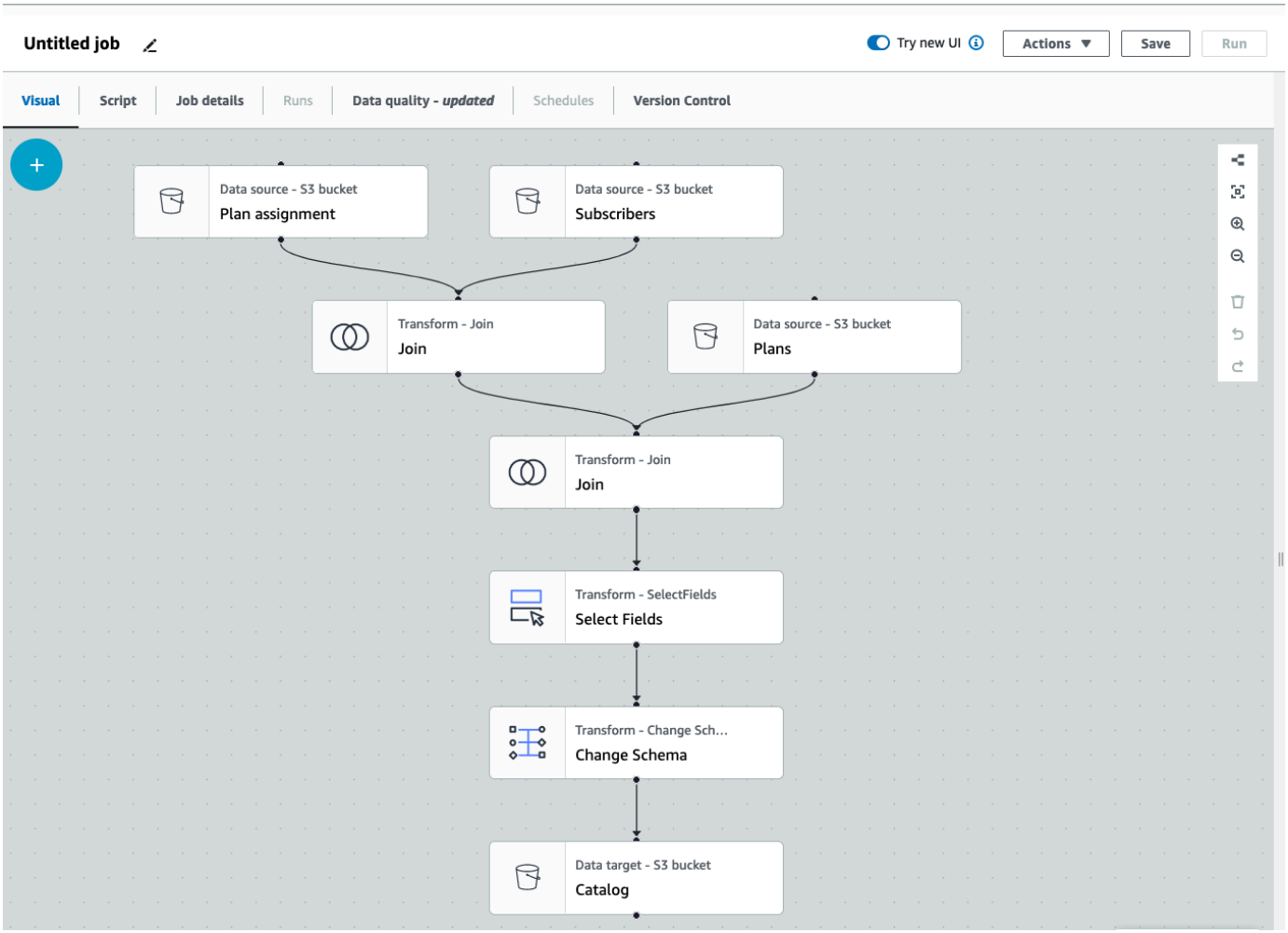
Visual ETL 작업을 수행할 때 제거된 노드에 연결된 노드를 다시 추가하거나 재구성하지 않고도 캔버스에서 노드를 제거할 수 있습니다.

아래 예제에서는 ETL 작업 > Visual ETL을 선택한 다음, 예제 작업에서 여러 소스를 조인하는 Visual ETL 작업을 선택합니다. 예제 작업 생성을 선택하여 작업을 생성하고 아래 단계를 따르세요.



캔버스에서 노드를 제거하려면

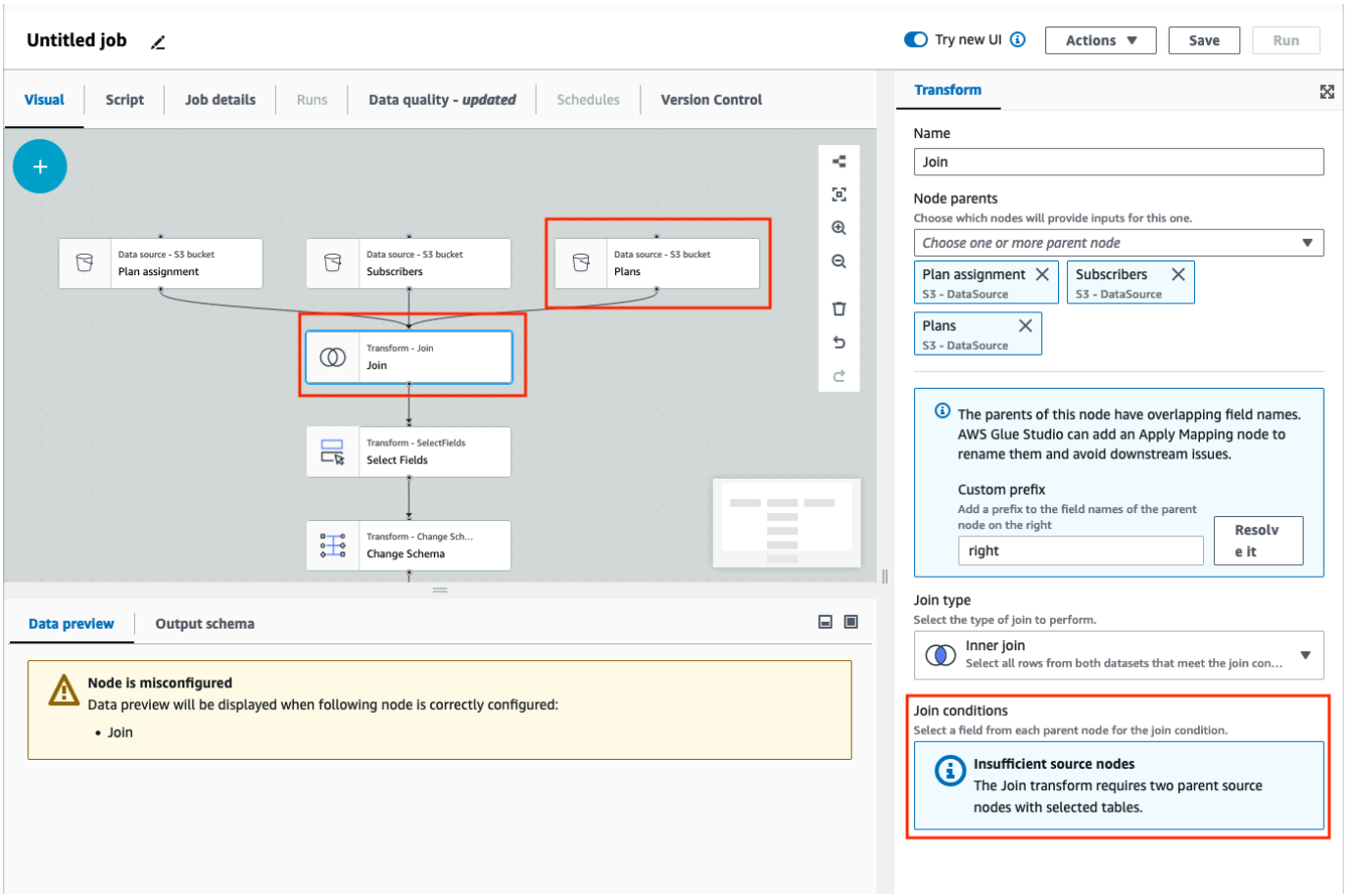
1. AWS Glue 콘솔의 탐색 메뉴에서 Visual ETL을 선택하고 기존 작업을 선택합니다. 작업 캔버스는 아래 그림과 같이 예제 작업을 표시합니다.



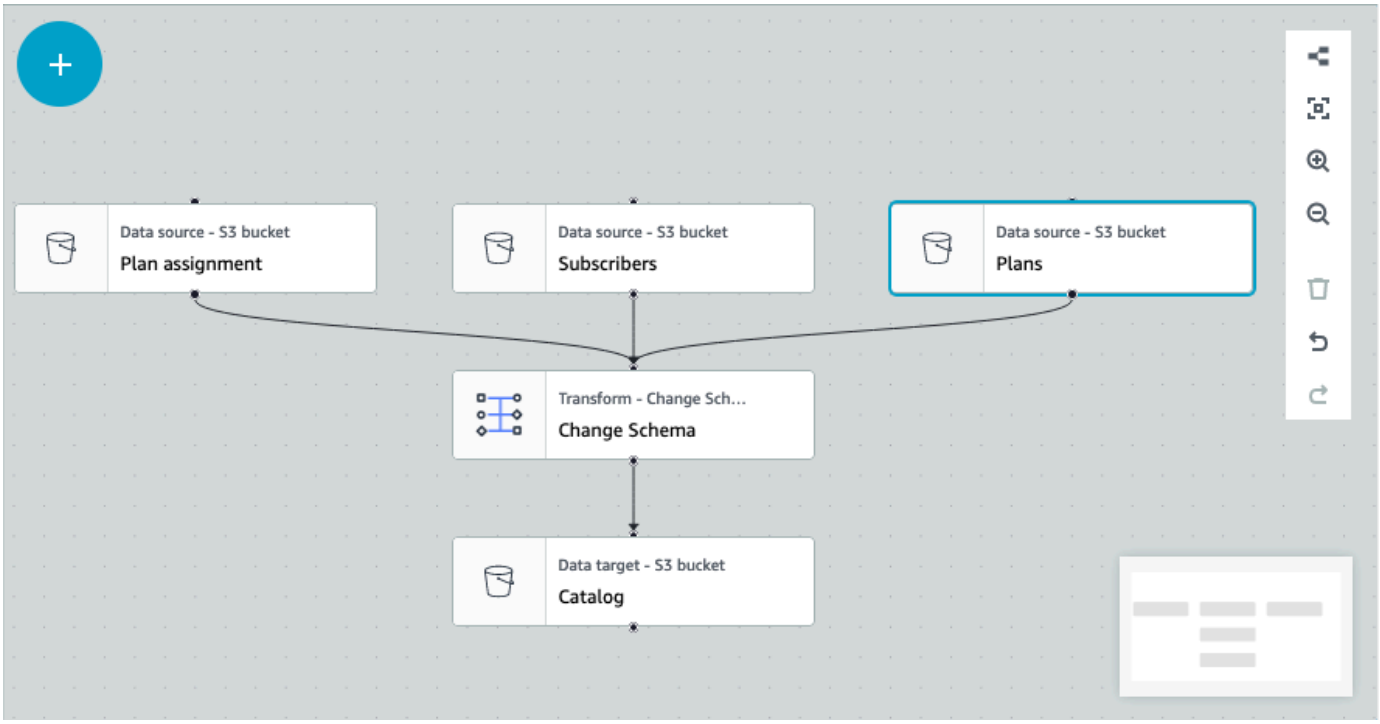
2. 제거할 노드를 선택합니다. 캔버스가 노드로 확대됩니다. 캔버스 오른쪽에 있는 도구 모음에서 휴지통 아이콘을 선택합니다. 이렇게 하면 노드가 제거되고 노드에 연결된 모든 노드가 워크플로에서 해당 위치로 이동됩니다. 이 예제에서는 첫 번째 Join 노드가 캔버스에서 삭제되었습니다.

   워크플로에서 노드를 삭제하면 AWS Glue에서 노드를 다시 정렬하여 워크플로가 유효하지 않은 방식으로 구성되지 않도록 합니다. 여전히 노드 구성을 수정해야 할 수도 있습니다.

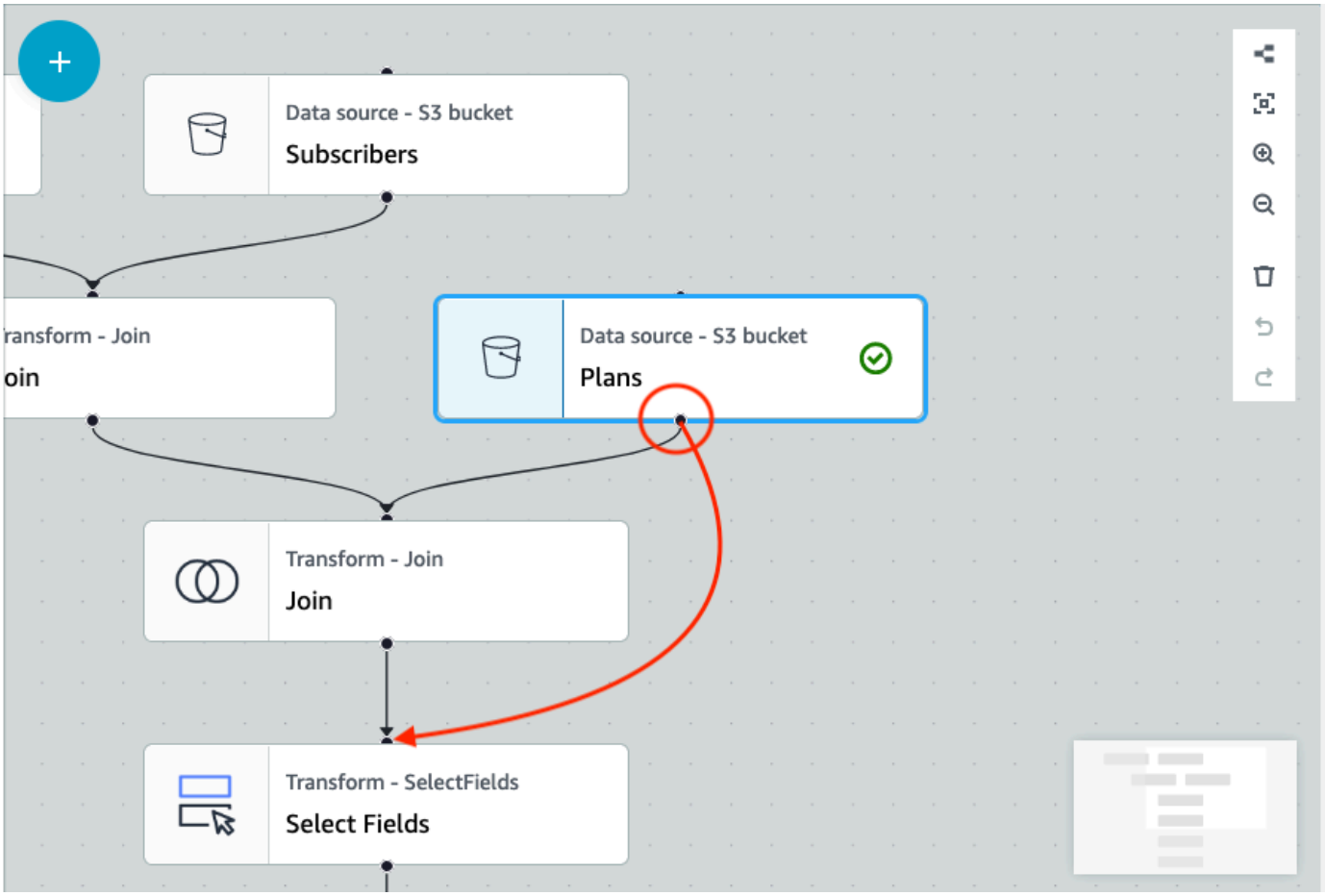
   이 예제에서는 Subscribers 노드 아래의 Join 노드가 제거되었습니다. 따라서 Plans 소스 노드는 최상위 수준으로 이동되었으며 여전히 하위 Join 노드에 연결되어 있습니다. Join에는 선택한 테이블이 있는 두 개의 상위 소스 노드가 필요하므로 Join 노드에는 추가 구성이 필요합니다. 캔버스 오른쪽에 있는 변환 탭에는 조인 조건에서 누락된 요구 사항이 표시됩니다.



3. 두 번째 Join 노드와 Select Fields 노드를 삭제합니다. 노드가 삭제되면 워크플로는 아래 예제와 같이 표시됩니다.



4. 노드 연결을 수정하려면 노드 핸들을 클릭하고 연결을 새 노드로 끌어 놓습니다. 이렇게 하면 노드를 삭제하고 논리적 흐름에서 노드를 다시 정렬할 수 있습니다. 예제에서는 빨간색 화살표로 표시된 대로 Plans 노드의 핸들을 클릭하고 연결을 Join 노드로 끌어서 새 연결을 생성합니다.



5. 작업을 취소해야 하는 경우 캔버스 오른쪽 도구 모음의 휴지통 아이콘 바로 아래에 있는 실행 취소 아이콘을 선택하세요.

AWS Glue 데이터 카탈로그 노드에 소스 및 대상 파라미터 추가

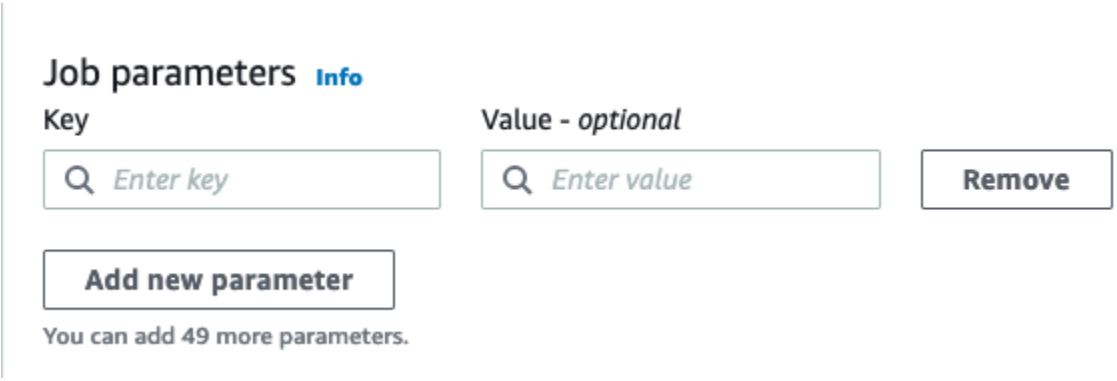
AWS Glue Studio은(는) 시각적 작업을 파라미터화할 수 있습니다. 프로덕션 및 개발 환경에서는 카탈로그 테이블 이름이 다를 수 있으므로 작업이 실행될 때 실행할 데이터베이스 및 테이블의 런타임 파라미터를 정의하고 선택할 수 있습니다.

작업 파라미터화를 사용하면 소스 및 대상을 파라미터화하고, AWS Glue 데이터 카탈로그 노드를 사용할 때 해당 파라미터를 작업에 저장할 수 있습니다. 소스 및 대상을 파라미터로 지정하면 특히 여러 환경에서 동일한 작업을 사용할 때 작업을 재사용할 수 있습니다. 소스 및 대상을 관리하는 데 드는 시간과 노력을 절약함으로써, 이는 배포 환경 전반에서 코드를 승격할 때 유용합니다. 또한 지정한 사용자 지정 파라미터는 AWS Glue 작업의 특정 실행에 대한 기본 인수를 재정의합니다.

소스 및 대상 파라미터 추가

AWS Glue 데이터 카탈로그 노드를 소스 또는 대상으로 사용하는지 관계없이 Job details(작업 세부 정보) 탭의 Advanced properties(고급 속성) 섹션에서 런타임 파라미터를 정의할 수 있습니다.

1. 소스 노드 또는 대상 노드로 AWS Glue 데이터 카탈로그를 선택합니다.
2. [작업 세부 정보(Job details)] 탭을 선택합니다.
3. Advanced properties(고급 속성)을 선택합니다.
4. 작업 파라미터 섹션에서 키 값을 입력합니다. 예를 들어, `--db.source`은(는) 데이터베이스 원본에 대한 파라미터가 될 것입니다. 키 이름 뒤에 'dash dash'가 오면 아무 이름이나 입력할 수 있습니다.

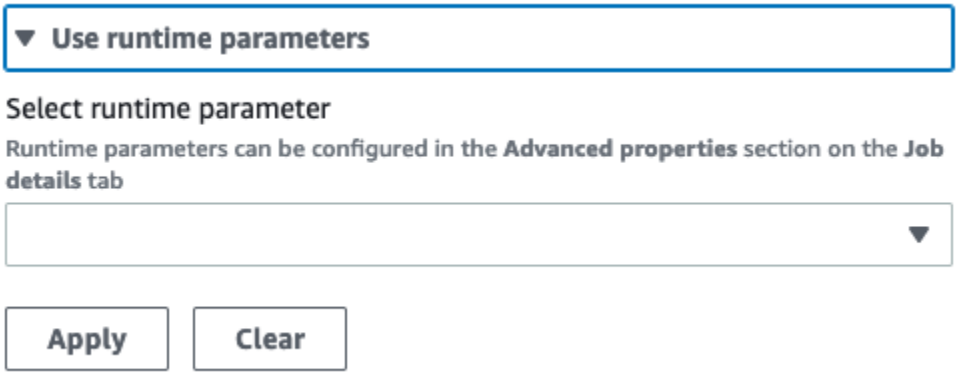


5. 값을 입력합니다. 예를 들어, `databasename`은(는) 파라미터화되는 데이터베이스의 값이 될 것입니다.
6. 파라미터를 더 추가하고자 하는 경우 Add new parameter(새 파라미터 추가)를 선택합니다. 최대 50개의 파리미터를 추가할 수 있습니다. 키 값 쌍이 정의되면 AWS Glue 데이터 카탈로그 노드에서 파라미터를 사용할 수 있습니다.

런타임 파라미터 선택

> **Note**
>
> 데이터베이스 및 테이블에 대한 런타임 파라미터를 선택하는 프로세스는 AWS Glue 데이터 카탈로그 노드가 소스 또는 대상인지와 상관없이 동일합니다.

1. 소스 노드 또는 대상 노드로 AWS Glue 데이터 카탈로그를 선택합니다.
2. Database(데이터베이스)의 Data source properties - Data Catalog(데이터 소스 속성 - 데이터 카탈로그) 탭에서 Use runtime parameters(런타임 파라미터 사용)을 선택합니다.



3. 드롭다운 메뉴에서 파라미터를 선택합니다. 예를 들어, 소스 데이터베이스에 대해 정의한 파라미터를 선택하면 Apply(적용)을 선택하면 데이터베이스가 자동으로 데이터베이스 드롭다운 메뉴에 채워집니다.
4. 테이블(Table) 섹션에서 이미 소스 테이블로 정의한 파라미터를 선택합니다. Apply(적용)을 선택하면 테이블이 사용할 테이블로 자동으로 채워집니다.
5. 작업을 저장하고 실행하면 AWS Glue Studio이(가) 작업 실행 중에 선택한 파라미터를 참조합니다.

AWS Glue에서 Git 버전 제어 시스템 사용

> **Note**
>
> 노트북은 현재 AWS Glue Studio에서 버전 관리가 지원되지 않습니다. 하지만 AWS Glue 작업 스크립트 및 시각적 ETL 작업에 대한 버전 제어는 지원됩니다.

원격 리포지토리가 있고 리포지토리를 사용하여 AWS Glue 작업을 관리하려는 경우 AWS Glue Studio 또는 AWS CLI을(를) 사용하여 AWS Glue의 리포지토리 및 작업에 대한 변경 사항을 동기화할 수 있습니다. 이러한 방식으로 변경 내용을 동기화하면 작업을 AWS Glue Studio에서 리포지토리로 푸시하거나 리포지토리에서 AWS Glue Studio(으)로 가져옵니다.

AWS Glue Studio의 Git 통합을 통해 다음 작업을 수행할 수 있습니다.

- AWS CodeCommit, GitHub, GitLab 및 Bitbucket과 같은 Git 버전 관리 시스템과 통합
- 시각적 작업을 사용하든 스크립트 작업을 사용하든 AWS Glue Studio에서 AWS Glue 작업을 편집하고 이를 리포지토리에 동기화
- 작업의 소스 및 대상을 파라미터화

- 리포지토리에서 작업을 가져와 AWS Glue Studio에서 편집합니다.
- AWS Glue Studio의 다중 브랜치 워크플로를 활용하여 브랜치에서 가져오거나 브랜치로 푸시하여 작업 테스트
- 교차 계정 작업 생성에 대해 리포지토리에서 파일 다운로드 및 AWS Glue Studio(으)로 작업 업로드
- 선택한 자동화 도구 사용(예: Jenkins, AWS CodeDeploy, 등.)

이 비디오에서는 AWS Glue를 Git와 통합하고 협업에 기반한 지속적인 코드 파이프라인을 구축하는 방법을 보여줍니다.

IAM 권한

작업에 다음 IAM 권한 중 하나가 있는지 확인합니다. IAM 권한 설정 방법에 대한 자세한 내용을 알아보려면 AWS Glue Studio에 대한 IAM 권한 설정을 참조하세요.

- `AWSGlueServiceRole`
- `AWSGlueConsoleFullAccess`

Git 통합을 위해서는 최소한 다음과 같은 작업이 필요합니다.

- `glue:UpdateJobFromSourceControl` - 버전 관리 시스템에 있는 작업으로 AWS Glue을(를) 업데이트할 수 있습니다.
- `glue:UpdateSourceControlFromJob` - AWS Glue에 저장된 작업으로 버전 관리 시스템을 업데이트할 수 있습니다.
- `s3:GetObject` - 버전 관리 시스템으로 푸시하는 동안 작업에 대한 스크립트를 검색할 수 있습니다.
- `s3:PutObject` - 소스 제어 시스템에서 작업을 가져올 때 스크립트를 업데이트할 수 있습니다.

사전 조건

작업을 소스 제어 리포지토리로 푸시하려면 다음 사항이 필요합니다.

- 관리자가 이미 생성한 리포지토리
- 리포지토리 내 브랜치
- 개인용 액세스 토큰(Bitbucket의 경우 리포지토리 액세스 토큰)
- 리포지토리 소유자의 사용자 이름

- AWS Glue Studio의 리포지토리에 대해 읽기 및 쓰기가 허용되도록 리포지토리의 권한을 설정합니다
  - GitLab — 토큰 범위를 API, read_repository, write_repository로 설정
  - Bitbucket — 권한 설정 위치:
    - Workspace 멤버십 — 읽기, 쓰기
    - 프로젝트 — 쓰기, 관리자 읽기
    - 리포지토리 — 읽기, 쓰기, 관리, 삭제

> **Note**
>
> AWS CodeCommit 사용 시 개인 액세스 토큰 및 리포지토리 소유자는 필요하지 않습니다. Git 및 AWS CodeCommit 시작하기를 참조하세요.

AWS Glue Studio에서 소스 제어 리포지토리의 작업 사용

AWS Glue Studio에 없는 작업을 소스 제어 리포지토리에서 가져오고 AWS Glue Studio에서 작업을 사용하려고 하면 작업 유형에 따라 전제 조건이 달라집니다.

시각적 작업의 경우:

- 작업 이름과 일치하는 작업 정의의 폴더 및 JSON 파일이 필요합니다

  예를 들어, 아래 작업 정의를 참조하세요. 리포지토리의 브랜치에는 폴더와 JSON 파일이 모두 작업 이름과 일치하는 경로 `my-visual-job/my-visual-job.json`이(가) 포함되어야 합니다

```
{
  "name" : "my-visual-job",
  "description" : "",
  "role" : "arn:aws:iam::aws_account_id:role/Rolename",
  "command" : {
    "name" : "glueetl",
    "scriptLocation" : "s3://foldername/scripts/my-visual-job.py",
    "pythonVersion" : "3"
  },
  "codeGenConfigurationNodes" : "{\"node-nodeID\":{\"S3CsvSource\":
{\"AdditionalOptions\":{\"EnableSamplePath\":false,\"SamplePath\":\"s3://notebook-
test-input/netflix_titles.csv\"},\"Escaper\":\"\",\"Exclusions\":[],\"Name\":\"Amazon
 S3\",\"OptimizePerformance\":false,\"OutputSchemas\":[{\"Columns\":[{\"Name\":
```

```
\"show_id\",\"Type\":\"string\"},{\"Name\":\"type\",\"Type\":\"string\"},{\"Name\":
\"title\",\"Type\":\"choice\"},{\"Name\":\"director\",\"Type\":\"string\"},{\"Name\":
\"cast\",\"Type\":\"string\"},{\"Name\":\"country\",\"Type\":\"string\"},{\"Name\":
\"date_added\",\"Type\":\"string\"},{\"Name\":\"release_year\",\"Type\":\"bigint\"},
{\"Name\":\"rating\",\"Type\":\"string\"},{\"Name\":\"duration\",\"Type\":\"string
\"},{\"Name\":\"listed_in\",\"Type\":\"string\"},{\"Name\":\"description\",\"Type
\":\"string\"}]}],\"Paths\":[\"s3://dalamgir-notebook-test-input/netflix_titles.csv
\"],\"QuoteChar\":\"quote\",\"Recurse\":true,\"Separator\":\"comma\",\"WithHeader
\":true}}}"
}
```

스크립트 작업의 경우:

- 폴더, 작업 정의의 JSON 파일 및 스크립트가 필요합니다.
- 폴더 및 JSON 파일은 작업 이름과 일치해야 합니다. 스크립트 이름은 파일 확장자와 함께 작업 정의의 `scriptLocation`와(과) 일치해야 합니다

  예를 들어, 아래 작업 정의에서 리포지토리의 브랜치에는 경로 `my-script-job/my-script-job.json` 및 `my-script-job/my-script-job.py`이(가) 포함되어야 합니다. 스크립트 이름은 스크립트의 확장자를 포함하여 `scriptLocation`의 이름과 일치해야 합니다.

```
{
  "name" : "my-script-job",
  "description" : "",
  "role" : "arn:aws:iam::aws_account_id:role/Rolename",
  "command" : {
    "name" : "glueetl",
    "scriptLocation" : "s3://foldername/scripts/my-script-job.py",
    "pythonVersion" : "3"
  }
}
```

제한 사항

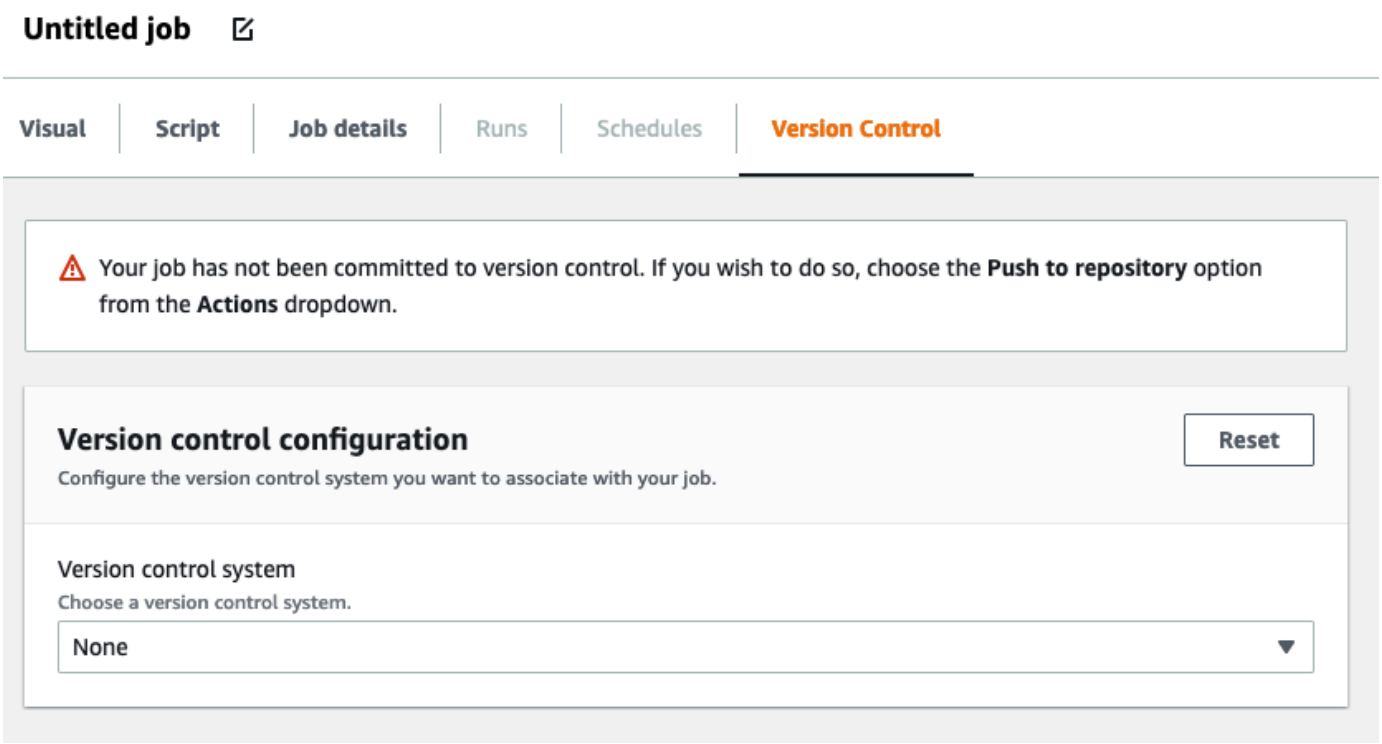
- AWS Glue는 현재 GitLab-Groups에서 푸시/풀링을 지원하지 않습니다.

버전 관리 리포지토리를 AWS Glue와(과) 연결

AWS Glue Studio 작업 편집기의 Version Control(버전 관리) 탭에서 버전 관리 리포지토리 세부 정보를 입력하고 관리할 수 있습니다. Git 리포지토리와 통합하려면 AWS Glue Studio에 로그인할 때마다 리포지토리에 연결해야 합니다.

Git 버전 관리 시스템 연결하기:

1. AWS Glue Studio에서 새 작업을 시작하고 Version Control(버전 관리) 탭을 선택합니다.



2. 버전 관리 시스템에서 다음 드롭다운 메뉴를 클릭하여 사용 가능한 옵션 중에서 Git Service를 선택합니다.

   - AWS CodeCommit
   - GitHub
   - GitLab
   - Bitbucket

3. 선택한 Git 버전 관리 시스템에 따라 완료해야 하는 필드가 달라집니다.

   AWS CodeCommit의 경우

   작업에 사용할 리포지토리와 브랜치를 선택하여 리포지토리 구성을 완료합니다.

- 리포지토리 - AWS CodeCommit에 리포지토리를 설정한 경우 드롭다운 메뉴에서 리포지토리를 선택합니다. 리포지토리가 목록에 자동으로 채워집니다.
- 브랜치 - 드롭다운 메뉴에서 브랜치를 선택합니다.
- 폴더 - 선택 사항- 작업을 저장할 폴더 이름을 입력합니다. 비워 두면 폴더가 자동으로 생성됩니다. 폴더 이름은 기본적으로 작업 이름입니다.

GitHub의 경우:

다음 필드를 작성하여 GitHub 구성을 완료합니다.

- Personal access token(개인 액세스 토큰) - GitHub 리포지토리에서 제공하는 토큰입니다. 개인 액세스 토큰에 대한 자세한 내용을 알아보려면 GitHub 문서를 참조하세요.
- 리포지토리 소유자 - GitHub 리포지토리의 소유자입니다.

GitHub에서 사용할 리포지토리와 브랜치를 선택하여 리포지토리 구성을 완료합니다.

- 리포지토리 - GitHub에 리포지토리를 설정한 경우 드롭다운 메뉴에서 리포지토리를 선택합니다. 리포지토리가 목록에 자동으로 채워집니다.
- 브랜치 - 드롭다운 메뉴에서 브랜치를 선택합니다.
- 폴더 - 선택 사항- 작업을 저장할 폴더 이름을 입력합니다. 비워 두면 폴더가 자동으로 생성됩니다. 폴더 이름은 기본적으로 작업 이름입니다.

GitLab의 경우:

> **Note**
>
> AWS Glue는 현재 GitLab-Groups에서 푸시/풀링을 지원하지 않습니다.

- 개인 액세스 토큰 - GitLab 리포지토리에서 제공하는 토큰입니다. 개인 액세스 토큰에 대한 자세한 내용은 GitLab 개인 액세스 토큰을 참조하세요
- 리포지토리 소유자 - GitLab 리포지토리의 소유자입니다.

GitLab에서 사용할 리포지토리와 브랜치를 선택하여 리포지토리 구성을 완료합니다.

- 리포지토리 - GitLab에 리포지토리를 설정한 경우 드롭다운 메뉴에서 리포지토리를 선택합니다. 리포지토리가 목록에 자동으로 채워집니다.
- 브랜치 - 드롭다운 메뉴에서 브랜치를 선택합니다.
- 폴더 - 선택 사항- 작업을 저장할 폴더 이름을 입력합니다. 비워 두면 폴더가 자동으로 생성됩니다. 폴더 이름은 기본적으로 작업 이름입니다.

Bitbucket의 경우:

- 앱 암호 - Bitbucket은 리포지토리 액세스 토큰이 아닌 앱 암호를 사용합니다. 앱 암호에 대한 자세한 내용은 앱 암호를 참조하세요.
- 리포지토리 소유자 - Bitbucket 리포지토리의 소유자입니다. Bitbucket에서 소유자는 리포지토리의 생성자입니다.

Bitbucket에서 사용할 워크스페이스, 리포지토리, 브랜치 및 폴더를 선택하여 리포지토리 구성을 완료합니다.

- 작업 영역 — Bitbucket에 작업 영역을 설정한 경우 드롭다운 메뉴에서 작업 영역을 선택합니다. 작업 공간이 자동으로 채워집니다
- 리포지토리- Bitbucket에 리포지토리를 설정한 경우 드롭다운 메뉴에서 리포지토리를 선택합니다. 리포지토리가 자동으로 채워집니다
- 브랜치 — 드롭다운 메뉴에서 브랜치를 선택합니다. 브랜치가 자동으로 채워집니다
- 폴더 - 선택 사항- 작업을 저장할 폴더 이름을 입력합니다. 비워 두면 작업 이름의 폴더가 자동으로 생성됩니다.

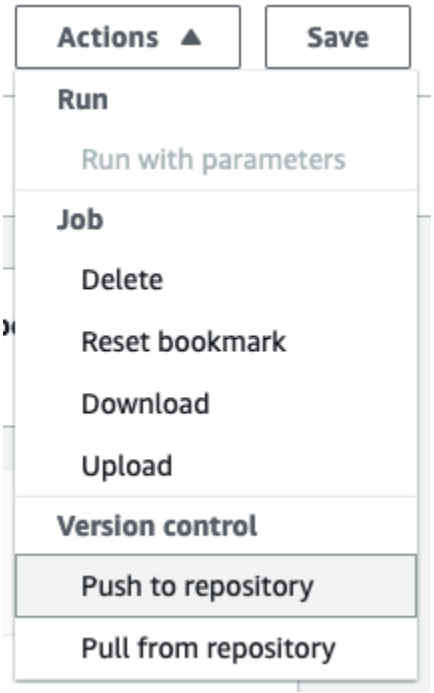
4. AWS Glue Studio 작업의 페이지 상단에서 Save(저장)을 선택합니다.

소스 리포지토리에 AWS Glue 작업 푸시

버전 관리 시스템의 세부 정보를 입력한 후에는 AWS Glue Studio에서 작업을 편집하고 작업을 소스 리포지토리로 푸시할 수 있습니다. 푸시 및 가져오기 같은 Git 개념에 익숙하지 않은 경우 Git 및 AWS CodeCommit 시작하기에 대한 이 자습서를 참조하세요.

작업을 리포지토리에 푸시하려면 버전 관리 시스템의 세부 정보를 입력하고 작업을 저장해야 합니다.

1. AWS Glue Studio 작업에서 Actions(작업)을 선택합니다. 그러면 추가 메뉴 옵션이 열립니다.



2. Push to repository(리포지토리로 푸시)를 선택합니다.

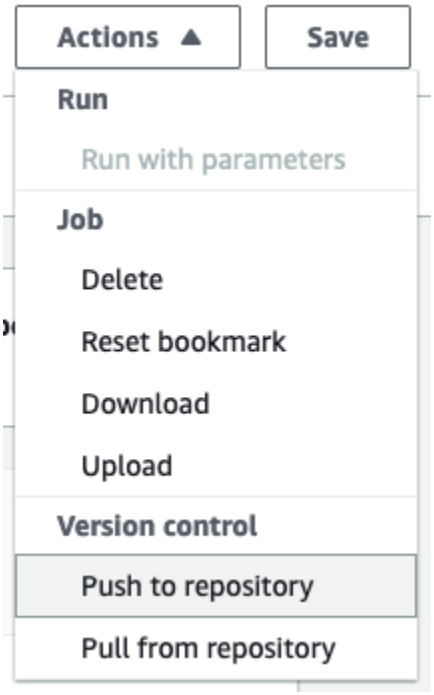
   이 작업을 수행하면 작업이 저장됩니다. 리포지토리로 푸시하면 AWS Glue Studio이(가) 마지막으로 저장한 변경 내용을 푸시합니다. 리포지토리의 작업이 사용자 본인 또는 다른 사용자에 의해 수정되었으며 AWS Glue Studio의 작업과 동기화되지 않은 경우 AWS Glue Studio에서 작업을 푸시할 때 AWS Glue Studio에서 저장된 작업으로 리포지토리의 작업이 덮어써 집니다.

3. Confirm(확인)을 선택하여 작업을 완료합니다. 이렇게 하면 리포지토리에서 새 커밋을 생성합니다. AWS CodeCommit을(를) 사용하는 경우 확인 메시지에 AWS CodeCommit에 대한 최신 커밋에 대한 링크가 표시됩니다.

소스 리포지토리에서 AWS Glue 작업 가져오기

Version control(버전 관리) 탭에 Git 리포지토리의 세부 정보를 입력한 후에는 리포지토리에서 작업을 가져와 AWS Glue Studio에서 편집할 수도 있습니다.

1. AWS Glue Studio 작업에서 Actions(작업)을 선택합니다. 그러면 추가 메뉴 옵션이 열립니다.



2. Pull from repository(리포지토리에서 가져오기)를 선택합니다.
3. 확인을 선택합니다. 이렇게 하면 리포지토리에서 최신 커밋을 가져와 AWS Glue Studio에서 작업을 업데이트합니다.
4. AWS Glue Studio에서 작업을 편집합니다. 변경할 경우 Actions(작업) 드롭다운 메뉴에서 Push to repository(리포지토리로 푸시)를 선택하여 작업을 리포지토리에 동기화할 수 있습니다.

AWS Glue Studio 노트북을 사용한 코드 작성

데이터 엔지니어는 AWS Glue Studio의 대화형 노트북 인터페이스 또는 AWS Glue의 대화형 세션을 사용하기 전보다 더 쉽고 빠르게 AWS Glue 작업을 작성할 수 있습니다.

제한 사항

- AWS Glue Studio 노트북은 Scala를 지원하지 않습니다.

주제

- 노트북 사용 개요
- AWS Glue Studio에서 노트북을 사용하여 ETL 작업 생성
- 노트북 편집기 구성 요소
- 노트북 및 작업 스크립트 저장

- 노트북 세션 관리
- Amazon Q Developer를 AWS Glue Studio 노트북과 함께 사용하기

노트북 사용 개요

AWS Glue Studio를 사용하면 Jupyter Notebook 기반의 노트북 인터페이스에서 작업을 대화형으로 작성할 수 있습니다. AWS Glue Studio의 노트북을 통해 작업 스크립트를 편집한 후 전체 작업을 실행하지 않고도 출력을 볼 수 있으며, 데이터 통합 코드를 편집한 후 전체 작업을 실행하지 않고도 출력을 볼 수 있으며, 마크다운을 추가하고 노트북을 .ipynb 파일과 작업 스크립트로 저장할 수 있습니다. 소프트웨어를 로컬로 설치하거나 서버를 관리하지 않고도 노트북을 시작할 수 있습니다. 코드가 만족스러우면 AWS Glue Studio에서 하나의 버튼을 클릭하여 노트북을 Glue 작업으로 변환할 수 있습니다.

노트북을 사용하는 경우의 몇 가지 이점은 다음과 같습니다.

- 프로비저닝하거나 관리할 클러스터가 없습니다.
- 요금을 결제할 유휴 클러스터가 없습니다.
- 사전 구성이 필요하지 않습니다.
- Jupyter Notebook을 설치할 필요가 없습니다.
- AWS Glue ETL과 동일한 런타임/플랫폼입니다.

AWS Glue Studio를 통해 노트북을 시작하는 경우 데이터를 탐색하고 몇 초 만에 작업 스크립트 개발을 시작할 수 있도록 모든 구성 단계가 완료됩니다. AWS Glue Studio는 AWS Glue Jupyter 커널을 사용하여 Jupyter Notebook을 구성합니다. 이 노트북을 사용하기 위해 VPC, 네트워크 연결 또는 개발 엔드포인트를 구성할 필요는 없습니다.

노트북 인터페이스를 사용하여 작업을 생성하려면

- 필요한 IAM 권한을 구성합니다.
- 노트북 세션을 시작하여 작업을 생성합니다.
- 노트북의 셀에 코드를 작성합니다.
- 코드를 실행하고 테스트하여 출력을 봅니다.
- 작업 저장

저장한 후의 노트북은 전체 AWS Glue 작업입니다. 작업 실행 예약, 작업 파라미터 설정, 노트북과 작업 실행 기록 나란히 보기 등 작업의 모든 측면을 관리할 수 있습니다.

AWS Glue Studio에서 노트북을 사용하여 ETL 작업 생성

AWS Glue Studio 콘솔에서 노트북 사용을 시작하려면

1. AWS Identity and Access Management 정책을 AWS Glue Studio 사용자에게 연결하고 ETL 작업 및 노트북에 대한 IAM 역할을 생성합니다.
2. IAM 역할에 권한 부여에 설명된 대로 노트북에 대한 추가 IAM 보안을 구성합니다.
3. https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.

   > **Note**
   >
   > 브라우저가 서드 파티 쿠키를 차단하지 않는지 확인합니다. 브라우저가 기본값 또는 사용자가 지정한 설정으로 서드 파티 쿠키를 차단하는 경우 노트북이 실행되지 않습니다. 쿠키 관리에 관한 자세한 내용은 다음을 참조하세요.

   - Chrome
   - Firefox
   - Safari
4. 왼쪽 탐색 메뉴에서 작업(Jobs) 링크를 선택합니다.
5. Jupyter Notebook(Jupyter notebook)을 선택한 다음 생성(Create)을 선택하여 새 노트북 세션을 시작합니다.
6. Jupyter Notebook에서 작업 생성 페이지에서 작업 이름을 제공하고 사용할 IAM 역할을 선택합니다. 작업 생성(Create job)을 선택합니다.

   잠시 후에 노트북 편집기가 나타납니다.
7. 코드를 추가한 후 셀을 실행하여 세션을 시작해야 합니다. 다음과 같은 여러 가지 방법으로 셀을 실행할 수 있습니다.

   - 재생 버튼을 누릅니다.
   - 키보드 단축키를 사용합니다.
     - MacOS에서 Command + Enter를 눌러 셀을 실행합니다.
     - Windows에서 Shift + Enter를 눌러 셀을 실행합니다.

Jupyter Notebook 인터페이스를 사용하여 코드를 작성하는 방법에 대한 자세한 내용은 Jupyter Notebook 사용 설명서를 참조하세요.

8. 스크립트를 테스트하려면 전체 스크립트 또는 개별 셀을 실행합니다. 모든 명령 출력은 셀 아래 영역에 표시됩니다.
9. 노트북 개발을 마친 후 작업을 저장하고 실행할 수 있습니다. 스크립트(Script) 탭에서 스크립트를 찾을 수 있습니다. 노트북에 추가한 모든 매직은 제거되고 생성된 AWS Glue 작업의 스크립트의 일부로 저장되지 않습니다. AWS Glue Studio는 노트북 내용에서 생성된 스크립트의 끝에 `job.commit()`을 자동으로 추가합니다.

   작업 실행 방법에 대한 자세한 내용은 작업 실행 시작 섹션을 참조하세요.

노트북 편집기 구성 요소

노트북 편집기 인터페이스에는 다음과 같은 기본 섹션이 있습니다.

- 노트북 인터페이스(기본 패널) 및 도구 모음
- 작업 편집 탭

노트북 편집기

AWS Glue Studio 노트북 편집기는 Jupyter Notebook 애플리케이션을 기반으로 합니다. AWS Glue Studio 노트북 인터페이스는 노트북 사용자 인터페이스 섹션에 설명된, Juypter Notebook에서 제공하는 인터페이스와 비슷합니다. 대화형 세션에서 사용하는 노트북은 Jupyter Notebook입니다.

AWS Glue Studio 노트북은 Juptyer 노트북과 비슷하지만 다음과 같은 몇 가지 주요 차이점이 있습니다.

- 현재 AWS Glue Studio 노트북은 확장을 설치할 수 없습니다.
- 여러 탭을 사용할 수 없습니다. 작업과 노트북 간에 1:1 관계입니다.
- AWS Glue Studio 노트북에는 Jupyter Notebook에 있는 것과 같은 상단 파일 메뉴가 없습니다.
- 현재 AWS Glue Studio 노트북은 AWS Glue 커널에서만 실행됩니다. 커널은 직접 업데이트할 수 없습니다.

AWS Glue Studio 작업 편집 탭

ETL 작업을 조작하는 데 사용하는 탭은 노트북 페이지 상단에 있습니다. AWS Glue Studio의 시각적 작업 편집기에 표시되는 탭과 비슷하며, 동일한 작업을 수행합니다.

- 노트북(Notebook) - 노트북 인터페이스를 사용하여 작업 스크립트를 보려면 이 탭을 사용합니다.
- 작업 세부 정보(Job details) - 작업 실행의 환경과 속성을 구성합니다.
- 실행(Runs) - 이 작업의 이전 실행 정보를 봅니다.
- 일정(Schedules) - 특정 시간에 작업 실행 일정을 구성합니다.

노트북 및 작업 스크립트 저장

생성 중인 작업 스크립트와 노트북을 언제든지 저장할 수 있습니다. 시각적 편집기나 스크립트 편집기를 사용하는 경우와 마찬가지로, 오른쪽 위에 있는 저장(Save) 버튼을 선택하면 됩니다.

저장을 선택하면 노트북 파일이 기본 위치에 저장됩니다.

- 기본적으로 작업 스크립트는 작업 세부 정보 탭의 고급 속성 아래 작업 세부 정보 속성 스크립트 경로에 표시된 Amazon S3 위치에 저장됩니다. 작업 스크립트는 `Scripts`라는 하위 폴더에 저장됩니다.
- 기본적으로 노트북 파일(`.ipynb`)은 작업 세부 정보 탭의 고급 속성 아래 작업 세부 정보 스크립트 경로에 표시된 Amazon S3 위치에 저장됩니다. 노트북 파일은 `Notebooks`라는 하위 폴더에 저장됩니다.

> **Note**
>
> 작업을 저장하면 작업 스크립트에는 노트북의 코드 셀만 포함됩니다. 마크다운 셀 및 매직은 작업 스크립트에 포함되지 않습니다. 그러나 `.ipynb` 파일에는 마크다운과 매직이 포함됩니다.

작업을 저장한 후 노트북에서 생성한 스크립트를 사용하여 작업을 실행할 수 있습니다.

노트북 세션 관리

AWS Glue Studio의 노트북은 AWS Glue의 대화형 세션 기능을 기반으로 합니다. 대화형 세션을 사용하면 비용이 발생합니다. 비용을 관리하기 위해 계정에 대해 생성된 세션을 모니터링하고 모든 세션의 기본 설정을 구성할 수 있습니다.

모든 노트북 세션의 기본 시간 제한 변경

기본적으로 프로비저닝된 AWS Glue Studio 노트북은 노트북이 시작되고 셀이 실행되지 않은 경우 12시간 후에 시간 초과됩니다. 관련된 비용이 없으며 제한 시간을 구성할 수 없습니다.

셀을 실행하면 대화형 세션이 시작됩니다. 이 세션의 기본 제한 시간은 48시간입니다. 셀을 실행하기 전에 `%idle_timeout` 매직을 전달하여 이 제한 시간을 구성할 수 있습니다.

AWS Glue Studio에서 노트북의 기본 세션 시간 제한을 수정하려면

1. 노트북의 셀에 `%idle_timeout` 매직을 입력하고 시간 제한 값을 분 단위로 지정합니다.
2. 예: `%idle_timeout 15`는 기본 시간 제한을 15분으로 변경합니다. 15분 내에 세션을 사용하지 않으면 세션이 자동으로 중지됩니다.

추가 Python 모듈 설치

pip를 사용하여 세션에 추가 모듈을 설치하려는 경우 `%additional_python_modules`를 사용하여 세션에 추가하면 됩니다.

```
%additional_python_modules awswrangler, s3://amzn-s3-demo-bucket/mymodule.whl
```

additional_python_modules에 대한 모든 인수가 `pip3 install -m <>`에 전달됩니다.

사용 가능한 Python 모듈 목록은 AWS Glue에서 Python 라이브러리 사용을 참조하세요.

AWS Glue 구성 변경

매직을 사용하여 AWS Glue 작업 구성 값을 제어할 수 있습니다. 작업 구성 값을 변경하려면 노트북에서 적절한 매직을 사용해야 합니다. Jupyter용 AWS Glue 대화형 세션에서 지원되는 매직을 참조하세요.

> **Note**
>
> 실행 중인 세션의 속성 재정의는 더 이상 사용할 수 없습니다. 세션 구성을 변경하려면 세션을 중지하고 새 구성을 설정한 후 새 세션을 시작하면 됩니다.

AWS Glue는 다양한 작업자 유형을 지원합니다. `%worker_type`을 사용하여 작업자 유형을 설정할 수 있습니다. 예를 들면 `%worker_type G.2X` 입니다. 기본값은 G.1X입니다.

`%number_of_workers`를 사용하여 작업자 수도 지정할 수 있습니다. 예를 들어 40개의 작업자를 지정하려면 `%number_of_workers 40`을 사용합니다.

자세한 내용은 작업 속성 수정을 참조하세요.

노트북 세션 중지

노트북 세션을 중지하려면 `%stop_session` 매직을 사용합니다.

AWS 콘솔에서 노트북을 나가는 경우 경고 메시지가 표시되며, 여기서 세션 중지를 선택할 수 있습니다.

Amazon Q Developer를 AWS Glue Studio 노트북과 함께 사용하기

AWS Glue Studio를 사용하면 Jupyter Notebook 기반의 노트북 인터페이스에서 작업을 대화형으로 작성할 수 있습니다. Amazon Q Developer를 사용하면 AWS Glue Studio 노트북에서 작성 경험이 향상됩니다.

Amazon Q Developer 확장 프로그램은 코드 권장 사항을 생성하고 코드 문제와 관련된 개선 사항을 제안하여 코드 작성을 지원합니다. Amazon Q Developer는 AWS Glue Studio 노트북에서 Spark 작업을 위한 ETL 스크립트를 코딩하는 데 사용되는 두 가지 언어인 Python과 Scala를 모두 지원합니다.

Amazon Q Developer란 무엇인가요?

Amazon Q Developer는 기계 학습에 기반하여 개발자의 생산성을 높일 수 있는 서비스입니다. Amazon Q Developer는 IDE에서 자연어로 된 개발자의 주석과 코드를 기반으로 코드 권장 사항을 생성하여 이를 지원합니다. 이 서비스는 JupyterLab, Amazon SageMaker AI Studio, Amazon SageMaker AI 노트북 인스턴스 및 기타 통합 개발 환경(IDE)과 통합됩니다.

자세한 내용은 AWS Glue Studio을(를) 통한 Amazon Q Developer 사용을 참조하세요.

콘솔의 AWS Glue 작업 실행 상태

AWS Glue 추출, 변환, 로드 작업이 실행되는 동안 또는 중지된 후 작업 상태를 볼 수 있습니다. AWS Glue 콘솔을 사용하여 상태를 볼 수 있습니다. 작업 실행 상태에 대한 자세한 내용은 the section called “작업 실행 상태” 섹션을 참조하십시오.

작업 모니터링 대시보드에 액세스

AWS Glue 탐색 창에서 ETL 작업 아래 작업 실행 모니터링 링크를 선택하여 작업 모니터링 대시보드에 액세스합니다.

작업 모니터링 대시보드 개요

작업 모니터링 대시보드는 [실행 중(Running)], [취소됨(Canceled)], [성공(Success)] 또는 [실패(Failed)] 상태의 작업에 대한 합계와 함께 작업 실행에 대한 전체 요약을 제공합니다. 추가 타일은 전체 작업 실행 성공률, 작업에 대한 예상 DPU 사용량, 작업 유형, 작업자 유형 및 일별 작업 상태 개수 분석을 제공합니다.

타일의 그래프는 대화형입니다. 그래프에서 블록을 선택하여 페이지 하단의 [작업 실행(Job runs)] 테이블에 해당 작업만 표시하는 필터를 실행할 수 있습니다.

[날짜 범위(Date range)] 선택기를 사용하여 이 페이지에 표시되는 정보의 날짜 범위를 변경할 수 있습니다. 날짜 범위를 변경하면 정보 타일이 조정되어 현재 날짜 이전의 지정된 일 수에 대한 값을 표시합니다. 날짜 범위 선택기에서 [사용자 정의(Custom)]를 선택하면 특정 날짜 범위를 사용할 수도 있습니다.

작업 실행 보기

> **Note**
>
> 워크플로 및 작업 실행의 경우 작업 실행 기록을 90일 동안 액세스할 수 있습니다.

[작업 실행(Job runs)] 리소스 목록에는 지정된 날짜 범위 및 필터에 대한 작업이 표시됩니다.

상태, 작업자 유형, 작업 유형 및 작업 이름과 같은 추가 기준에 따라 작업을 필터링할 수 있습니다. 테이블 상단의 필터 상자에 필터로 사용할 텍스트를 입력할 수 있습니다. 텍스트를 입력할 때 일치하는 텍스트가 포함된 행으로 테이블 결과가 업데이트됩니다.

작업 모니터링 대시보드의 그래프에서 요소를 선택하여 작업의 하위 집합을 볼 수 있습니다. 예를 들어 [작업 실행 요약(Job runs summary)] 타일에서 실행 중인 작업 수를 선택하면 [작업 실행(Job runs)] 목록에 현재 상태가 `Running`인 작업만 표시됩니다. [작업자 유형 분석(Worker type breakdown)] 막대 차트에서 막대 중 하나를 선택하면 작업자 유형 및 상태가 일치하는 작업 실행만 [작업 실행(Job runs)] 목록에 표시됩니다.

[작업 실행(Job runs)] 리소스 목록에는 작업 실행에 대한 세부 정보가 표시됩니다. 열 머리글을 선택하여 테이블의 행을 정렬할 수 있습니다. 표에는 다음 정보가 포함되어 있습니다.

속성	설명
작업 이름	작업의 이름입니다.
유형	작업 환경의 유형입니다. • [Glue ETL]: AWS Glue에서 관리하는 Apache Spark 환경에서 실행됩니다. • [Glue 스트리밍(Glue Streaming)]: Apache Spark 환경에서 실행되고 데이터 스트림에서 ETL을 수행합니다. • Python 셸: Python 스크립트를 쉘로 실행합니다.
시작 시간	이 작업이 시작된 날짜 및 시간.
종료 시간	이 작업 실행이 완료된 날짜 및 시간입니다.
실행 상태	작업 실행의 현재 상태입니다. 값은 다음과 같습니다. • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`

속성	설명
런타임	작업이 리소스를 소비한 시간입니다.
Capacity	이 작업 실행에 할당된 AWS Glue 데이터 처리 장치(DPU) 수입니다. 용량 계획에 대한 자세한 내용은 AWS Glue Developer Guide의 Monitoring for DPU Capacity Planning을 참조하세요.

속성	설명
작업자 유형	작업 실행 시 할당된 미리 정의된 작업자 유형입니다. 값은 `G.1X`, `G.2X`, `G.4X` 또는 `G.8X`일 수 있습니다. • **`G.1X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 84GB 디스크(약 34GB의 여유 공간)에서 1개의 DPU(vCPU 4개, 메모리 16GB)에 매핑됩니다. 메모리 집약적인 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다. AWS Glue 버전 2.0 이상 작업의 기본 [작업자 유형(Worker type)]입니다. • **`G.2X`** – 이 유형을 선택할 경우 [작업자 수(Number of workers)] 값도 제공합니다. 각 작업자는 128GB 디스크(약 77GB의 여유 공간)에서 2개의 DPU(vCPU 8개, 메모리 32GB)에 매핑됩니다. 메모리 집약적인 작업과 기계 학습 변환을 실행하는 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다. • **`G.4X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 256GB 디스크(약 235GB의 여유 공간)에서 4개의 DPU(vCPU 16개, 메모리 64GB)에 매핑됩니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(스톡홀름)과 같은AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.

속성	설명
	• **`G.8X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 512GB 디스크(약 487GB의 여유 공간)에서 8개의 DPU(vCPU 32개, 메모리 128GB)에 매핑됩니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 `G.4X` 작업자 유형에 지원되는 동일한 AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
DPU 시간	작업 실행에 사용된 예상 DPU입니다. DPU는 처리 능력의 상대적인 측정치입니다. DPU는 작업 실행 비용을 결정하는 데 사용됩니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하십시오.

목록에서 원하는 작업 실행을 선택하고 추가 정보를 볼 수 있습니다. 작업 실행을 선택하고 다음 중 하나를 수행합니다.

- [작업(Actions)] 메뉴와 [작업 보기(View job)] 옵션을 선택하여 시각적 편집기에서 작업을 봅니다.
- [작업(Actions)] 메뉴와 [실행 중지(Stop run)] 옵션을 선택하여 작업의 현재 실행을 중지합니다.
- [CloudWatch Logs 보기(View CloudWatch logs)] 버튼을 선택하여 해당 작업에 대한 작업 실행 로그를 봅니다.
- 세부 정보 보기를 선택하여 작업 실행 세부 정보 페이지를 봅니다.

작업 실행 로그 보기

다양한 방법으로 작업 로그를 볼 수 있습니다.

- [모니터링(Monitoring)] 페이지의 [작업 실행(Job runs)] 테이블에서 작업 실행을 선택한 다음 [CloudWatch Logs 보기(View CloudWatch logs)]를 선택합니다.
- 시각적 작업 편집기의 작업에 대한 [실행(Runs)] 탭에서 하이퍼링크를 선택하여 로그를 봅니다.

- [로그(Logs)] - 작업 실행에 연속 로깅이 사용될 때 작성된 Apache Spark 작업 로그에 대한 링크입니다. 이 링크를 선택하면 `/aws-glue/jobs/logs-v2` 로그 그룹의 Amazon CloudWatch 로그로 이동합니다. 기본적으로 로그는 유용하지 않은 Apache Hadoop YARN 하트비트 및 Apache Spark 드라이버 또는 실행기 로그 메시지를 제외합니다. 연속 로깅에 대한 자세한 내용은 AWS Glue Developer Guide의 Continuous Logging for AWS Glue Jobs를 참조하세요.
- [오류 로그(Error logs)] - 이 작업을 실행할 때 `stderr`에 작성되는 로그와 연결됩니다. 이 링크를 선택하면 `/aws-glue/jobs/error` 로그 그룹의 Amazon CloudWatch 로그로 이동합니다. 이 로그를 사용하여 작업 실행 중에 발생한 오류에 대한 세부 정보를 볼 수 있습니다.
- [출력 로그(Output logs)] - 이 작업을 실행할 때 `stdout`에 작성되는 로그와 연결됩니다. 이 링크를 선택하면 `/aws-glue/jobs/output` 로그 그룹의 Amazon CloudWatch 로그로 이동합니다. 이러한 로그를 사용하여 AWS Glue Data Catalog에서 생성된 테이블 및 발생한 오류에 대한 모든 세부 정보를 볼 수 있습니다.

작업 실행의 세부 정보 보기

[모니터링(Monitoring)] 페이지의 [작업 실행(Job runs)] 목록에서 작업을 선택한 다음 [실행 세부 정보 보기(View run details)]를 선택하여 해당 작업 실행에 대한 세부 정보를 볼 수 있습니다.

작업 실행 세부 정보 페이지에 표시되는 정보는 다음과 같습니다.

속성	설명
작업 이름	작업의 이름입니다.
실행 상태	작업 실행의 현재 상태입니다. 값은 다음과 같습니다. • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`
Glue 버전	작업 실행에 사용된 AWS Glue 버전입니다.

속성	설명
최근 시도	이 작업 실행에 대한 자동 재시도 횟수입니다.
시작 시간	이 작업이 시작된 날짜 및 시간.
종료 시간	이 작업 실행이 완료된 날짜 및 시간입니다.
시작 시간	작업 실행 준비에 소요된 시간입니다.
실행 시간	작업 스크립트 실행에 소요된 시간입니다.
트리거 이름	작업과 연결된 트리거의 이름입니다.
최근 수정 시간	작업이 마지막으로 수정된 날짜입니다.
보안 구성	Amazon S3 암호화, CloudWatch 암호화 및 작업 북마크 암호화 설정을 포함하는 작업에 대한 보안 구성입니다.
제한 시간	작업 실행 시간 제한 임곗값입니다.
할당된 용량	이 작업 실행에 할당된 AWS Glue 데이터 처리 장치(DPU) 수입니다. 용량 계획에 대한 자세한 내용은 AWS Glue Developer Guide의 Monitoring for DPU Capacity Planning을 참조하세요.
최대 용량	작업 실행에 사용할 수 있는 최대 용량.
작업자 수	작업 실행에 사용된 작업자 수입니다.

속성	설명
작업자 유형	작업 실행에 할당된 사전 정의된 작업자 유형. 값은 `G.1X` 또는 `G.2X`일 수 있습니다. • **`G.1X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 1개의 DPU(vCPU 4개, 16GB 메모리, 64GB 디스크)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 메모리 집약적인 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다. AWS Glue 버전 2.0 이상 작업의 기본 [작업자 유형(Worker type)]입니다. • **`G.2X`** – 이 유형을 선택할 경우 [작업자 수(Number of workers)] 값도 제공합니다. 각 작업자는 2개의 DPU(vCPU 8개, 32GB 메모리, 128GB 디스크)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 메모리 집약적인 작업과 기계 학습 변환을 실행하는 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다.
로그	연속 로깅을 위한 작업 로그에 대한 링크(`/aws-glue/jobs/logs-v2`)입니다.
출력 로그	작업 출력 로그 파일(`/aws-glue/jobs/output`)에 대한 링크입니다.
오류 로그	작업 오류 로그 파일(`/aws-glue/jobs/error`)에 대한 링크입니다.

최근 작업 실행에 대한 정보를 볼 때 제공되는 다음의 추가 항목도 볼 수 있습니다. 자세한 내용은 the section called “최근 작업 실행에 대한 정보 보기” 단원을 참조하십시오.

- 입력 인수
- 연속 로그
- 지표 - 기본 지표의 시각화를 볼 수 있습니다. 포함된 지표에 대한 자세한 내용은 the section called “Spark 작업 실행에 대한 Amazon CloudWatch 지표 보기” 섹션을 참조하세요.

- Spark UI - Spark UI에서 작업에 대한 Spark 로그를 시각화할 수 있습니다. Spark Web UI에 관해 자세한 내용은 the section called “Spark UI로 모니터링”를 참조하십시오. the section called “작업을 위한 Spark UI 사용 설정”의 절차에 따라 이 기능을 활성화합니다.

Spark 작업 실행에 대한 Amazon CloudWatch 지표 보기

작업 실행에 대한 세부 정보 페이지의 실행 세부 정보(Run details) 섹션 아래에서 작업 지표를 볼 수 있습니다. AWS Glue Studio는 모든 작업 실행의 작업 지표를 Amazon CloudWatch로 보냅니다.

AWS Glue은 30초마다 Amazon CloudWatch에 지표를 보고합니다. AWS Glue 지표는 이전에 보고한 값의 델타 값을 나타냅니다. 적절한 경우 지표 대시보드는 30초 값을 집계(합)하여 마지막 1분 전체에 대한 값을 얻습니다. 그러나 AWS Glue가 Amazon CloudWatch에 전달하는 Apache Spark 지표는 일반적으로 보고되는 시점의 현재 상태를 나타내는 절대값입니다.

> **Note**
>
> Amazon CloudWatch에 액세스하려면 계정을 구성해야 합니다.

지표는 다음과 같은 작업 실행에 대한 정보를 제공합니다.

- [ETL 데이터 이동(ETL Data Movement)] – Amazon S3에서 읽거나 쓴 바이트 수입니다.
- [메모리 프로파일: 사용된 힙(Memory Profile: Heap used)] – Java 가상 머신(JVM) 힙에서 사용하는 메모리 바이트 수입니다.
- [메모리 프로파일: 힙 사용량(Memory Profile: heap usage)] – JVM 힙에서 사용하는 메모리 부분(규모: 0~1, 백분율로 표시)입니다.
- [CPU 로드(CPU Load)] – 사용된 CPU 시스템 로드 부분(규모: 0~1, 백분율로 표시)입니다.

Ray 작업 실행에 대한 Amazon CloudWatch 지표 보기

작업 실행에 대한 세부 정보 페이지의 실행 세부 정보(Run details) 섹션 아래에서 작업 지표를 볼 수 있습니다. AWS Glue Studio는 모든 작업 실행의 작업 지표를 Amazon CloudWatch로 보냅니다.

AWS Glue은 30초마다 Amazon CloudWatch에 지표를 보고합니다. AWS Glue 지표는 이전에 보고한 값의 델타 값을 나타냅니다. 적절한 경우 지표 대시보드는 30초 값을 집계(합)하여 마지막 1분 전체에 대한 값을 얻습니다. 그러나 AWS Glue가 Amazon CloudWatch에 전달하는 Apache Spark 지표는 일반적으로 보고되는 시점의 현재 상태를 나타내는 절대값입니다.

> **Note**
>
> 다음 사항에 설명된 대로 Amazon CloudWatch에 액세스하려면 계정을 구성해야 합니다.

Ray 작업에서는 다음과 같은 집계된 지표 그래프를 볼 수 있습니다. 이를 통해 클러스터 및 작업의 프로파일을 구축하고 각 노드에 대한 세부 정보에 액세스할 수 있습니다. 이 그래프를 지원하는 시계열 데이터는 추가 분석을 위해 CloudWatch에서 사용할 수 있습니다.

작업 프로파일: 작업 상태

시스템에 있는 Ray 작업 수를 표시합니다. 각 작업 수명 주기에는 고유한 시계열이 지정됩니다.

작업 프로파일: 작업 이름

시스템에 있는 Ray 작업 수를 표시합니다. 대기 중인 작업과 진행 중인 작업만 표시됩니다. 각 작업 유형 (이름 기준)에는 고유한 시계열이 지정됩니다.

클러스터 프로파일: 사용 중인 CPU

사용된 CPU 코어 수를 표시합니다. 각 노드에는 고유한 시계열이 지정됩니다. 노드는 임시 IP 주소로 식별되며, IP 주소는 임시 주소이며 식별용으로만 사용됩니다.

클러스터 프로파일: 객체 스토어 메모리 사용

Ray 객체 캐시의 메모리 사용을 보여줍니다. 각 메모리 위치(물리적 메모리, 디스크에 캐시됨, Amazon S3에서 유출됨)에는 고유한 시계열이 지정됩니다. 객체 스토어는 클러스터의 모든 노드에서 데이터 스토리지를 관리합니다. 자세한 내용은 Ray 설명서의 Objects를 참조하세요.

클러스터 프로파일: 노드 수

클러스터에 프로비저닝된 노드 수를 표시합니다.

노드 세부 정보: CPU 사용

각 노드에서 CPU 사용률을 백분율로 표시합니다. 각 시리즈는 노드에 있는 모든 코어에서 집계된 CPU 사용률을 보여줍니다.

노드 세부 정보: 메모리 사용

각 노드에서 메모리 사용(GB)을 표시합니다. 각 시리즈는 Ray 작업 및 Plasma 저장 프로세스를 포함하여 노드의 모든 프로세스 사이에서 집계된 메모리를 보여줍니다. 디스크에 저장되거나 Amazon S3로 유출된 객체는 반영되지 않습니다.

노드 세부 정보: 디스크 사용

각 노드에서 디스크 사용(GB)을 표시합니다.

노드 세부 정보: 디스크 I/O 속도

각 노드에서 디스크 I/O(KB/s)를 표시합니다.

노드 세부 정보: 네트워크 I/O 처리량

각 노드에서 네트워크 I/O(KB/s)를 표시합니다.

노드 세부 정보: Ray 구성 요소별 CPU 사용

CPU 사용을 코어의 비율로 표시합니다. 각 노드의 각 Ray 구성 요소에 고유한 시계열이 지정됩니다.

노드 세부 정보: Ray 구성 요소별 메모리 사용

메모리 사용(GiB)을 표시합니다. 각 노드의 각 Ray 구성 요소에 고유한 시계열이 지정됩니다.

민감한 데이터 감지 및 처리

PII 탐지 변환은 데이터 원본에서 개인 식별 정보(PII)를 식별합니다. 식별할 PII 엔터티, 데이터를 스캔하는 방법, PII 탐지 변환에서 식별된 PII 엔터티로 수행할 작업을 선택합니다.

PII 탐지 변환은 사용자가 정의하거나 AWS에서 사전 정의한 엔터티를 탐지, 마스크 또는 제거하는 기능을 제공합니다. 이를 통해 규정 준수 수준을 높이고 책임 부담을 줄일 수 있습니다. 예를 들어, 읽을 수 있는 데이터에 개인 식별 정보가 포함되지 않도록 하고 xxx-xx-xxxx와 같은 고정 문자열 형식의 사회보장번호, 전화번호 또는 주소를 마스킹하고 싶을 수 있습니다.

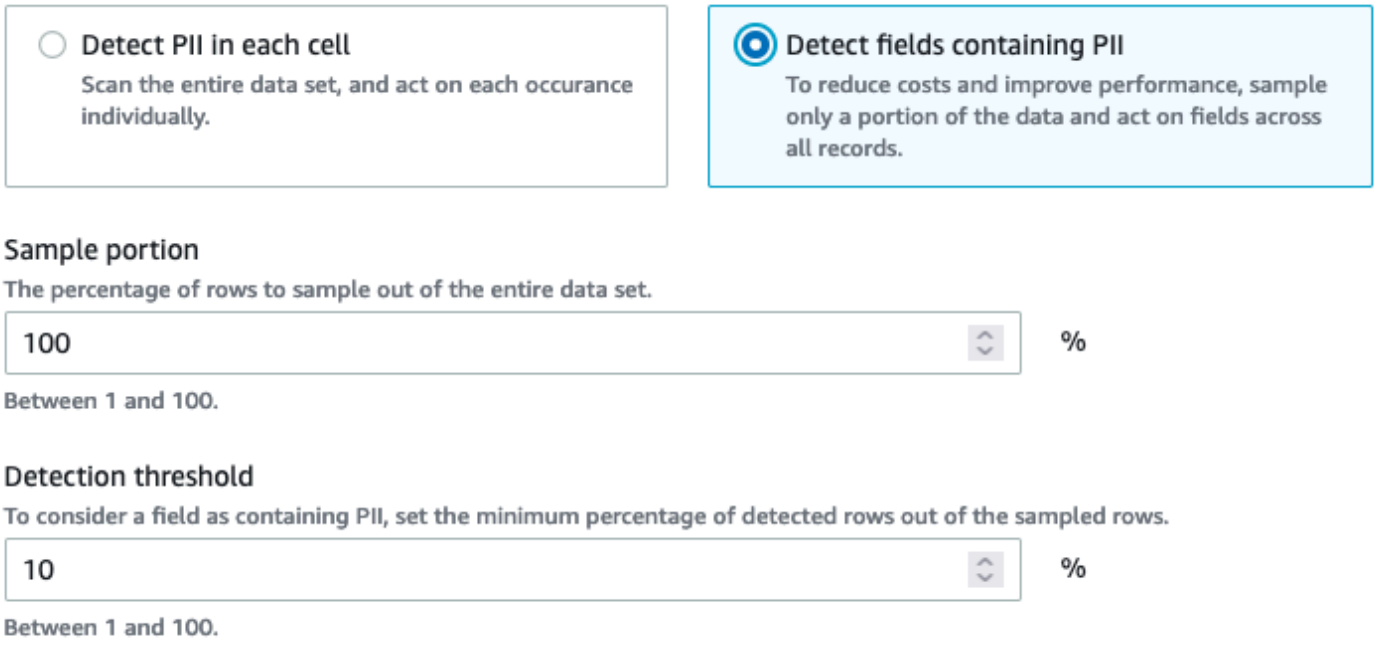
AWS Glue Studio 외부에서 민감한 데이터를 처리하려면 AWS Glue Studio 외부에서 민감한 데이터 감지 사용를 참조하십시오

주제

- 데이터를 스캔하는 방법 선택
- 탐지할 PII 엔터티 선택
- 탐지 민감도 수준 지정
- 식별된 PII 데이터로 수행할 작업 선택
- 세분화된 작업 오버라이드 추가

데이터를 스캔하는 방법 선택

개인 식별 정보(PII)와 같은 민감한 데이터가 있는지 데이터 세트를 스캔할 때 각 행에서 PII를 탐지할지 아니면 PII 데이터가 포함된 열을 검색할지 선택할 수 있습니다.



각 셀에서 PII 탐지(Detect PII in each cell)를 선택할 경우 데이터 원본의 모든 행을 스캔하도록 선택하는 것입니다. PII 엔터티를 식별하기 위한 포괄적인 스캔입니다.

PII가 포함된 필드 탐지(Detect fields containing PII)를 선택할 경우 행 샘플에서 PII 엔터티를 스캔하도록 선택하는 것입니다. PII 엔터티가 있는 필드도 식별하면서 비용과 리소스를 낮게 유지하는 방법입니다.

PII가 포함된 필드를 탐지하도록 선택할 경우 행의 일부를 샘플링하여 비용을 절감하고 성능을 향상시킬 수 있습니다. 이 옵션을 선택하면 다음과 같은 추가 옵션을 지정할 수 있습니다.

- 부분 샘플링(Sample portion): 샘플링할 행의 백분율을 지정할 수 있습니다. 예를 들어 '50'을 입력할 경우 50%의 행에서 PII 엔터티를 스캔하도록 지정하는 것입니다.
- 탐지 임곗값(Detection threshold): 전체 열이 PII 엔터티를 포함하는 것으로 식별되기 위해 PII 엔터티를 포함해야 하는 행의 백분율을 지정할 수 있습니다. 예를 들어 '10'을 입력할 경우 필드가 PII 엔터티(미국 전화)를 포함하는 것으로 식별되려면 스캔된 행의 PII 엔터티(미국 전화) 수가 10% 이상이어야 한다고 지정하는 것입니다. PII 엔터티를 포함하는 행의 백분율이 10% 미만일 경우 해당 필드는 PII 엔터티(미국 전화)를 포함하는 것으로 레이블이 지정되지 않습니다.

탐지할 PII 엔터티 선택

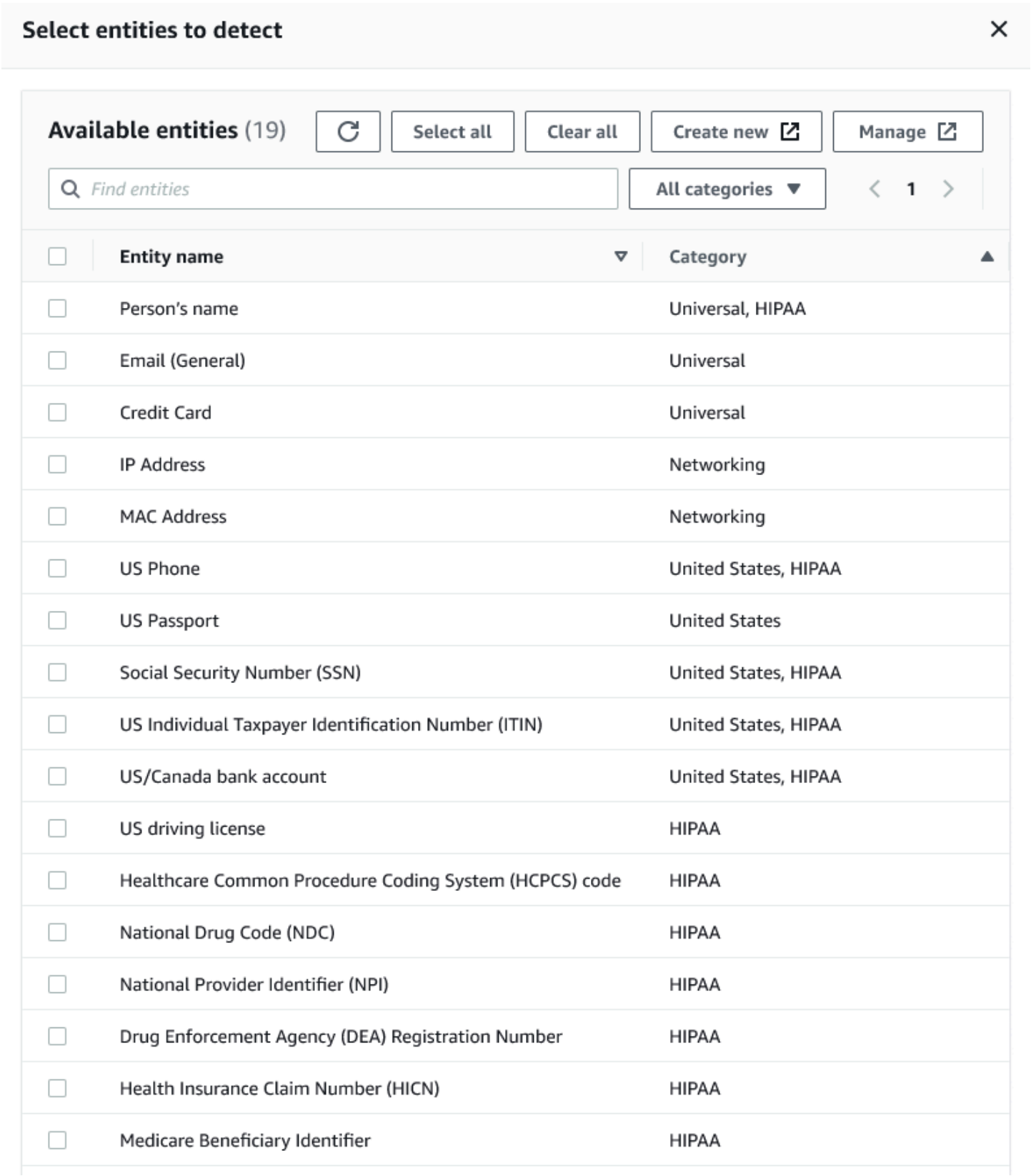
각 셀에서 PII 탐지(Detect PII in each cell)를 선택한 경우 다음 세 옵션 중 하나를 선택할 수 있습니다.

- 사용 가능한 모든 PII 패턴 - 여기에는 AWS 엔터티가 포함됩니다.
- 카테고리 선택 - 범주를 선택하면 PII 패턴이 선택한 범주의 패턴을 자동으로 포함합니다.
- 특정 패턴 선택(Select specific patterns) - 선택한 패턴만 탐지됩니다.

민감한 관리형 데이터 형식의 전체 목록은 관리형 데이터 형식을 참조하세요.

사용 가능한 모든 PII 패턴 중에서 선택

사용 가능한 모든 PII 패턴(All available PII patterns)을 선택한 경우 AWS에서 사전 정의한 엔터티를 선택합니다. 엔터티를 하나 이상 또는 모두 선택할 수 있습니다.


Select entities to detect
Available entities (19)
Select all
Clear all
Create new
Manage
Find entities
All categories
1
Entity name
Category
Person's name
Universal, HIPAA
Email (General)
Universal
Credit Card
Universal
IP Address
Networking
MAC Address
Networking
US Phone
United States, HIPAA
US Passport
United States
Social Security Number (SSN)
United States, HIPAA
US Individual Taxpayer Identification Number (ITIN)
United States, HIPAA
US/Canada bank account
United States, HIPAA
US driving license
HIPAA
Healthcare Common Procedure Coding System (HCPCS) code
HIPAA
National Drug Code (NDC)
HIPAA
National Provider Identifier (NPI)
HIPAA
Drug Enforcement Agency (DEA) Registration Number
HIPAA
Health Insurance Claim Number (HICN)
HIPAA
Medicare Beneficiary Identifier
HIPAA

카테고리 선택

탐지할 PII 패턴으로 카테고리 선택(Select categories)을 선택한 경우 드롭다운 메뉴의 옵션에서 선택할 수 있습니다. 일부 엔터티는 둘 이상의 카테고리에 속할 수 있습니다. 예를 들어, 사람 이름(Person's name)은 일반(Universal) 및 HIPAA 카테고리에 속하는 엔터티입니다.

- 일반(예: 이메일, 신용카드)
- HIPAA(예: 미국 운전 면허증, HCPCS(Healthcare Common Procedure Coding System) 코드)
- 네트워킹(예: IP 주소, MAC 주소)
- 아르헨티나
- 호주
- 오스트리아
- 벨기에
- 보스니아
- 불가리아
- 캐나다
- 칠레
- 콜롬비아
- 크로아티아
- 사이프러스
- 체코
- 덴마크
- 에스토니아
- 핀란드
- 프랑스
- 독일
- 그리스
- 헝가리
- 아일랜드
- 한국
- 일본

- 멕시코
- 네덜란드
- 뉴질랜드
- 노르웨이
- 포르투갈
- 루마니아
- 싱가포르
- 슬로바키아
- 슬로베니아
- 스페인
- 스웨덴
- 스위스
- 터키
- 우크라이나
- 미국
- 영국
- 베네수엘라

특정 패턴 선택

탐지할 PII 패턴으로 특정 패턴 선택(Select specific patterns)을 선택한 경우 이미 만든 패턴 목록에서 검색하거나 찾아볼 수 있으며, 새 탐지 엔터티 패턴을 만들 수도 있습니다.

아래 단계에서는 민감한 데이터를 감지하기 위한 새로운 사용자 정의 패턴을 생성하는 방법을 설명합니다. 사용자 지정 패턴의 이름을 입력하여 사용자 지정 패턴을 생성하고 정규식을 추가하고 선택 사항으로 컨텍스트 단어를 정의합니다.

1. 새 패턴을 생성하려면 새로 생성(Create new) 버튼을 클릭합니다.

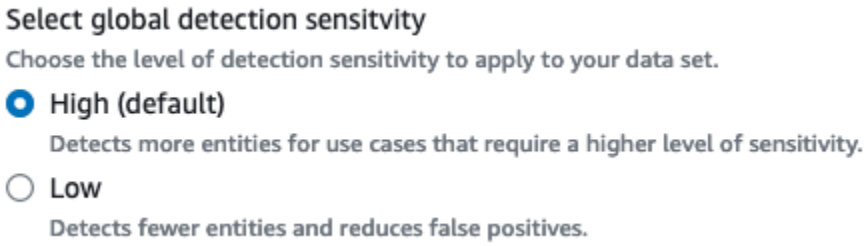
2. 탐지 엔터티 생성 페이지에서 엔터티 이름과 정규 표현식을 입력합니다. 정규 표현식(Regex)은 AWS Glue가 엔터티를 일치시키는 데 사용됩니다.
3. 검증(Validate)을 클릭합니다. 검증에 성공하면 문자열이 유효한 정규 표현식이라는 확인 메시지가 표시됩니다. 검증에 실패하면 문자열이 적절한 형식 및 허용된 문자 리터럴, 연산자 또는 구조를 준수하지 않는다는 메시지가 표시됩니다.
4. 정규 표현식 외에 컨텍스트 단어를 추가하도록 선택할 수 있습니다. 컨텍스트 단어는 일치 가능성을 높일 수 있습니다. 필드 이름이 엔터티를 설명하지 않는 경우에 유용할 수 있습니다. 예를 들어, 사회보장번호의 이름은 'SSN' 또는 'SS'일 수 있습니다. 이러한 컨텍스트 단어를 추가하면 엔터티를 일치시키는 데 도움이 될 수 있습니다.
5. 생성(Create)을 클릭하여 탐지 엔터티를 생성합니다. 생성된 모든 엔터티는 AWS Glue Studio 콘솔에 표시됩니다. 왼쪽 탐색 메뉴에서 탐지 엔터티(Detection entities)를 클릭합니다.

   탐지 엔터티(Detection entities) 페이지에서 탐지 엔터티를 편집, 삭제 또는 생성할 수 있습니다. 검색 필드를 사용하여 패턴을 검색할 수도 있습니다.

탐지 민감도 수준 지정

민감한 데이터 탐지를 사용할 때 민감도 수준을 설정할 수 있습니다.

- 높음 - (기본값) 더 높은 수준의 민감도가 필요한 사용 사례에서 더 많은 개체를 탐지합니다. 2023년 11월 이후에 생성된 모든 AWS Glue 작업에는 이 설정이 자동으로 적용됩니다.
- 낮음 - 탐지되는 개체 수를 줄이고 오탐을 줄입니다.

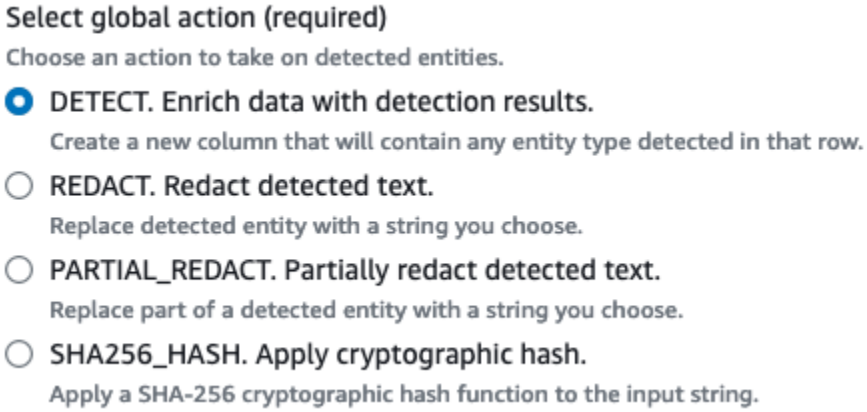


식별된 PII 데이터로 수행할 작업 선택

전체 데이터 소스에서 PII를 탐지하도록 선택한 경우 적용할 글로벌 옵션을 선택할 수 있습니다.

- 탐지 결과로 데이터 보강(Enrich data with detection results): 각 셀에서 PII 탐지(Detect PII in each cell)를 선택한 경우 탐지된 엔터티를 새 열에 저장할 수 있습니다.

- 탐지된 텍스트 교정(Redact detected text): 탐지된 PII 값을 선택적 바꾸기(Replacing) 텍스트 입력 필드에 지정한 문자열로 바꿀 수 있습니다. 문자열을 지정하지 않으면 탐지된 PII 엔터티가 '*******'로 바뀝니다.
- 탐지된 텍스트 부분 교정: 탐지된 PII 값을 지정한 문자열로 바꿀 수 있습니다. 두 가지 옵션이 있습니다. 끝을 마스킹하지 않은 상태로 두거나 명시적인 정규식 패턴을 제공하여 마스킹하는 것입니다. 이 기능은 AWS Glue 2.0에서는 아직 사용할 수 없습니다.
- Apply cryptographic hash(암호화 해시 적용): 탐지된 PII 값을 SHA-256 암호화 해시 함수에 전달하고 이 값을 함수의 출력으로 바꿀 수 있습니다.



AWS Glue 버전 2.0과 3.0 이상의 차이점

AWS Glue 2.0 작업은 보조 열의 각 열에 대해 탐지된 PII 정보가 포함된 새 DataFrame을 반환합니다. 모든 수정 또는 해시 작업은 비주얼 탭의 AWS Glue 스크립트에서 볼 수 있습니다.

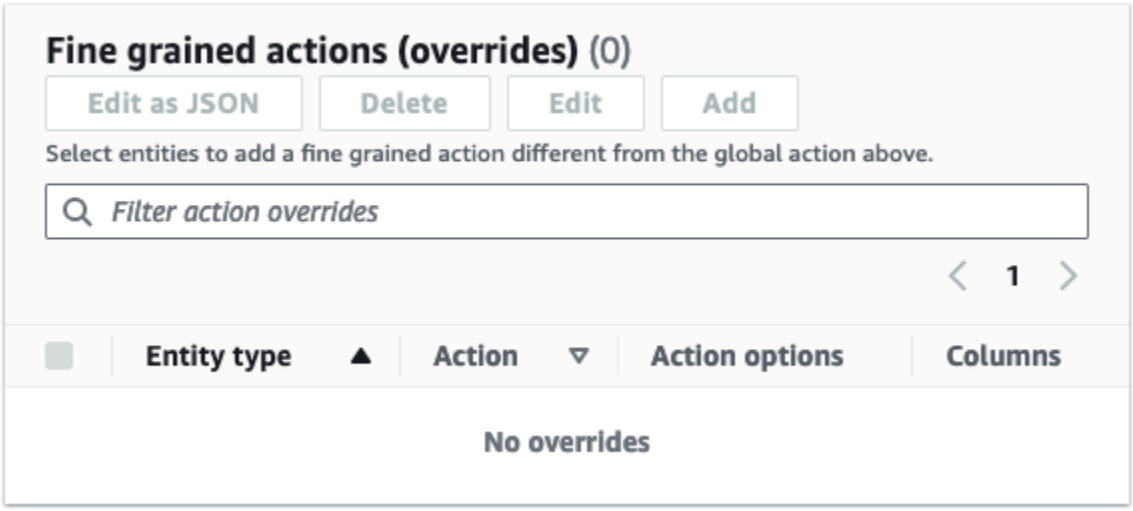
AWS Glue 3.0 및 4.0 작업은 동일한 추가 열이 포함된 새 DataFrame을 반환합니다. “actionUsed”의 새 키가 존재하며, 이 키는 `DETECT`, `REDACT`, `PARTIAL_REDACT`, `SHA256_HASH` 중 하나가 될 수 있습니다. 마스킹 동작을 선택하면 DataFrame은 민감한 데이터가 마스킹된 데이터를 반환합니다.

세분화된 작업 오버라이드 추가

세부 조치 재정의 테이블에 탐지 및 조치 설정을 추가할 수 있습니다. 다음 작업을 수행할 수 있습니다.

- 특정 열을 탐지에서 포함 또는 제외 - 데이터 소스에서 추론된 스키마가 테이블을 사용 가능한 열로 채웁니다.
- 글로벌 액션을 사용하는 것보다 더 세밀한 특정 설정을 지정 - 예를 들어 엔터티 유형별로 다른 수정 텍스트 설정을 지정할 수 있습니다.

- 글로벌 액션과 다른 액션 지정 - 다른 민감한 데이터 유형에 다른 액션을 적용하려는 경우 여기에서 수행할 수 있습니다. 동일한 열에 두 가지 다른 내부 편집 작업(수정 및 해싱)을 사용할 수는 없지만 감지는 항상 사용할 수 있습니다.



AWS Glue Studio를 사용하여 ETL 작업 관리

AWS Glue Studio의 간단한 그래픽 인터페이스를 사용하여 ETL 작업을 관리할 수 있습니다. 탐색 메뉴를 사용하여 [작업(Jobs)]을 선택하여 [작업(Jobs)] 페이지를 봅니다. 이 페이지에서 AWS Glue Studio 또는 AWS Glue 콘솔로 생성한 모든 작업을 볼 수 있습니다. 이 페이지에서 작업을 보고, 관리하고, 실행할 수 있습니다.

이 페이지에서 다음 태스크도 수행할 수 있습니다.

- 작업 실행 시작
- 작업 실행 예약
- 작업 일정 관리
- 작업 실행 중지
- 작업 보기
- 최근 작업 실행에 대한 정보 보기
- 작업 스크립트 보기
- 작업 속성 수정
- 작업 저장
- 작업 복제

- 작업 삭제

작업 실행 시작

AWS Glue Studio에서 온디맨드로 작업을 실행할 수 있습니다. 작업은 여러 번 실행할 수 있으며 작업을 실행할 때마다 AWS Glue는 작업 활동 및 성능에 대한 정보를 수집합니다. 이 정보를 작업 실행이라고 하며 작업 실행 ID로 식별됩니다.

다음과 같은 방법으로 AWS Glue Studio에서 작업 실행을 시작할 수 있습니다.

- [작업(Jobs)] 페이지에서 시작할 작업을 선택한 다음 [작업 실행(Run job)] 버튼을 선택합니다.
- 비주얼 편집기에서 작업을 보고 있고 작업이 저장된 경우 [실행(Run)] 버튼을 선택하여 작업 실행을 시작할 수 있습니다.

작업 실행에 대한 자세한 내용은 AWS Glue Developer Guide의 Working with Jobs on the AWS Glue Console을 참조하세요.

작업 실행 예약

AWS Glue Studio에서 특정 시간에 작업이 실행되도록 일정을 생성할 수 있습니다. 작업이 실행되는 횟수, 실행되는 요일 및 실행되는 시간과 같은 제약 조건을 지정할 수 있습니다. 이러한 제약 조건은 `cron`을 기준으로 하며 `cron`과 동일한 제약 조건을 갖습니다. 예를 들어, 매월 31일에 작업을 실행하고자 한다면 매월 31일이 없다는 점을 유의하기 바랍니다. `cron`에 대한 자세한 내용은 AWS Glue Developer Guide의 Cron Expressions를 참조하세요.

일정에 따라 작업을 실행하려면

1. 다음 방법 중 하나를 사용하여 작업 일정을 생성합니다.
   - [작업(Jobs)] 페이지에서 일정을 생성할 작업을 선택하고 [작업(Actions)]을 선택한 다음 [작업 예약(Schedule job)]을 선택합니다.
   - 시각적 편집기에서 작업을 보고 있고 작업이 저장된 경우 [일정(Schedules)] 탭을 선택합니다. 그런 다음 [일정 생성(Create Schedule)]을 선택합니다.
2. [작업 실행 예약(Schedule job run)] 페이지에서 다음 정보를 입력합니다.
   - [이름(Name)]: 작업 예약의 이름을 입력합니다.
   - [빈도(Frequency)]: 작업 일정의 빈도를 입력합니다. 다음을 선택할 수 있습니다.

- [매시간(Hourly)]: 작업이 특정 분에 시작하여 매시간 실행됩니다. 작업을 실행해야 하는 [분(Minute)]을 지정할 수 있습니다. 기본적으로 매시간을 선택하면 작업이 시작 시간(0분)에 실행됩니다.
- [매일(Daily)]: 작업이 특정 시간에 시작하여 매일 실행됩니다. 작업을 실행해야 하는 [분(Minute)]과 작업의 [시작 시간(Start hour)]을 지정할 수 있습니다. 시간은 23시간제를 사용하여 지정되며 오후 시간에는 13~23의 숫자를 사용합니다. 분 및 시간의 기본값은 0입니다. 즉, [매일(Daily)]을 선택하면 기본적으로 작업이 자정에 실행됩니다.
- [매주(Weekly)]: 작업이 매주 하나 이상의 요일에 실행됩니다. 매일에 대해 앞에서 설명한 것과 동일한 설정 외에도 작업을 실행할 요일을 선택할 수 있습니다. 하루 이상을 선택할 수 있습니다.
- [매월(Monthly)]: 작업이 매월 특정 날짜에 실행됩니다. 매일에 대해 앞에서 설명한 것과 동일한 설정 외에도 작업을 실행할 날짜를 선택할 수 있습니다. 1~31의 숫자 값으로 날짜를 지정합니다. 존재하지 않는 날짜(예: 2월 30일)를 선택하면 해당 달에 작업이 실행되지 않습니다.
- [사용자 정의(Custom)]: `cron` 구문을 사용하여 작업 일정에 대한 표현식을 입력합니다. Cron 표현식을 사용하면 매월 말일(특정 날짜 대신) 또는 3개월마다 해당 월의 7일과 21일 같은 더 복잡한 일정을 생성할 수 있습니다.

  AWS Glue Developer Guide의 Cron Expressions를 참조하세요.

- [설명(Description)]: 선택적으로 작업 일정에 대한 설명을 입력할 수 있습니다. 여러 작업에 대해 동일한 일정을 사용하려는 경우 설명이 있으면 작업 일정이 수행하는 작업을 더 쉽게 결정할 수 있습니다.

3. [일정 생성(Create schedule)]을 선택하여 작업 일정을 저장합니다.
4. 일정을 생성하면 콘솔 페이지 상단에 성공 메시지가 나타납니다. 이 배너에서 [작업 세부 정보(Job details)]를 선택하여 작업 세부 정보를 볼 수 있습니다. 그러면 [일정(Schedules)] 탭이 선택된 상태로 시각적 작업 편집기 페이지가 열립니다.

작업 일정 관리

작업에 대한 일정을 생성한 후 시각적 편집기에서 작업을 열고 [일정(Schedules)] 탭을 선택하여 일정을 관리할 수 있습니다.

시각적 편집기의 [일정(Schedules)] 탭에서 다음 태스크를 수행할 수 있습니다.

- 새 예약을 생성합니다.

[일정 생성(Create schedule)]을 선택하고 the section called “작업 실행 예약”에 설명된 대로 일정 정보를 입력합니다.

- 기존 일정을 편집합니다.

  편집할 일정을 선택한 다음 [작업(Action)]과 [일정 편집(Edit schedule)]을 차례로 선택합니다. 기존 일정을 편집하도록 선택하면 [빈도(Frequency)]가 [사용자 정의(Custom)]로 표시되고 일정이 `cron` 표현식으로 표시됩니다. `cron` 표현식을 수정하거나 [빈도(Frequency)] 버튼을 사용하여 새 일정을 지정할 수 있습니다. 변경을 마쳤으면 [예약 업데이트(Update schedule)]를 선택합니다.

- 활성 일정을 일시 중지합니다.

  활성 일정을 선택한 다음 [작업(Action)]과 [일정 일시 중지(Pause schedule)]를 차례로 선택합니다. 일정이 즉시 비활성화됩니다. 새로 고침(다시 로드) 단추를 선택하여 업데이트된 작업 일정 상태를 봅니다.

- 일시 중지된 일정을 재개합니다.

  비활성화된 일정을 선택한 다음 [작업(Action)]과 [일정 재개(Resume schedule)]를 차례로 선택합니다. 일정이 즉시 활성화됩니다. 새로 고침(다시 로드) 단추를 선택하여 업데이트된 작업 일정 상태를 봅니다.

- 일정을 삭제합니다.

  제거할 일정을 선택한 다음 [작업(Action)]과 [일정 삭제(Delete schedule)]를 차례로 선택합니다. 일정이 즉시 삭제됩니다. 새로 고침(다시 로드) 단추를 선택하여 업데이트된 작업 일정 목록을 봅니다. 일정은 완전히 제거될 때까지 [삭제 중(Deleting)] 상태로 표시됩니다.

작업 실행 중지

작업 실행이 완료되기 전에 작업을 중지할 수 있습니다. 작업이 제대로 구성되지 않았거나 작업을 완료하는 데 너무 오래 걸리는 경우 이 옵션을 선택할 수 있습니다.

[모니터링(Monitoring)] 페이지의 [작업 실행(Job runs)] 목록에서 중지할 작업을 선택하고 [작업(Actions)]을 선택한 다음 [실행 중지(Stop run)]를 선택합니다.

작업 보기

[작업(Jobs)] 페이지에서 모든 작업을 볼 수 있습니다. 탐색 창에서 [작업(Jobs)]을 선택하여 이 페이지에 액세스할 수 있습니다.

[작업(Jobs)] 페이지에서 계정에 생성된 모든 작업을 볼 수 있습니다. [작업(Your jobs)] 목록에는 작업 이름, 유형, 해당 작업의 마지막 실행 상태 및 작업이 생성되고 마지막으로 수정된 날짜가 표시됩니다. 작업의 이름을 선택하여 해당 작업에 대한 자세한 정보를 볼 수 있습니다.

모니터링 대시보드를 사용하여 모든 작업을 볼 수도 있습니다. 탐색 창에서 [모니터링(Monitoring)]을 선택하여 대시보드에 액세스할 수 있습니다.

작업 표시 사용자 지정

[작업(Jobs)] 페이지의 [작업(Your jobs)] 섹션에 작업이 표시되는 방식을 사용자 지정할 수 있습니다. 또한 검색 텍스트 필드에 텍스트를 입력하여 해당 텍스트가 포함된 이름의 작업만 표시할 수 있습니다.

작업(Your jobs) 섹션에서 설정 아이콘
( )
을 선택한 경우 AWS Glue Studio가 테이블에 정보를 표시하는 방식을 사용자 지정할 수 있습니다. 표시에서 텍스트 줄 바꿈을 선택하고, 페이지에 표시되는 작업 수를 변경하고, 표시할 열을 지정할 수 있습니다.

최근 작업 실행에 대한 정보 보기

소스 위치에 새 데이터가 추가되면 작업이 여러 번 실행될 수 있습니다. 작업이 실행될 때마다 작업 실행에 고유한 ID가 할당되고 해당 작업 실행에 대한 정보가 수집됩니다. 다음 방법을 사용하여 이 정보를 볼 수 있습니다.

- 현재 표시된 작업에 대한 작업 실행 정보를 보려면 시각적 편집기의 [실행(Runs)] 탭을 선택합니다.

  [실행(Runs)] 탭([최근 작업 실행(Recent job runs)] 페이지)에는 각 작업 실행에 대한 카드가 있습니다. [실행(Runs)] 탭에 표시되는 정보는 다음과 같습니다.
  - 작업 실행 ID
  - 이 작업을 실행하려는 시도 횟수
  - 작업 실행 상태
  - 작업 실행의 시작 및 종료 시간
  - 작업 실행을 위한 런타임
  - 작업 로그 파일에 대한 링크
  - 작업 오류 로그 파일에 대한 링크
  - 실패한 작업에 대해 반환된 오류
- 작업 실행을 선택하면 다음을 포함하여 작업에 대한 추가 정보를 볼 수 있습니다.

- 입력 인수
- 연속 로그
- 지표 - 기본 지표의 시각화를 볼 수 있습니다. 포함된 지표에 대한 자세한 내용은 the section called “Spark 작업 실행에 대한 Amazon CloudWatch 지표 보기” 섹션을 참조하세요.
- Spark UI - Spark UI에서 작업에 대한 Spark 로그를 시각화할 수 있습니다. Spark Web UI에 관해 자세한 내용은 the section called “Spark UI로 모니터링”를 참조하십시오. the section called “작업을 위한 Spark UI 사용 설정”의 절차에 따라 이 기능을 활성화합니다.

세부 정보 보기를 선택하여 작업 실행 세부 정보 페이지에서 유사한 정보를 볼 수 있습니다. 모니터링 페이지를 통해 작업 실행 세부 정보 페이지로 이동할 수도 있습니다. 탐색 창에서 모니터링을 선택합니다. [작업 실행(Job runs)] 목록까지 아래로 스크롤합니다. 작업을 선택한 후 [실행 세부 정보 보기(View run details)]를 선택합니다. 내용은 작업 실행의 세부 정보 보기에 설명되어 있습니다.

작업 로그에 대한 자세한 내용은 작업 실행 로그 보기 섹션을 참조하세요.

작업 스크립트 보기

작업의 모든 노드에 대한 정보를 제공하면 AWS Glue Studio는 작업이 소스에서 데이터를 읽고, 데이터를 변환하고, 대상 위치에 데이터를 쓰는 데 사용하는 스크립트를 생성합니다. 작업을 저장하면 이 스크립트를 언제든지 볼 수 있습니다.

작업에 대해 생성된 스크립트를 보려면

1. 탐색 창에서 [작업(Jobs)]을 선택합니다.
2. [작업(Jobs)] 페이지의 [작업 목록(Your Jobs)]에서 검토할 작업의 이름을 선택합니다. 또는 목록에서 작업을 선택하고 [작업(Actions)] 메뉴를 선택한 다음 [작업 편집(Edit job)]을 선택할 수 있습니다.
3. 시각적 편집기 페이지에서 맨 위에 있는 [스크립트(Script)] 탭을 선택하여 작업 스크립트를 봅니다.

   작업 스크립트를 편집하려면 AWS Glue 프로그래밍 안내서 섹션을 참조하세요.

작업 속성 수정

작업 다이어그램의 노드는 작업에 의해 수행되는 작업을 정의하지만 작업에 대해 구성 할 수있는 몇 가지 속성도 있습니다. 이러한 속성은 작업이 실행되는 환경, 작업이 사용하는 리소스, 임곗값 설정, 보안 설정 등을 결정합니다.

작업 실행 환경을 사용 지정하려면

1. 탐색 창에서 [작업(Jobs)]을 선택합니다.
2. [작업(Jobs)] 페이지의 [작업 목록(Your Jobs)]에서 검토할 작업의 이름을 선택합니다.
3. 시각적 편집기 페이지에서 작업 편집 창 상단의 [작업 세부 정보(Job details)] 탭을 선택합니다.
4. 필요에 따라 작업 속성을 수정합니다.

   작업 속성에 대한 자세한 내용은 AWS Glue Developer Guide의 Defining Job Properties를 참조하세요.
5. 다음과 같은 추가 작업 속성을 지정해야 하는 경우 [고급 속성(Advanced properties)] 섹션을 확장합니다.

   - [스크립트 파일 이름(Script filename)] - Amazon S3에 작업 스크립트를 저장하는 파일의 이름입니다.
   - [스크립트 경로(Script path)] - 작업 스크립트가 저장되는 Amazon S3 위치입니다.
   - [작업 지표(Job metrics)] - (Python 셸 작업에는 사용할 수 없음) 이 작업이 실행될 때 Amazon CloudWatch 지표 생성을 설정합니다.
   - [연속 로깅(Continuous logging)] - (Python 셸 작업에는 사용할 수 없음) CloudWatch에 대한 연속 로깅을 설정하여 작업이 완료되기 전에 로그를 볼 수 있습니다.
   - [Spark UI] 및 [Spark UI 로그 경로(Spark UI logs path)] - (Python 셸 작업에는 사용할 수 없음) 이 작업을 모니터링하기 위해 Spark UI를 사용하도록 설정하고 Spark UI 로그의 위치를 지정합니다.
   - [최대 동시성(Maximum concurrency)] - 이 작업에 허용된 최대 동시 실행 수를 설정합니다.
   - [임시 경로(Temporary path)] – AWS Glue가 스크립트를 실행할 때 임시 중간 결과가 작성된 Amazon S3의 작업 디렉터리의 위치를 제공합니다.
   - [지연 알림 임곗값(분)(Delay notification threshold (minutes))] - 작업에 대한 지연 임곗값을 지정합니다. 작업이 임곗값에 지정된 시간보다 오래 실행되면 AWS Glue는 작업에 대한 지연 알림을 CloudWatch로 보냅니다.
   - [보안 구성(Security configuration)] 및 [서버 측 암호화(Server-side encryption)] - 이 필드를 사용하여 작업에 대한 암호화 옵션을 선택합니다.
   - [Glue Data Catalog를 Hive 메타스토어로 사용(Use Glue Data Catalog as the Hive metastore)] - Apache Hive Metastore의 대안으로 AWS Glue Data Catalog를 사용하려면 이 옵션을 선택합니다.
   - [추가 네트워크 연결(Additional network connection)] – VPC의 데이터 원본에 대해 `Network` 유형의 연결을 지정하여 작업이 VPC를 통해 데이터에 액세스할 수 있도록 할 수 있습니다.

- [Python 라이브러리 경로(Python library path)], [종속 jar 경로(Dependent jars path)](Python 셸 작업에는 사용할 수 없음) 또는 [참조된 파일 경로(Referenced files) path] – 이 필드를 사용하여 스크립트를 실행할 때 작업에서 사용하는 추가 파일의 위치를 지정합니다.
- [작업 파라미터(Job Parameters)] – 이름이 지정된 파라미터로 작업 스크립트에 전달되는 키-값 페어의 집합을 추가할 수 있습니다. Python은 AWS Glue API를 호출할 경우, 파라미터를 이름에 따라 정확하게 통과하는 것이 최선의 방법입니다. 작업 스크립트에서 파라미터 사용에 대한 자세한 내용은 AWS Glue Developer Guide의 Passing and Accessing Python Parameters in AWS Glue를 참조하세요.
- [태그(Tags)] - 작업에 태그를 추가하면 태그를 구성하고 식별하는 데 도움이 됩니다.

6. 작업 속성을 수정한 후 작업을 저장합니다.

Amazon S3에 Spark 셔플 파일 저장

일부 ETL 작업은 예를 들어 조인 변환을 사용할 때 여러 파티션에서 정보를 읽고 결합해야 합니다. 이 작업을 셔플링이라고 합니다. 셔플 중에 데이터가 디스크에 기록되고 네트워크를 통해 전송됩니다. AWS Glue 버전 3.0을 사용하면 Amazon S3를 이러한 파일의 저장 위치로 구성할 수 있습니다. AWS Glue는 Amazon S3에서 셔플 파일을 쓰고 읽는 셔플 관리자를 제공합니다. Amazon S3에서 셔플 파일 쓰기 및 읽기는 로컬 디스크(또는 Amazon EC2에 크게 최적화된 Amazon EBS)에 비해 느립니다(5~20%). 그러나 Amazon S3는 무제한 스토리지 용량을 제공하므로 작업을 실행할 때 "`No space left on device`" 오류에 대해 걱정할 필요가 없습니다.

셔플 파일에 Amazon S3를 사용하도록 작업을 구성하려면

1. [작업(Jobs)] 페이지의 [작업 목록(Your Jobs)]에서 수정할 작업의 이름을 선택합니다.
2. 시각적 편집기 페이지에서 작업 편집 창 상단의 [작업 세부 정보(Job details)] 탭을 선택합니다.

   [작업 파라미터(Job parameters)] 섹션으로 스크롤을 내립니다.
3. 다음 키-값 페어를 지정합니다.

   - `--write-shuffle-files-to-s3` — `true`

     AWS Glue의 셔플 관리자가 Amazon S3 버킷을 사용하여 셔플 데이터를 쓰고 읽도록 구성하는 주요 파라미터입니다. 이 파라미터의 기본값은 `false`입니다.
   - (선택 사항) `--write-shuffle-spills-to-s3` - `true`

이 파라미터를 사용하면 유출 파일을 Amazon S3 버킷으로 오프로드할 수 있으므로 AWS Glue에서 Spark 작업에 대한 추가 복원력을 제공합니다. 이는 많은 데이터를 디스크로 유출하는 대규모 워크로드에만 필요합니다. 이 파라미터의 기본값은 `false`입니다.

- (선택 사항) `--conf spark.shuffle.glue.s3ShuffleBucket=s3://<shuffle-bucket>`

  이 파라미터는 셔플 파일을 작성할 때 사용할 Amazon S3 버킷을 지정합니다. 이 파라미터를 설정하지 않으면 위치는 [임시 경로(Temporary path)](`--TempDir`)에 지정된 위치의 `shuffle-data` 폴더입니다.

> **Note**
>
> 셔플 버킷의 위치가 작업이 실행되는 동일한 AWS 리전에 있는지 확인합니다.
> 또한 셔플 서비스는 작업 실행이 완료된 후 파일을 정리하지 않으므로 셔플 버킷 위치에서 Amazon S3 스토리지 수명 주기 정책을 구성해야 합니다. 자세한 내용은 Amazon S3 사용 설명서의 스토리지 수명 주기 관리를 참조하세요.

작업 저장

작업을 저장할 때까지 빨간색 [작업이 저장되지 않음(Job has not been saved)] 설명선이 [저장(Save)] 버튼 왼쪽에 표시됩니다.

작업을 저장하려면

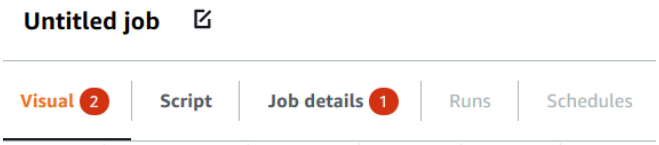
1. [시각적(Visual)] 및 [작업 세부 정보(Job details)] 탭에 필요한 모든 정보를 제공합니다.
2. 저장 버튼을 선택합니다.

   작업을 저장한 후 '저장되지 않음(not saved)' 설명선이 변경되어 작업이 마지막으로 저장된 시간과 날짜를 표시합니다.

작업을 저장하기 전에 AWS Glue Studio를 종료하면 다음에 AWS Glue Studio에 로그인할 때 알림이 표시됩니다. 알림은 저장되지 않은 작업이 있음을 나타내며 복원할 것인지 묻습니다. 작업을 복원하도록 선택한 경우 작업을 계속 편집할 수 있습니다.

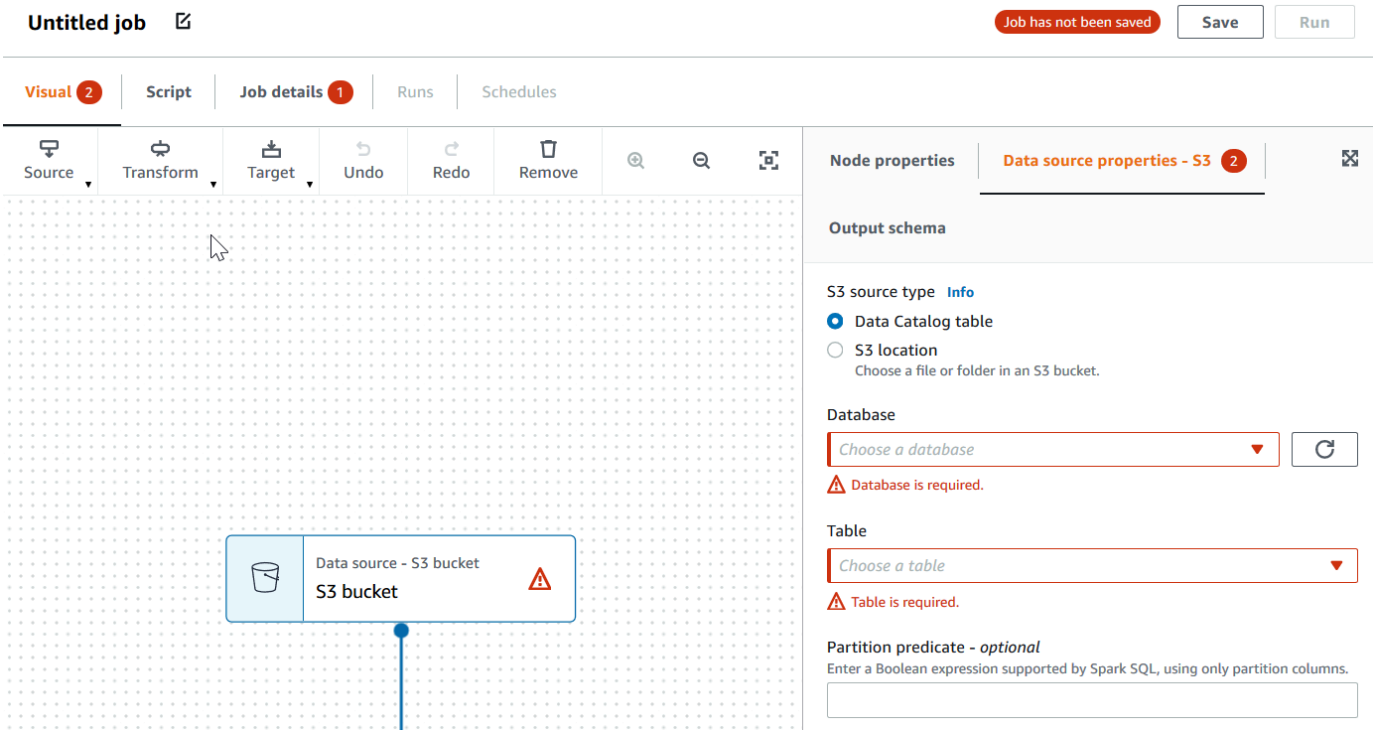
작업 저장 시 발생하는 오류 문제 해결

[저장(Save)] 버튼을 선택했지만 작업에 일부 필수 정보가 누락된 경우 정보가 누락된 탭에 빨간색 설명선이 나타납니다. 설명선의 숫자는 감지된 누락된 필드 수를 나타냅니다.

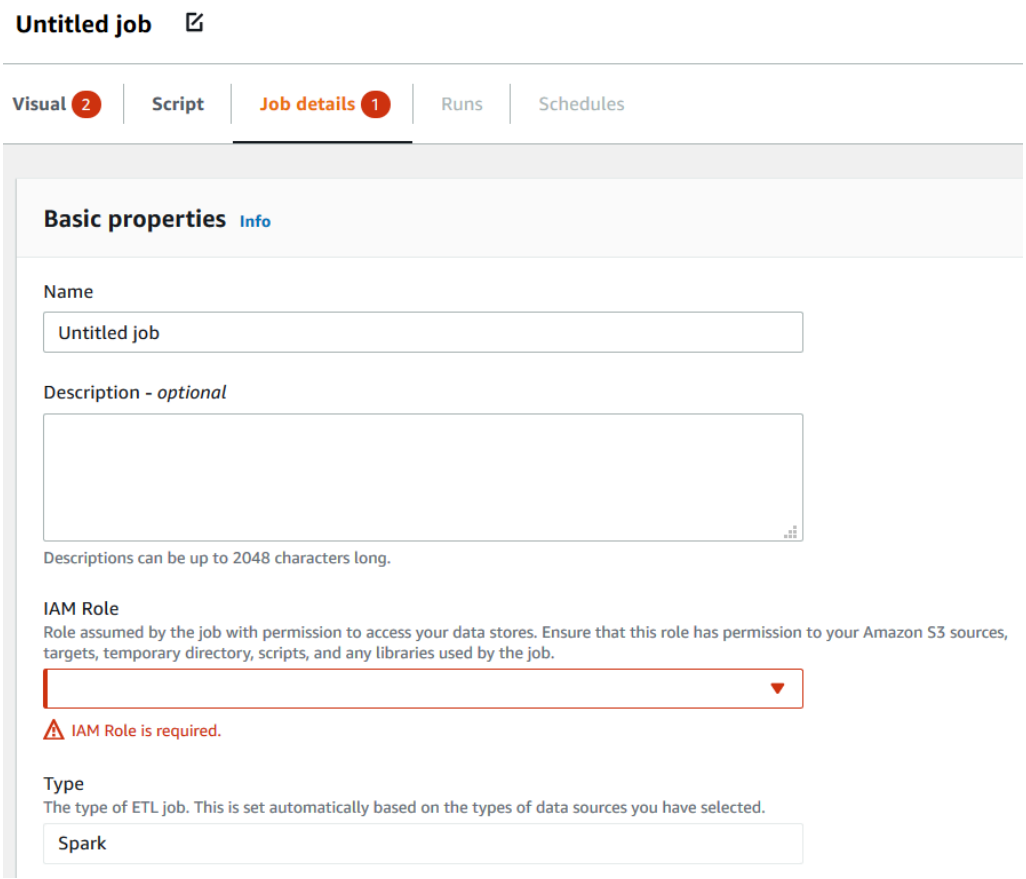


- 시각적 편집기의 노드가 올바르게 구성되지 않은 경우 [시각적(Visual)] 탭은 빨간색 설명선을 표시하고 오류가 있는 노드는 경고 기호 ⚠ 를 표시합니다.

  1. 노드를 선택합니다. 노드 세부 정보 패널에서 누락되거나 잘못된 정보가 있는 탭에 빨간색 설명선이 나타납니다.
  2. 노드 세부 정보 패널에서 빨간색 설명선이 표시된 탭을 선택한 다음 강조 표시된 문제 필드를 찾습니다. 필드 아래의 오류 메시지는 문제에 대한 추가 정보를 제공합니다.



- 작업 속성에 문제가 있는 경우 [작업 세부 정보(Job details)] 탭에 빨간색 설명선이 표시됩니다. 해당 탭을 선택하고 강조 표시된 문제 필드를 찾습니다. 필드 아래의 오류 메시지는 문제에 대한 추가 정보를 제공합니다.



작업 복제

[작업 복제(Clone job)] 작업을 사용하여 기존 작업을 새 작업으로 복사할 수 있습니다.

기존 작업을 복사하여 새 작업을 생성하려면

1. [작업(Jobs)] 페이지의 [작업 목록(Your jobs)]에서 복제할 작업을 선택합니다.
2. [작업(Actions)] 메뉴에서 [작업 복제(Clone job)]를 선택합니다.
3. 새 작업의 이름을 입력합니다. 그런 다음 작업을 저장하거나 편집할 수 있습니다.

작업 삭제

더 이상 필요 없는 작업을 제거할 수 있습니다. 단일 작업으로 하나 이상의 작업을 삭제할 수 있습니다.

AWS Glue Studio에서 작업을 제거하려면

1. [작업(Jobs)] 페이지의 [작업 목록(Your jobs)]에서 삭제할 작업을 선택합니다.

2. [작업(Actions)] 메뉴에서 [작업 삭제(Delete job)]를 선택합니다.
3. **`delete`**를 입력하여 작업 삭제를 확인합니다.

시각적 편집기에서 해당 작업에 대한 [작업 세부 정보(Job details)] 탭을 볼 때 저장된 작업을 삭제할 수도 있습니다.

AWS Glue에서 작업 사용

AWS Glue 작업은 소스 데이터에 연결하여 처리한 다음 데이터 대상에 작성하는 스크립트를 캡슐화합니다. 일반적으로 작업은 추출, 변환 및 로드(ETL) 스크립트를 실행합니다. 작업은 범용 Python 스크립트(Python 셸 작업)를 실행할 수도 있습니다. AWS Glue 트리거는 일정 또는 이벤트에 따라 또는 필요에 따라 작업을 시작할 수 있습니다. 작업 실행을 모니터링하여 완료 상태, 지속 시간, 시작 시간 같은 실행 시간 지표를 이해할 수 있습니다.

AWS Glue에서 생성하는 스크립트를 사용하거나 직접 제공할 수 있습니다. 소스 스키마와 대상 위치 또는 스키마가 있을 경우 AWS Glue 코드 생성기는 Apache Spark API(PySpark) 스크립트를 자동적으로 생성할 수 있습니다. 이 스크립트를 시작 포인트로 사용할 수 있고 목적에 부합하기 위해 편집할 수도 있습니다.

AWS Glue는 JSON, CSV, ORC(Optimized Row Columnar), Apache Parquet 및 Apache Avro를 포함해 몇 가지 데이터 포맷으로 출력 파일을 작성할 수 있습니다. 몇 가지 데이터 포맷의 경우, 일반 압축 포맷이 작성될 수 있습니다.

AWS Glue에서 지원하는 작업 유형은 다음과 같습니다.

- Spark 작업은 AWS Glue에서 관리하는 Apache Spark 환경에서 실행됩니다. 데이터를 배치로 처리합니다.
- 스트리밍 ETL 작업은 데이터 스트림에 대해 ETL을 수행한다는 점을 제외하고 Spark 작업과 유사합니다. 이는 Apache Spark Structured Streaming 프레임워크를 사용합니다. 스트리밍 ETL 작업에 일부 Spark 작업 기능을 사용할 수 없습니다.
- Python 셸 작업은 Python 스크립트를 셸로 실행하고 사용 중인 AWS Glue 버전에 따라 다른 Python 버전을 지원합니다. 이러한 작업을 사용하여 Apache Spark 환경이 필요하지 않은 작업을 예약하고 실행할 수 있습니다.
- Ray는 Python을 중심으로 워크로드를 확장하는 데 사용될 수 있는 오픈 소스 분산 계산 프레임워크입니다. AWS Glue Ray 작업 및 대화형 세션을 활용하면 Ray를 AWS Glue 내에서 사용할 수 있습니다.

다음 섹션에서는 AWS Glue에서 ETL 및 Ray 작업에 대한 정보를 제공합니다.

주제

- AWS Glue 버전
- AWS Glue에서 Spark 작업 사용

- AWS Glue에서 Ray 작업 사용
- AWS Glue에서 Python 쉘 작업에 대한 작업 속성 구성
- AWS Glue 모니터링
- AWS Glue 작업 실행 상태

AWS Glue 버전

작업을 추가하거나 업데이트할 때 AWS Glue 버전 파라미터를 구성할 수 있습니다. AWS Glue 버전은 AWS Glue가 지원하는 Apache Spark 및 Python 버전을 결정합니다. Python의 버전으로 Spark 유형의 작업에 대해 지원되는 버전을 확인할 수 있습니다. 다음 테이블에는 이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전, 그리고 다른 기능 변경 사항이 나열됩니다.

AWS Glue 버전

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
AWS Glue 5.0	• Spark 3.5.4 • Python 3.11 • Scala 2.12.18	Java 17	프레임워크 업데이트 외에도 이 AWS Glue 릴리스에는 다음과 같은 최적화 및 업그레이드가 내장되어 있습니다. • Amazon SageMaker Unified Studio 지원 • Amazon SageMaker Lakehouse 지원 • 오픈 테이블 형식(OTF)이 Hudi 0.15.0, Iceberg 1.7.1 및 Delta Lake 3.3.0으로 업데이트됨

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			• Lake Formation을 사용한 Spark 네이티브 세분화된 액세스 제어. • 액세스 권한 부여 지원 • 추가 Python 라이브러리를 설치하기 위한 `requirements.txt` 지원 • Amazon DataZone의 데이터 계보 지원 • Amazon S3 테이블 버킷 지원 • AWS Glue Data Catalog 다중 언어 뷰 지원 제한 사항 다음은 AWS Glue 5.0의 제한 사항입니다. • Glue 5.0에서는 Glue 4.0 이하에서 지원되는 AWS Lake Formation 권한을 가진 Glue Dynamic Frame/`GlueContext` 기반 테이블 수준 액세스 제어가 지원되지 않습니다.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			Glue 5.0에서는 새로운 Spark 네이티브 세분화된 액세스 제어(FGAC)를 사용하세요. AWS Glue 버전 5.0으로 마이그레이션에 대한 자세한 내용은 AWS Glue 버전 5.0으로 AWS Glue for Spark 작업 마이그레이션 섹션을 참조하세요.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항

AWS Glue 4.0	Spark 환경 버전
	• Spark 3.3.0
	• Python 3.10
		Java 8	이 AWS Glue 릴리스에는 다음과 같은 AWS Glue 4.0의 여러 최적화 및 업그레이드가 기본으로 제공됩니다.
			• Spark 3.1에서 Spark 3.3으로 다양한 Spark 기능 업그레이드
			  • Pandas와 연결할 경우 몇 가지 기능이 개선됩니다. 자세한 내용은 Spark 3.3의 새로운 기능을 참조하세요.
			  • Amazon EMR에서 개발된 추가 최적화 포함
			  • EMR 파일 시스템(EMRFS) 2.53으로 업그레이드
			• Log4j 1.x에서 Log4j 2로 마이그레이션
			• Boto의 업그레이드 버전과 같은 AWS Glue 3.0의 여러 Python 모듈 업데이트
			• 기본 Amazon Redshift 커넥터를

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			비롯한 여러 커넥터 업그레이드 부록 C: 커넥터 업그레이드을(를) 참조하세요. • 여러 JDBC 드라이버 업그레이드 부록 B: JDBC 드라이버 업그레이드을(를) 참조하세요. • 새로운 Amazon Redshift 커넥터와 JDBC 드라이버로 업데이트 • Apache Hudi, Delta Lake 및 Apache Iceberg를 통해 개방형 데이터 레이크 프레임워크 기본 지원 • Amazon S3를 사용하여 셔플링 및 탄력적인 스토리지 용량을 지원할 수 있도록 Amazon S3 기반 클라우드 셔플 스토리지 플러그인(Apache Spark 플러그인) 기본 지원 제한 사항 다음은 AWS Glue 4.0의 제한 사항입니다.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			• AWS Glue 기계 학습 및 개인 식별 정보(PII) 변환은 AWS Glue 4.0에서 아직 사용할 수 없습니다. AWS Glue 버전 4.0으로 마이그레이션에 대한 자세한 내용은 AWS Glue 버전 4.0으로 AWS Glue for Spark 작업 마이그레이션 섹션을 참조하세요.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
	Ray 환경 버전 • Ray 2.4.0 Python 3.9	N/A	AWS Glue for Ray에서 분산 Python 애플리케이션을 구축하고 실행합니다. • Python 3.9에서 Ray-2.4.0 데이터 배포(`ray[data]`)를 지원합니다. 이번 Ray 릴리스에 대한 자세한 내용은 Ray GitHub 리포지토리의 Ray-2.4.0을 참조하세요. • `Ray2.4` 런타임 환경에 추가 Python 라이브러리 설치를 지원합니다. 자세한 내용은 the section called “Ray 작업을 위한 추가 Python 모듈” 섹션을 참조하세요. • Ray 작업의 로그와 지표를 Amazon CloudWatch와 통합합니다. 자세한 내용은 the section called “Ray 오류 해결” 및 the section called “Ray 작업 지표”(을)를 참조하세요.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			• 각 작업 실행 페이지에서 AWS Glue Studio의 Ray 작업에 대한 지표를 집계하고 시각화합니다. • 클러스터의 각 작업 디렉터리에 파일을 배포하고, Ray 객체 스토어에서 Amazon S3로 객체를 유출하며, Ray 작업에 할당된 워커 노드의 최소 수를 제어하도록 지원합니다. 자세한 내용은 the section called “Ray 작업 파라미터” 섹션을 참조하세요. AWS Glue 4.0에서 Ray 작업 제한 • Ray에 대한 AWS Glue 대화형 세션은 이번 릴리스에서 계속 평가판으로 유지됩니다. • AWS Glue for Ray 및 Amazon VPC의 통합은 현재 사용할 수 없습니다. AWS에서 VPC 내 리소스는 퍼블릭 경로로만

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			액세스할 수 있습니다. Amazon VPC에서 AWS Glue를 사용하는 방법에 대한 자세한 내용은 the section called “AWS Glue에 대한 인터페이스 VPC 엔드포인트(AWS PrivateLink) 구성 ” 섹션을 참조하세요. • AWS Glue for Ray는 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤), 아시아 태평양(도쿄), 유럽(아일랜드)에서 사용 가능합니다.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항

AWS Glue 3.0	• Spark 3.1.1
	• Python 3.7
		Java 8	Spark 엔진을 3.0으로 업그레이드하는 것 외에도 이 AWS Glue 릴리스에는 다음과 같은 최적화 및 업그레이드가 내장되어 있습니다.
			• Spark의 주요 릴리스인 Spark 3.0에 대해 AWS Glue ETL 라이브러리를 구축합니다.
			• 스트리밍 작업은 AWS Glue 3.0에서 지원됩니다.
			• 성능과 안정성을 위한 새로운 AWS Glue Spark 런타임 최적화를 포함합니다.
			  • CSV 데이터 읽기를 위한 Apache Arrow 기반의 더 빠른 인메모리 열 형식 처리.
			  • CSV 데이터로 벡터화된 읽기를 위한 SIMD 기반 실행.
			  • Spark 업그레이드에는 Amazon EMR에서 개발된

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			추가 최적화도 포함됩니다. • EMRFS가 2.38에서 2.46으로 업그레이드되어 Amazon S3 액세스에 대한 새로운 기능과 버그 수정이 가능합니다. • 새 Spark 버전에 필요한 여러 종속성이 업그레이드되었습니다. • 기본적으로 지원되는 데이터 원본용으로 JDBC 드라이버가 업그레이드되었습니다. 제한 사항 다음은 AWS Glue 3.0의 제한 사항입니다. • AWS Glue 기계 학습 변환은 아직 AWS Glue 3.0에서 사용할 수 없습니다. • 일부 사용자 정의 Spark 커넥터는 Spark 2.4에 종속되고 Spark 3.1과 호환되지 않는 경우

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			AWS Glue 3.0에서 작동하지 않습니다.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
AWS Glue 2.0(2026년 4월 1일 수명 종료)	• Spark 2.4.3 • Python 3.7	N/A	AWS Glue 버전 1.0에서 제공되는 기능 외에도 AWS Glue 버전 2.0은 다음을 제공합니다. • AWS Glue에서 Apache Spark ETL 작업을 실행하고 시작 시간을 단축하기 위한 업그레이드된 인프라. • 기본 로깅은 이제 드라이버와 실행기, 출력 및 오류에 대한 별도의 스트림을 사용하여 실시간으로 이루어집니다. • 작업 수준에서 추가 Python 모듈 또는 다른 버전 지정 지원. **Note** AWS Glue 버전 2.0은 기본 아키텍처 변경으로 인해 일부 종속성 및 버전에 대해 AWS Glue 버전 1.0과 다릅니다. 주요 AWS Glue 버

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			전 릴리스 간에 마이그레이션하기 전에 AWS Glue 작업을 검증합니다.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
AWS Glue 1.0(2026년 4월 1일 수명 종료)	• Spark 2.4.3 • Python 2.7 • Python 3.6	N/A	AWS Glue ETL 작업에서는 Parquet 및 ORC 포맷에 대해 작업 북마크를 유지할 수 있습니다(AWS Glue 버전 1.0 사용). 이전에는 AWS Glue ETL 작업에서 JSON, CSV, Apache Avro, XML처럼 일반적인 Amazon S3 소스 형식만 북마크가 가능했습니다. ETL 입력 및 출력의 포맷 옵션을 설정할 때 Apache Avro 리더/라이터 포맷 1.8을 사용해 Avro 논리적 유형 읽기 및 쓰기를 지원하도록 지정할 수 있습니다(AWS Glue 버전 1.0 사용). 이전에는 버전 1.7 Avro 리더/라이터 포맷만 지원되었습니다. DynamoDB 연결 유형은 라이터 옵션(AWS Glue 버전 1.0 사용)을 지원합니다. 제한 사항 다음은 AWS Glue 1.0의 제한 사항입니다.

AWS Glue 버전	지원되는 런타임 환경 버전	지원되는 Java 버전	기능 변경 사항
			• AWS Glue 버전 0.9 및 1.0은 아시아 태평양(자카르타)(ap-southeast-3), 중동(UAE)(me-central-1) 또는 향후 기타 신규 리전에서 사용할 수 없습니다.
AWS Glue 0.9(2026년 4월 1일 수명 종료)	• Spark 2.2.1 • Python 2.7	N/A	AWS Glue 버전 지정 없이 생성된 작업은 AWS Glue 0.9로 기본 지정됩니다. 제한 사항 다음은 AWS Glue 0.9의 제한 사항입니다. • AWS Glue 버전 0.9 및 1.0은 아시아 태평양(자카르타)(ap-southeast-3), 중동(UAE)(me-central-1) 또는 향후 기타 신규 리전에서 사용할 수 없습니다.

> **Note**
>
> 다음 Glue 버전은 다음 버전의 PythonShell을 지원합니다.
>
> • PythonShell v3.6은 Glue 버전 1.0에서 지원됩니다.

- PythonShell v3.9는 Glue 버전 3.0에서 지원됩니다.

추가로 개발 엔드포인트는 Glue 버전 1.0 및 0.9에서만 지원됩니다.

AWS Glue 버전 지원 정책

AWS Glue는 분석, 기계 학습, 애플리케이션 개발을 위해 데이터를 쉽게 검색, 준비, 결합할 수 있게 해 주는 서버리스 데이터 통합 서비스입니다. AWS Glue 작업에는 AWS Glue에서 데이터 통합 작업을 수행하는 비즈니스 로직이 포함되어 있습니다. AWS Glue에는 Spark(배치 및 스트리밍), Ray 및 Python 쉘이라는 세 가지 작업 유형이 있습니다. 작업을 정의할 때 기본 Spark, Ray 또는 Python 런타임 환경에서 버전을 구성하는 AWS Glue 버전을 지정합니다. 예: AWS Glue 버전 5.0 Spark 작업은 Spark 3.5.4 및 Python 3.11을 지원합니다.

지원 정책

AWS Glue 버전은 유지 관리 및 보안 업데이트가 적용되는 운영 체제, 프로그래밍 언어 및 소프트웨어 라이브러리의 조합을 기반으로 합니다. AWS Glue의 버전 지원 정책은 버전의 주요 구성 요소가 커뮤니티 장기 지원(LTS) 종료에 도달하여 보안 업데이트를 더 이상 사용할 수 없는 경우 버전 지원을 중단하는 것입니다. AWS Glue의 버전 지원 정책에는 다음 상태가 포함됩니다.

EOS(지원 종료) - AWS Glue 버전이 EOS에 도달하는 경우:

- AWS Glue는 더 이상 EOS 버전에 보안 패치 또는 기타 업데이트를 적용하지 않습니다.
- EOS 버전의 AWS Glue 작업은 기술 지원을 받을 수 없습니다.
- AWS Glue는 EOS 버전에서 작업이 실행될 때 SLA를 준수하지 않을 수 있습니다.

수명 종료(EOL) - AWS Glue 버전이 EOL에 도달하는 경우:

- EOL 버전에서는 더 이상 새로운 AWS Glue 작업 또는 대화형 세션을 생성할 수 없습니다.
- AWS Glue 버전에서는 더 이상 작업 실행을 시작할 수 없습니다.
- AWS Glue는 EOL 버전에서 기존 작업 실행 및 대화형 세션을 중지합니다.
- EOL 버전은 AWS Glue SDK 및 API에서 제거됩니다.

다음 AWS Glue 버전은 지원 종료에 도달했으며 수명 종료 날짜 이후에는 더 이상 사용할 수 없습니다. 버전의 지원 상태에 대한 변경 사항은 지정된 날짜의 자정(태평양 시간대 기준)에 시작됩니다.

유형	Glue 버전	지원 종료	수명 종료
Spark	Glue 버전 0.9(Spark 2.2, Scala 2, Python 2)	2022년 6월 1일	2026년 4월 1일
Spark	Glue 버전 1.0(Spark 2.4, Python 2)	2022년 6월 1일	2026년 4월 1일
Spark	Glue 버전 1.0(Spark 2.4, Scala 2, Python 3)	2022년 9월 30일	2026년 4월 1일
Spark	Glue 버전 2.0(Spark 2.4, Python 3)	2024년 1월 31일	2026년 4월 1일
유형	Python 버전	지원 종료	수명 종료
Python 셸	Python 2(Glue 버전 1.0)	2022년 6월 1일	2026년 4월 1일
Python 셸	PythonShell 3.6(Glue 버전 1.0)	2026년 3월 31일	NA
유형	노트북 버전	지원 종료	수명 종료
개발 엔드포인트	Zeppelin 노트북	2022년 9월 30일	NA

AWS는 해당 작업을 지원되는 버전으로 마이그레이션할 것을 적극 권장합니다.

Spark 작업을 최신 AWS Glue 버전으로 마이그레이션하는 방법에 대한 자세한 내용은 AWS Glue 작업을 AWS Glue 버전 5.0으로 마이그레이션을 참조하세요.

Python 셸 작업을 최신 AWS Glue 버전으로 마이그레이션하는 경우:

- 콘솔에서 `Python 3 (Glue Version 4.0)`을 선택합니다.
- CreateJob/UpdateJob API에서 `GlueVersion` 파라미터를 `2.0`으로 설정하고 `Command` 파라미터에서 `PythonVersion`을 `3`으로 설정합니다. `GlueVersion` 구성은 Python 쉘 작업의 동작에 영향을 미치지 않으므로 `GlueVersion`을 늘려도 이점이 없습니다.

- 작업 스크립트가 Python 3과 호환되도록 해야 합니다.

AWS Glue 버전 5.0으로 AWS Glue for Spark 작업 마이그레이션

이 주제에서는 Spark 애플리케이션 및 ETL 작업을 AWS Glue 5.0으로 마이그레이션할 수 있도록 하는 AWS Glue 버전 0.9, 1.0, 2.0, 3.0 및 4.0 간의 변경 사항에 대해 설명합니다. 또한 AWS Glue 5.0의 기능과 이를 사용할 때의 이점에 대해 설명합니다.

AWS Glue ETL 작업에 이 기능을 사용하려면 작업 생성 시 `Glue version`으로 **`5.0`**을 선택합니다.

주제

- 새로운 특성
- AWS Glue 5.0으로 마이그레이션할 작업
- 마이그레이션 체크리스트
- AWS Glue 5.0 기능
- AWS Glue 4.0에서 AWS Glue 5.0으로 마이그레이션
- AWS Glue 3.0에서 AWS Glue 5.0으로 마이그레이션
- AWS Glue 2.0에서 AWS Glue 5.0으로 마이그레이션
- AWS Glue 5.0에서의 로깅 동작 변경 사항
- AWS Glue 5.0용 커넥터 및 JDBC 드라이버 마이그레이션

새로운 특성

이 섹션에서는 AWS Glue 버전 5.0의 새로운 기능과 장점에 대해 설명합니다.

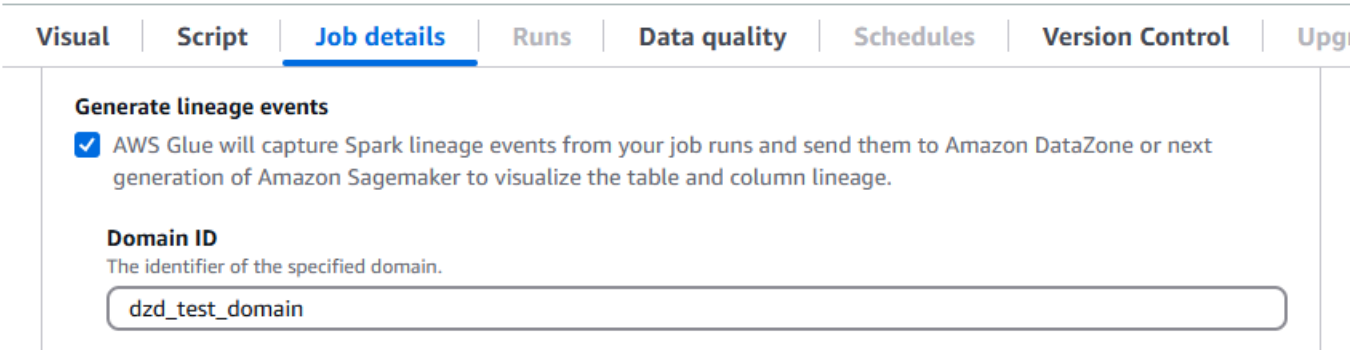
- Apache Spark는 AWS Glue 4.0의 3.3.0에서 AWS Glue 5.0의 3.5.4로 업데이트됩니다. Spark 3.3.0에서 Spark 3.5.4로의 주요 개선 사항을(를) 참조하세요.
- Lake Formation을 사용한 Spark 네이티브 세분화된 액세스 제어(FGAC). 여기에는 Iceberg, Delta 및 Hudi 테이블에 대한 FGAC가 포함됩니다. 자세한 내용은 세분화된 액세스 제어를 위해 AWS Lake Formation과 함께 AWS Glue 사용을 참조하세요.

  Spark 네이티브 FGAC에 대한 다음 고려 사항 또는 제한 사항에 유의하세요.
  - 현재 데이터 쓰기는 지원되지 않습니다.
  - Lake Formation으로 `GlueContext`를 통해 Iceberg에 쓰려면 대신 IAM 액세스 제어의 사용이 필요합니다.

Spark 네이티브 FGAC 사용 시 제한 사항 및 고려 사항에 대한 전체 목록은 the section called “고려 사항” 섹션을 참조하세요.

- AWS Glue의 Amazon S3 데이터에 대한 확장 가능한 액세스 제어 솔루션으로 Amazon S3 Access Grants를 지원합니다. 자세한 내용은 AWS Glue를 통해 Amazon S3 Access Grants 사용 섹션을 참조하세요.
- 오픈 테이블 형식(OTF)이 Hudi 0.15.0, Iceberg 1.7.1 및 Delta Lake 3.3.0으로 업데이트됨
- Amazon SageMaker Unified Studio 지원.
- Amazon SageMaker Lakehouse 및 데이터 추상화 통합. 자세한 내용은 AWS Glue ETL에서 메타스토어 Data Catalog 쿼리 섹션을 참조하세요.
- `requirements.txt`를 사용하여 추가 Python 라이브러리 설치를 지원합니다. 자세한 내용은 requirements.txt를 사용하여 AWS Glue 5.0에 추가 Python 라이브러리 설치 섹션을 참조하세요.
- AWS Glue 5.0은 Amazon DataZone에서 데이터 계보를 지원합니다. Spark 작업 실행 중에 계보 정보를 자동으로 수집하고 Amazon DataZone에서 시각화할 계보 이벤트를 보내도록 AWS Glue를 구성할 수 있습니다. 자세한 내용은 Amazon DataZone의 데이터 계보를 참조하세요.

  AWS Glue 콘솔에서 이를 구성하려면 작업 세부 정보 탭에서 계보 이벤트 생성을 켜고 Amazon DataZone 도메인 ID를 입력합니다.



  또는 다음 작업 파라미터를 제공할 수 있습니다(DataZone 도메인 ID 제공).

  - 키: `--conf`
  - 값:

```
extraListeners=io.openlineage.spark.agent.OpenLineageSparkListener
—conf spark.openlineage.transport.type=amazon_datazone_api
-conf spark.openlineage.transport.domainId=<your-domain-ID>
```

- 커넥터 및 JDBC 드라이버 업데이트. 자세한 내용은 부록 B: JDBC 드라이버 업그레이드 및 부록 C: 커넥터 업그레이드(을)를 참조하세요.

- Java를 8에서 17로 업데이트.
- AWS Glue `G.1X` 및 `G.2X` 작업자에 대한 디스크 공간이 각각 94GB 및 138GB로 증가하여 스토리지 증가. 자세한 내용은 작업 섹션을 참조하세요.
- AWS SDK for Java, 버전 2 지원 - AWS Glue 5.0 작업은 작업이 v2를 지원하는 경우 Java 버전 1.12.569 또는 2.28.8을 사용할 수 있습니다. Java 2.x용 AWS SDK는 버전 1.x 코드 베이스를 크게 재작성한 것입니다. Java 8+에 토대를 두고 있으며, 요청이 많았던 기능들을 몇 가지 추가했습니다. 여기에는 비차단 I/O에 대한 지원과 런타임에 다른 HTTP 구현을 연결하는 기능이 포함됩니다. SDK for Java v1에서 v2로의 마이그레이션 가이드를 포함한 자세한 내용을 확인하려면 AWS SDK for Java, 버전 2 가이드를 참조하세요.
-

호환성에 영향을 미치는 변경 사항

호환성에 영향을 미치는 다음과 같은 변경 사항에 유의하세요.

- AWS Glue 5.0에서는 S3A 파일 시스템을 사용할 때 `fs.s3a.endpoint`와 `fs.s3a.endpoint.region`이 모두 설정되지 않은 경우 S3A에서 사용하는 기본 리전은 `us-east-2`입니다. 이로 인해 특히 VPC 작업의 경우 S3 업로드 제한 시간 오류와 같은 문제가 발생할 수 있습니다. 이 변경으로 인한 문제를 완화하려면 AWS Glue 5.0에서 S3A 파일 시스템을 사용할 때 `fs.s3a.endpoint.region` Spark 구성을 설정합니다.
- Lake Formation 세분화된 액세스 제어(FGAC)
  - AWS Glue 5.0은 Spark DataFrames를 사용하는 새로운 Spark 네이티브 FGAC만 지원합니다. AWS Glue DynamicFrames를 사용하는 FGAC는 지원하지 않습니다.
    - 5.0에서 FGAC를 사용하려면 AWS Glue DynamicFrames에서 Spark DataFrames로 마이그레이션 필요
    - FGAC가 필요하지 않은 경우 Spark DataFrame으로의 마이그레이션이 필요하지 않고, 작업 북마크 및 푸시다운 조건자와 같은 GlueContext 기능은 계속 작동합니다.
  - Spark 네이티브 FGAC를 사용하는 작업에는 사용자 드라이버, 시스템 드라이버, 시스템 실행기, 대기 사용자 실행기 등 최소 4개의 작업자가 필요합니다.
  - 자세한 내용은 세분화된 액세스 제어를 위해 AWS Lake Formation과 함께 AWS Glue 사용을 참조하세요.
- Lake Formation 전체 테이블 액세스(FTA)
  - AWS Glue 5.0은 Spark 네이티브 DataFrames(신규) 및 GlueContext DynamicFrames(레거시, 제한 사항 있음)에서 FTA를 지원합니다.
  - Spark 네이티브 FTA

    - 4.0 스크립트가 GlueContext를 사용하는 경우 네이티브 Spark를 사용하여 마이그레이션합니다.
    - 이 기능은 Hive 및 Iceberg 테이블로 제한됩니다.
    - Spark 네이티브 FTA를 사용하도록 5.0 작업을 구성하는 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.
  - GlueContext DynamicFrame FTA
    - 코드 변경 필요 없음
    - 이 기능은 OTF가 아닌 테이블로 제한되며, Iceberg, Delta Lake 및 Hudi에서는 작동하지 않습니다.
- 벡터화된 SIMD CSV 리더는 지원되지 않습니다.
- 출력 로그 그룹에 대한 지속적 로깅은 지원되지 않습니다. 대신 `error` 로그 그룹을 사용합니다.
- AWS Glue 작업 실행 인사이트 `job-insights-rule-driver`가 더 이상 사용되지 않습니다. 이제 `job-insights-rca-driver` 로그 스트림이 오류 로그 그룹에 위치합니다.
- Athena 기반 사용자 지정/마켓플레이스 커넥터는 지원되지 않습니다.
- Adobe Marketo Engage, Facebook Ads, Google Ads, Google Analytics 4, Google Sheets, Hubspot, Instagram Ads, Intercom, Jira Cloud, Oracle NetSuite, Salesforce, Salesforce Marketing Cloud, Salesforce Marketing Cloud Account Engagement, SAP OData, ServiceNow, Slack, Snapchat Ads, Stripe, Zendesk 및 Zoho CRM 커넥터는 지원되지 않습니다.
- AWS Glue 5.0에서는 사용자 지정 log4j 속성이 지원되지 않습니다.

Spark 3.3.0에서 Spark 3.5.4로의 주요 개선 사항

다음과 같은 개선 사항에 유의합니다.

- Spark Connect용 Python 클라이언트(SPARK-39375).
- 테이블의 열에 대한 DEFAULT 값 지원 구현(SPARK-38334).
- '측면 열 별칭 참조' 지원(SPARK-27561).
- 오류 클래스에 대한 SQLSTATE 용법 강화(SPARK-41994).
- 기본적으로 블룸 필터 조인 활성화(SPARK-38841).
- 대규모 애플리케이션을 위한 Spark UI 확장성 및 드라이버 안정성 향상(SPARK-41053).
- 구조화된 스트리밍의 비동기 진행 상황 추적(SPARK-39591).
- 구조화된 스트리밍의 Python 임의 상태 저장 처리(SPARK-40434).
- Pandas API 적용 범위 개선(SPARK-42882) 및 PySpark의 NumPy 입력 지원(SPARK-39405).

- PySpark 사용자 정의 함수용 메모리 프로파일러 제공(SPARK-40281).
- PyTorch 배포자 구현(SPARK-41589).
- SBOM 아티팩트 게시(SPARK-41893).
- IPv6 전용 환경 지원(SPARK-39457).
- 사용자 지정 K8s 스케줄러(Apache YuniKorn 및 Volcano) GA(SPARK-42802).
- Spark Connect의 Scala 및 Go 클라이언트 지원(SPARK-42554) 및 (SPARK-43351).
- Spark Connect에 대한 PyTorch 기반 분산 ML 지원(SPARK-42471).
- Python 및 Scala에서 Spark Connect에 대한 구조화된 스트리밍 지원(SPARK-42938).
- Python Spark Connect 클라이언트에 대한 Pandas API 지원(SPARK-42497).
- Arrow Python UDF 도입(SPARK-40307).
- Python 사용자 정의 테이블 함수 지원(SPARK-43798).
- PySpark 오류를 오류 클래스로 마이그레이션(SPARK-42986).
- PySpark 테스트 프레임워크(SPARK-44042).
- Datasketches HllSketch에 대한 지원 추가(SPARK-16484).
- 내장 SQL 함수 개선(SPARK-41231).
- IDENTIFIER 절(SPARK-43205).
- Scala, Python 및 R API에 SQL 함수 추가(SPARK-43907).
- SQL 함수에 대한 명명된 인수 지원 추가(SPARK-43922).
- 셔플 데이터가 마이그레이션된 경우 폐기된 실행기에서 불필요한 작업이 다시 실행되지 않도록 방지(SPARK-41469).
- 분산 ML <> Spark Connect(SPARK-42471).
- DeepSpeed 배포자(SPARK-44264).
- RocksDB 상태 저장소에 대한 변경 로그 체크포인트 구현(SPARK-43421).
- 연산자 간의 워터마크 전파 도입(SPARK-42376).
- dropDuplicatesWithinWatermark 도입(SPARK-42931).
- RocksDB 상태 저장소 공급자 메모리 관리 개선 사항(SPARK-43311).

AWS Glue 5.0으로 마이그레이션할 작업

기존 작업의 경우 작업 구성에서 `Glue version`을 이전 버전에서 `Glue 5.0`으로 변경합니다.

- AWS Glue Studio의 `Glue version`에서 `Glue 5.0 - Supports Spark 3.5.4, Scala 2, Python 3`을 선택합니다.
- API에서 `UpdateJob` API 작업의 `GlueVersion` 파라미터에서 **`5.0`**을 선택합니다.

새 작업의 경우 작업을 생성할 때 `Glue 5.0`을 선택합니다.

- 콘솔의 `Glue version`에서 `Spark 3.5.4, Python 3 (Glue Version 5.0) or Spark 3.5.4, Scala 2 (Glue Version 5.0)`를 선택합니다.
- AWS Glue Studio의 `Glue version`에서 `Glue 5.0 - Supports Spark 3.5.4, Scala 2, Python 3`을 선택합니다.
- API에서 `CreateJob` API 작업의 `GlueVersion` 파라미터에서 **`5.0`**을 선택합니다.

AWS Glue 2.0 또는 이전 버전에서 가져온 AWS Glue 5.0의 Spark 이벤트 로그를 보려면 AWS CloudFormation 또는 Docker를 사용하여 AWS Glue 5.0용으로 업그레이드된 Spark 기록 서버를 시작합니다.

마이그레이션 체크리스트

마이그레이션을 위해 이 체크리스트를 검토합니다.

- Java 17 업데이트
- [Scala] v1에서 v2로 AWS SDK 직접 호출 업그레이드
- Python 3.10에서 3.11로 마이그레이션
- [Python] boto 참조를 1.26에서 1.34로 업데이트

AWS Glue 5.0 기능

이 섹션에서는 AWS Glue 기능에 대해 자세히 설명합니다.

AWS Glue ETL에서 메타스토어 Data Catalog 쿼리

AWS Glue 작업을 등록하여 AWS Glue Data Catalog에 액세스할 수 있으며, 이를 통해 다양한 소비자에게 테이블 및 기타 메타스토어 리소스를 제공할 수 있습니다. Data Catalog는 Amazon S3 데이터 레이크의 모든 데이터를 통합하는 다중 카탈로그 계층 구조를 지원합니다. 또한 데이터에 액세스하기 위한 Hive 메타스토어 API와 오픈 소스 Apache Iceberg API를 모두 제공합니다. 이러한 기능은 AWS Glue 및 기타 데이터 중심 서비스(Amazon EMR, Amazon Athena, Amazon Redshift 등)에서 사용할 수 있습니다.

Data Catalog에서 리소스를 생성하면 Apache Iceberg REST API를 지원하는 모든 SQL 엔진에서 해당 리소스에 액세스할 수 있습니다. AWS Lake Formation은 권한을 관리합니다. 구성 후 AWS Glue의 기능을 활용하면 익숙한 애플리케이션에서 이러한 메타스토어 리소스를 쿼리하여 다양한 데이터를 쿼리할 수 있습니다. 여기에는 Apache Spark 및 Trino가 포함됩니다.

메타데이터 리소스 구성 방법

데이터는 AWS Glue Data Catalog를 사용하여 카탈로그, 데이터베이스 및 테이블의 논리적 계층 구조로 구성됩니다.

- 카탈로그 - 스키마 또는 테이블과 같은 데이터 스토어의 객체를 유지하는 논리적 컨테이너입니다.
- 데이터베이스 - 카탈로그의 테이블 및 뷰와 같은 데이터 객체를 구성합니다.
- 테이블 및 뷰 - 이해하기 쉬운 스키마를 사용하여 추상화 계층을 제공하는 데이터베이스의 데이터 객체입니다. 이를 통해 다양한 형식과 다양한 위치의 기본 데이터에 쉽게 액세스할 수 있습니다.

AWS Glue 4.0에서 AWS Glue 5.0으로 마이그레이션

기계 학습 변환을 제외하면 AWS Glue 4.0에 존재하는 모든 기존 작업 파라미터와 주요 기능은 AWS Glue 5.0에 존재합니다.

다음과 같은 새로운 파라미터가 추가되었습니다.

- `--enable-lakeformation-fine-grained-access`: AWS Lake Formation 테이블에서 세분화된 액세스 제어(FGAC) 기능을 활성화합니다.

Spark 마이그레이션 설명서를 참조하세요.

- Migration Guide: Spark Core
- Migration Guide: SQL, Datasets and DataFrame
- Migration Guide: Structured Streaming
- Upgrading PySpark

AWS Glue 3.0에서 AWS Glue 5.0으로 마이그레이션

> **Note**
>
> AWS Glue 4.0과 관련된 마이그레이션 단계는 AWS Glue 3.0에서 AWS Glue 4.0으로 마이그레이션 섹션을 참조하세요.

기계 학습 변환을 제외하면 AWS Glue 3.0에 존재하는 모든 기존 작업 파라미터와 주요 기능은 AWS Glue 5.0에 존재합니다.

AWS Glue 2.0에서 AWS Glue 5.0으로 마이그레이션

> **Note**
>
> AWS Glue 4.0과 관련된 마이그레이션 단계와 AWS Glue 버전 3.0과 4.0 간의 마이그레이션 차이점 목록은 AWS Glue 3.0에서 AWS Glue 4.0으로 마이그레이션 섹션을 참조하세요.

또한 AWS Glue 버전 3.0과 2.0 간의 다음과 같은 마이그레이션 차이점에 유의하세요.

- 기계 학습 변환을 제외하면 AWS Glue 2.0에 존재하는 모든 기존 작업 파라미터와 주요 기능은 AWS Glue 5.0에 존재합니다.
- 몇 가지 Spark 변경만으로도 제거된 기능이 참조되지 않도록 스크립트를 수정해야 할 수 있습니다. 예를 들어 Spark 3.1.1 이상은 Scala 유형이 지정되지 않은 UDF를 사용하지 않지만 Spark 2.4는 이를 허용합니다.
- Python 2.7을 지원하지 않습니다.
- 기존 AWS Glue 2.0 작업에 제공된 추가 jar는 여러 종속성에서 업그레이드가 있었기 때문에 종속성 충돌을 일으킬 수 있습니다. `--user-jars-first` 작업 파라미터를 사용하여 클래스 경로 충돌을 피할 수 있습니다.
- parquet 파일에서 타임스탬프를 로드/저장하는 동작이 변경됩니다. 자세한 내용은 Spark SQL 3.0에서 3.1로 업그레이드를 참조하세요.
- 드라이버/실행기 구성을 위한 다양한 Spark 작업 병렬 처리. `--executor-cores` 작업 인수를 전달하여 작업 병렬 처리를 조정할 수 있습니다.

AWS Glue 5.0에서의 로깅 동작 변경 사항

다음은 AWS Glue 5.0에서의 로깅 동작 변경 사항입니다. 자세한 내용은 AWS Glue 작업에 대한 로깅을 참조하세요.

- 이제 모든 로그(시스템 로그, Spark 대몬 로그, 사용자 로그 및 Glue Logger 로그)가 기본적으로 `/aws-glue/jobs/error` 로그 그룹에 기록됩니다.
- 이전 버전에서 지속 로깅에 사용되는 `/aws-glue/jobs/logs-v2` 로그 그룹은 더 이상 사용되지 않습니다.
- 제거된 지속 로깅 인수를 사용하여 로그 그룹 또는 로그 스트림 이름을 더 이상 변경 또는 사용자 지정할 수 없습니다. 대신 AWS Glue 5.0의 새 작업 인수를 참조하세요.

AWS Glue 5.0에는 두 가지 새로운 작업 인수가 도입됩니다.

- `--custom-logGroup-prefix`: `/aws-glue/jobs/error` 및 `/aws-glue/jobs/output` 로그 그룹에 대한 사용자 지정 접두사를 지정할 수 있습니다.
- `--custom-logStream-prefix`: 로그 그룹 내에 있는 로그 스트림 이름의 사용자 지정 접두사를 지정할 수 있습니다.

  사용자 지정 접두사의 검증 규칙 및 제한 사항:

  - 전체 로그 스트림 이름은 1~512자여야 합니다.
  - 로그 스트림 이름의 사용자 지정 접두사는 400자로 제한됩니다.
  - 접두사에서 허용되는 문자로는 영숫자 문자, 밑줄(`_`), 하이픈(`-`) 및 슬래시(`/`)가 있습니다.

AWS Glue 5.0에서 더 이상 사용되지 않는 지속 로깅 인수

지속 로깅에 대한 다음 작업 인수는 AWS Glue 5.0에서 더 이상 사용되지 않습니다.

- `--enable-continuous-cloudwatch-log`
- `--continuous-log-logGroup`
- `--continuous-log-logStreamPrefix`
- `--continuous-log-conversionPattern`
- `--enable-continuous-log-filter`

AWS Glue 5.0용 커넥터 및 JDBC 드라이버 마이그레이션

업그레이드된 JDBC 및 데이터 레이크 커넥터 버전은 다음을 참조하세요.

- 부록 B: JDBC 드라이버 업그레이드
- 부록 C: 커넥터 업그레이드
- 부록 D: 오픈 테이블 형식 업그레이드

다음 변경 사항은 Glue 5.0의 부록에 나와 있는 커넥터 또는 드라이버 버전에 적용됩니다.

Amazon Redshift

다음과 같은 변경 사항에 유의하세요.

- 커넥터가 Redshift 데이터 공유 테이블을 쿼리할 수 있도록 세 부분으로 구성된 테이블 이름에 대한 지원을 추가합니다.
- 예상 데이터 크기와의 일치도를 높이기 위해 Spark `ShortType` 매핑을 Redshift `INTEGER` 대신 `SMALLINT`를 사용하도록 수정합니다.
- Amazon Redshift Serverless의 사용자 지정 클러스터 이름(CNAME)에 대한 지원이 추가되었습니다.

Apache Hudi

다음과 같은 변경 사항에 유의하세요.

- 레코드 수준 인덱스를 지원합니다.
- 레코드 키의 자동 생성을 지원합니다. 이제 레코드 키 필드를 지정할 필요가 없습니다.

Apache Iceberg

다음과 같은 변경 사항에 유의하세요.

- AWS Lake Formation을 사용하는 세분화된 액세스 제어를 지원합니다.
- 고유한 독립적 수명 주기가 있는 스냅샷에 대한 명명된 참조인 분기 및 태그 지정을 지원합니다.
- 지정된 기간 동안 또는 특정 스냅샷 사이에서 테이블에 대한 변경 사항을 포함하는 보기를 생성하는 변경 로그 보기 프로시저가 추가되었습니다.

Delta Lake

다음과 같은 변경 사항에 유의하세요.

- Apache Iceberg 및 Apache Hudi를 통해 원활하게 액세스할 수 있는 Delta Universal Format(UniForm)을 지원합니다.
- MoR(Merge-on-Read) 패러다임을 구현하는 삭제 벡터를 지원합니다.

AzureCosmos

다음과 같은 변경 사항에 유의하세요.

- 계층적 파티션 키 지원이 추가되었습니다.
- 중첩 속성에 대해 StringType(원시 json)과 함께 사용자 지정 스키마를 사용하는 옵션을 추가했습니다.
- 클라이언트 보안 암호 대신 인증서와 함께 SPN(ServicePrincipal Name) 인증을 사용하도록 허용하는 `spark.cosmos.auth.aad.clientCertPemBase64` 구성 옵션을 추가했습니다.

자세한 내용은 Azure Cosmos DB Spark connector change log를 참조하세요.

Microsoft SQL Server

다음과 같은 변경 사항에 유의하세요.

- 기본적으로 TLS 암호화가 활성화됩니다.
- encrypt = false이지만 서버에 암호화가 필요한 경우 `trustServerCertificate` 연결 설정을 기반으로 인증서가 검증됩니다.
- `aadSecurePrincipalId` 및 `aadSecurePrincipalSecret`가 사용되지 않습니다.
- `getAADSecretPrincipalId` API가 제거되었습니다.
- 영역이 지정될 때 CNAME 확인이 추가되었습니다.

MongoDB

다음과 같은 변경 사항에 유의하세요.

- Spark Structured Streaming을 사용한 마이크로 배치 모드를 지원합니다.
- BSON 데이터 유형을 지원합니다.

- 마이크로 배치 또는 연속 스트리밍 모드를 사용할 때 여러 컬렉션을 읽을 수 있는 지원이 추가되었습니다.
  - `collection` 구성 옵션에 사용되는 컬렉션 이름에 쉼표가 포함된 경우 Spark 커넥터는 이를 두 개의 서로 다른 컬렉션으로 취급합니다. 이를 방지하려면 쉼표 앞에 백슬래시(\)를 붙여 이스케이프 처리해야 합니다.
  - `collection` 구성 옵션에 사용되는 컬렉션 이름이 "*"인 경우 Spark 커넥터는 이를 모든 컬렉션을 스캔하는 사양으로 해석합니다. 이를 방지하려면 별표 앞에 백슬래시(\)를 붙여 이스케이프 처리해야 합니다.
  - `collection` 구성 옵션에 사용되는 컬렉션 이름에 백슬래시(\)가 포함된 경우 Spark 커넥터는 백슬래시를 이스케이프 문자로 취급하며, 이로 인해 값을 해석하는 방식이 변경될 수 있습니다. 이를 방지하려면 백슬래시 앞에 다른 백슬래시를 붙여 이스케이프 처리해야 합니다.

자세한 내용은 MongoDB connector for Spark release notes를 참조하세요.

Snowflake

다음과 같은 변경 사항에 유의하세요.

- Snowflake 테이블에 저장할 때 `StringType` 열 값을 자동으로 트리밍하는 데 사용할 수 있는 새로운 `trim_space` 파라미터를 도입했습니다. 기본값: `false`.
- 기본적으로 세션 수준에서 `abort_detached_query` 파라미터를 비활성화했습니다.
- OAUTH를 사용할 때 `SFUSER` 파라미터 요구 사항을 제거했습니다.
- 고급 쿼리 푸시다운 기능을 제거했습니다. 대체 기능을 사용할 수 있습니다. 예를 들어, Snowflake 테이블에서 데이터를 로드하는 대신 사용자가 Snowflake SQL 쿼리에서 직접 데이터를 로드할 수 있습니다.

자세한 내용은 Snowflake Connector for Spark release notes를 참조하세요.

부록 A: 중요한 종속성 업그레이드

다음은 종속성 업그레이드입니다.

종속성	AWS Glue 5.0 버전	AWS Glue 4.0 버전	AWS Glue 3.0 버전	AWS Glue 2.0 버전	AWS Glue 1.0 버전
Java	17	8	8	8	8

종속성	AWS Glue 5.0 버전	AWS Glue 4.0 버전	AWS Glue 3.0 버전	AWS Glue 2.0 버전	AWS Glue 1.0 버전
Spark	3.5.4	3.3.0-amzn-1	3.1.1-amzn-0	2.4.3	2.4.3
Hadoop	3.4.1	3.3.3-amzn-0	3.2.1-amzn-3	2.8.5-amzn-5	2.8.5-amzn-1
Scala	2.12.18	2.12	2.12	2.11	2.11
Jackson	2.15.2	2.12	2.12	2.11	2.11
Hive	2.3.9-amzn-4	2.3.9-amzn-2	2.3.7-amzn-4	1.2	1.2
EMRFS	2.69.0	2.54.0	2.46.0	2.38.0	2.30.0
Json4s	3.7.0-M11	3.7.0-M11	3.6.6	3.5.x	3.5.x
화살표	12.0.1	7.0.0	2.0.0	0.10.0	0.10.0
AWS Glue 데이터 카탈로그 클라이언트	4.5.0	3.7.0	3.0.0	1.10.0	N/A
Java용 AWS SDK	2.29.52	1.12	1.12		
Python	3.11	3.10	3.7	2.7 및 3.6	2.7 및 3.6
Boto	1.34.131	1.26	1.18	1.12	N/A
EMR DynamoDB 커넥터	5.6.0	4.16.0			

부록 B: JDBC 드라이버 업그레이드

다음은 JDBC 드라이버 업그레이드입니다.

드라이버	AWS Glue 5.0의 JDBC 드라이버 버전	AWS Glue 4.0의 JDBC 드라이버 버전	AWS Glue 3.0의 JDBC 드라이버 버전	과거 AWS Glue 버전의 JDBC 드라이버 버전
MySQL	8.0.33	8.0.23	8.0.23	5.1
Microsoft SQL Server	10.2.0	9.4.0	7.0.0	6.1.0
Oracle Database	23.3.0.23.09	21.7	21.1	11.2
PostgreSQL	42.7.3	42.3.6	42.2.18	42.1.0
Amazon Redshift	redshift-jdbc42-2.1.0.29	redshift-jdbc42-2.1.0.16	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017
SAP Hana	2.20.17	2.17.12		
Teradata	20.00.00.33	20.00.00.06		

부록 C: 커넥터 업그레이드

다음은 커넥터 업그레이드입니다.

드라이버	AWS Glue 5.0의 커넥터 버전	AWS Glue 4.0의 커넥터 버전	AWS Glue 3.0의 커넥터 버전
EMR DynamoDB 커넥터	5.6.0	4.16.0	
Amazon Redshift	6.4.0	6.1.3	
OpenSearch	1.2.0	1.0.1	
MongoDB	10.4.0	10.0.4	3.0.0
Snowflake	3.0.0	2.12.0	

드라이버	AWS Glue 5.0의 커넥터 버전	AWS Glue 4.0의 커넥터 버전	AWS Glue 3.0의 커넥터 버전
Google BigQuery	0.32.2	0.32.2	
AzureCosmos	4.33.0	4.22.0	
AzureSQL	1.3.0	1.3.0	
Vertica	3.3.5	3.3.5	

부록 D: 오픈 테이블 형식 업그레이드

다음은 오픈 테이블 형식 업그레이드입니다.

OTF	AWS Glue 5.0의 커넥터 버전	AWS Glue 4.0의 커넥터 버전	AWS Glue 3.0의 커넥터 버전
Hudi	0.15.0	0.12.1	0.10.1
Delta Lake	3.3.0	2.1.0	1.0.0
Iceberg	1.7.1	1.0.0	0.13.1

AWS Glue 버전 4.0으로 AWS Glue for Spark 작업 마이그레이션

이 주제에서는 Spark 애플리케이션 및 ETL 작업을 AWS Glue 4.0으로 마이그레이션할 수 있도록 하는 AWS Glue 버전 0.9, 1.0, 2.0 및 3.0 간의 변경 사항에 대해 설명합니다. 또한 AWS Glue 4.0의 기능과 이를 사용할 때의 이점에 대해 설명합니다.

AWS Glue ETL 작업에 이 기능을 사용하려면 작업 생성 시 `Glue version`으로 **`4.0`**을 선택합니다.

주제

- 지원되는 새로운 기능
- AWS Glue 4.0으로 마이그레이션할 작업
- 마이그레이션 체크리스트
- AWS Glue 3.0에서 AWS Glue 4.0으로 마이그레이션

- AWS Glue 2.0에서 AWS Glue 4.0으로 마이그레이션
- AWS Glue 1.0에서 AWS Glue 4.0으로 마이그레이션
- AWS Glue 0.9에서 AWS Glue 4.0으로 마이그레이션
- AWS Glue 4.0용 커넥터 및 JDBC 드라이버 마이그레이션
- 부록 A: 중요한 종속성 업그레이드
- 부록 B: JDBC 드라이버 업그레이드
- 부록 C: 커넥터 업그레이드

지원되는 새로운 기능

이 섹션에서는 AWS Glue 버전 4.0의 새로운 기능과 장점에 대해 설명합니다.

- Apache Spark 3.3.0을 기반으로 하지만 적응형 쿼리 실행, 벡터화된 리더, 최적화된 셔플, 파티션 병합과 같은 Amazon EMR 및 AWS Glue의 최적화를 포함합니다.
- MySQL, Microsoft SQL Server, Oracle, PostgreSQL, MongoDB 및 Spark 3.3.0에서 가져온 업그레이드된 Spark 라이브러리 및 종속성을 포함한 모든 AWS Glue 기본 소스용 JDBC 드라이버가 업그레이드되었습니다.
- 새로운 Amazon Redshift 커넥터와 JDBC 드라이버로 업데이트.
- 업그레이드된 EMR 파일 시스템(EMRFS)으로 Amazon S3 액세스가 최적화되었으며 Amazon S3 최적화 출력 커미터가 기본적으로 사용됩니다.
- 파티션 인덱스, 푸시다운 조건자, 파티션 목록 및 업그레이드된 Hive 메타스토어 클라이언트로 Data Catalog 액세스가 최적화되었습니다.
- 셀 수준 필터링 및 데이터 레이크 트랜잭션이 있는 관리되는 카탈로그 테이블을 위해 Lake Formation과 통합됩니다.
- 시작 대기 시간이 단축되어 전반적인 작업 완료 시간과 상호 작용성이 개선되었습니다.
- Spark 작업은 최소 10분에서 1분으로 10배 더 짧은 최소 청구 기간으로 1초 단위로 청구됩니다.
- Apache Hudi, Delta Lake 및 Apache Iceberg를 통해 개방형 데이터 레이크 프레임워크 기본 지원.
- Amazon S3를 사용하여 셔플링 및 탄력적인 스토리지 용량을 지원할 수 있도록 Amazon S3 기반 클라우드 셔플 스토리지 플러그인(Apache Spark 플러그인) 기본 지원.

Spark 3.1.1에서 Spark 3.3.0으로 개선된 주요 기능

다음과 같은 개선 사항에 유의합니다.

- 행 수준 런타임 필터링(SPARK-32268).
- ANSI 개선 사항(SPARK-38860).
- 오류 메시지 개선 사항(SPARK-38781).
- Parquet 벡터화된 리더(SPARK-34863)에 대해 복합 유형을 지원합니다.
- Spark SQL(SPARK-37273)에 대해 숨김 파일 메타데이터를 지원합니다.
- Python/Pandas UDF(SPARK-37443)에 대한 프로파일러를 제공합니다.
- 여러 배치에서 Triggger.Once과 같은 스트리밍 쿼리를 실행하기 위해 Triggger.AvailableNow를 소개합니다(SPARK-36533).
- 보다 포괄적인 Datasource V2 푸시다운 기능(SPARK-38788).
- log4j 1에서 log4j 2(SPARK-37814)로 마이그레이션합니다.

기타 주요 변경 사항

다음과 같은 변경 사항에 유의하세요.

- 호환성에 영향을 미치는 변경 사항
  - 문서 및 Python/문서(SPARK-36977)에서 Python 3.6 지원에 대한 참조를 삭제합니다.
  - 기본 제공 pickle을 cloudpickle(SPARK-32079)로 대체하여 명명된 tuple hack을 제거합니다.
  - 최소 pandas 버전을 1.0.5(SPARK-37465)로 올립니다.

AWS Glue 4.0으로 마이그레이션할 작업

기존 작업의 경우 작업 구성에서 `Glue version`을 이전 버전에서 `Glue 4.0`으로 변경합니다.

- AWS Glue Studio의 `Glue version`에서 `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3`을 선택합니다.
- API에서 `UpdateJob` API 작업의 `GlueVersion` 파라미터에서 **`4.0`**을 선택합니다.

새 작업의 경우 작업을 생성할 때 `Glue 4.0`을 선택합니다.

- 콘솔의 `Glue version`에서 `Spark 3.3, Python 3 (Glue Version 4.0) or Spark 3.3, Scala 2 (Glue Version 3.0)`를 선택합니다.
- AWS Glue Studio의 `Glue version`에서 `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3`을 선택합니다.

- API에서 `CreateJob` API 작업의 `GlueVersion` 파라미터에서 **`4.0`**을 선택합니다.

AWS Glue 2.0 이전 버전에서 가져온 AWS Glue 4.0의 Spark 이벤트 로그를 보려면 AWS CloudFormation 또는 Docker를 사용하여 AWS Glue 4.0용으로 업그레이드된 Spark 기록 서버를 시작합니다.

마이그레이션 체크리스트

- 작업의 외부 Python 라이브러리가 Python 2.7/3.6에 종속되나요?
  - Spark 3.3.0에서 Python 2.7 및 3.6 지원이 완전히 제거되었으므로 종속 라이브러리를 Python 2.7/3.6에서 Python 3.10으로 업데이트합니다.

AWS Glue 3.0에서 AWS Glue 4.0으로 마이그레이션

마이그레이션할 때 다음 변경 사항에 유의합니다.

- AWS Glue 3.0에 존재하는 모든 기존 작업 파라미터와 주요 기능은 AWS Glue 4.0에 존재합니다.
- AWS Glue 3.0은 Amazon EMR 최적화 Spark 3.1.1을 사용하고 AWS Glue 4.0은 Amazon EMR 최적화 Spark 3.3.0을 사용합니다.

  몇 가지 Spark 변경만으로도 제거된 기능이 참조되지 않도록 스크립트를 수정해야 할 수 있습니다.
- AWS Glue 4.0에는 EMRFS와 Hadoop에 대한 업데이트도 제공됩니다. 특정 버전 번호는 부록 A: 중요한 종속성 업그레이드 섹션을 참조하세요.
- ETL 작업에 제공되는 AWS SDK가 이제 1.11에서 1.12로 업그레이드되었습니다.
- 모든 Python 작업에서는 Python 버전 3.10을 사용합니다. 이전에는 Python 3.7이 AWS Glue 3.0에서 사용되었습니다.

  따라서 AWS Glue에서 기본적으로 가져오는 일부 pymodule이 업그레이드됩니다.
- Log4j가 Log4j2로 업그레이드되었습니다.
  - Log4j2 마이그레이션 경로에 대한 자세한 내용은 Log4j 설명서를 참조하세요.
  - 대신 사용자 지정 log4j.properties 파일의 이름을 적절한 log4j2 속성을 사용하여 log4j2.properties 파일로 바꿔야 합니다.
- 특정 커넥터를 마이그레이션하려면 AWS Glue 4.0용 커넥터 및 JDBC 드라이버 마이그레이션 섹션을 참조하세요.

- AWS 암호화 SDK가 1.x에서 2.x로 업그레이드되었습니다. AWS Glue보안 구성을 사용하는 AWS Glue 작업과 런타임에 제공된 AWS 암호화 SDK 종속성에 종속되는 작업이 영향을 받습니다. AWS Glue 작업 마이그레이션 지침을 참조하세요.

  AWS Glue 2.0/3.0에는 이미 AWS 암호화 SDK 브리지 버전이 포함되어 있으므로 AWS Glue 2.0/3.0 작업을 AWS Glue 4.0 작업으로 안전하게 업그레이드할 수 있습니다.

Spark 마이그레이션 설명서를 참조하세요.

- Spark SQL 3.1에서 3.2로 업그레이드
- Spark SQL 3.2에서 3.3으로 업그레이드

AWS Glue 2.0에서 AWS Glue 4.0으로 마이그레이션

마이그레이션할 때 다음 변경 사항에 유의합니다.

> **Note**
>
> AWS Glue 3.0과 관련된 마이그레이션 단계는 AWS Glue 3.0에서 AWS Glue 4.0으로 마이그레이션 섹션을 참조하세요.

- AWS Glue 2.0에 존재하는 모든 기존 작업 파라미터와 주요 기능은 AWS Glue 4.0에 존재합니다.
- AWS Glue 3.0 이상에서는 Amazon S3에 Parquet 데이터를 쓰기 위한 EMRFS S3 최적화 커미터가 기본값으로 사용됩니다. 그러나 `--enable-s3-parquet-optimized-committer`를 `false`로 설정하여 사용하지 않을 수도 있습니다.
- AWS Glue 2.0은 오픈 소스 Spark 2.4를 사용하고 AWS Glue 4.0은 Amazon EMR 최적화 Spark 3.3.0을 사용합니다.
  - 몇 가지 Spark 변경만으로도 제거된 기능이 참조되지 않도록 스크립트를 수정해야 할 수 있습니다.
  - 예를 들어 Spark 3.3.0은 Scala 유형이 지정되지 않은 UDF를 사용하지 않지만 Spark 2.4는 이를 허용합니다.
- ETL 작업에 제공되는 AWS SDK가 이제 1.11에서 1.12로 업그레이드되었습니다.
- AWS Glue 4.0에는 EMRFS 업데이트, JDBC 드라이버 업데이트, AWS Glue에서 제공하는 Spark 자체에 대한 추가 최적화 기능도 포함되어 있습니다.

- Scala는 2.11에서 2.12로 업데이트되었으며 Scala 2.12는 Scala 2.11과 역호환되지 않습니다.
- Python 3.10은 Python 스크립트에 사용되는 기본 버전입니다. AWS Glue 2.0은 Python 3.7 및 2.7만 사용했기 때문입니다.
  - Python 2.7은 Spark 3.3.0에서 지원되지 않습니다. 작업 구성에서 Python 2를 요청하는 작업은 IllegalArgumentException과 함께 실패합니다.
  - AWS Glue 2.0 이상에서는 추가 Python 모듈을 설치하는 새로운 메커니즘을 사용할 수 있습니다.
- 부록 A: 중요한 종속성 업그레이드에서 강조 표시된 여러 종속성 업데이트
- 기존 AWS Glue 2.0 작업에 제공된 추가 JAR 파일은 2.0에서 4.0의 여러 종속성에서 업그레이드가 있었기 때문에 종속성 충돌을 일으킬 수 있습니다. AWS Glue 4.0에서 `--user-jars-first` AWS Glue 작업 파라미터를 사용하여 클래스 경로 충돌을 피할 수 있습니다.
- AWS Glue 4.0은 Spark 3.3을 사용합니다. Spark 3.1부터 parquet 파일에서 타임스탬프를 로드하거나 해당 파일로 저장하는 동작이 변경되었습니다. 자세한 내용은 Spark SQL 3.0에서 3.1로 업그레이드를 참조하세요.

  타임스탬프 열이 포함된 parquet 데이터를 읽거나 쓸 때는 다음 파라미터를 설정하는 것이 좋습니다. 이러한 파라미터를 설정하면 Spark 2에서 Spark 3으로 업그레이드하는 동안 AWS Glue 동적 프레임과 Spark 데이터 프레임 둘 다에서 발생하는 달력 비호환성 문제를 해결할 수 있습니다. datetime 값을 있는 그대로 읽으려면 CORRECTED 옵션을 사용하고, 읽는 동안 달력 차이를 기준으로 datetime 값을 다시 지정하려면 LEGACY 옵션을 사용합니다.

  ```
  - Key: --conf
  - Value: spark.sql.legacy.parquet.int96RebaseModeInRead=[CORRECTED|LEGACY] --
  conf spark.sql.legacy.parquet.int96RebaseModeInWrite=[CORRECTED|LEGACY] --conf
   spark.sql.legacy.parquet.datetimeRebaseModeInRead=[CORRECTED|LEGACY]
  ```

- 특정 커넥터를 마이그레이션하려면 AWS Glue 4.0용 커넥터 및 JDBC 드라이버 마이그레이션 섹션을 참조하세요.
- AWS 암호화 SDK가 1.x에서 2.x로 업그레이드되었습니다. AWS Glue보안 구성을 사용하는 AWS Glue 작업과 런타임에 제공된 AWS 암호화 SDK 종속성에 종속되는 작업이 영향을 받습니다. AWS Glue 작업 마이그레이션 지침을 참조하세요.
  - AWS Glue 2.0에는 이미 AWS 암호화 SDK 브리지 버전이 포함되어 있으므로 AWS Glue 2.0 작업을 AWS Glue 4.0 작업으로 안전하게 업그레이드할 수 있습니다.

Spark 마이그레이션 설명서를 참조하세요.

- Spark SQL 2.4에서 3.0으로 업그레이드

- Spark SQL 3.1에서 3.2로 업그레이드
- Spark SQL 3.2에서 3.3으로 업그레이드
- Spark 3.0 이후 예상되는 Datetime 동작의 변경 사항

AWS Glue 1.0에서 AWS Glue 4.0으로 마이그레이션

마이그레이션할 때 다음 변경 사항에 유의합니다.

- AWS Glue 1.0은 오픈 소스 Spark 2.4를 사용하고 AWS Glue 4.0은 Amazon EMR 최적화 Spark 3.3.0을 사용합니다.
  - 몇 가지 Spark 변경만으로도 제거된 기능이 참조되지 않도록 스크립트를 수정해야 할 수 있습니다.
  - 예를 들어 Spark 3.3.0은 Scala 유형이 지정되지 않은 UDF를 사용하지 않지만 Spark 2.4는 이를 허용합니다.
- AWS Glue 4.0의 모든 작업은 시작 시간이 크게 개선되어 실행됩니다. Spark 작업은 시작 대기 시간이 최대 10분에서 1분으로 단축되어 최소 청구 기간이 10배 더 짧으며 1초 단위로 청구됩니다.
- AWS Glue 4.0에서는 로깅 동작이 크게 변경되었습니다. Spark 3.3.0의 최소 요구 사항은 Log4j2입니다.
- 부록에서 강조 표시된 여러 종속성 업데이트.
- Scala도 2.11에서 2.12로 업데이트되었으며 Scala 2.12는 Scala 2.11과 역호환되지 않습니다.
- Python 3.10은 Python 스크립트에 사용되는 기본 버전이기도 합니다. AWS Glue 0.9는 Python 2만 사용했기 때문입니다.

  Python 2.7은 Spark 3.3.0에서 지원되지 않습니다. 작업 구성에서 Python 2를 요청하는 작업은 IllegalArgumentException과 함께 실패합니다.
- AWS Glue 2.0 이상에서는 pip를 통해 추가 Python 모듈을 설치하는 새로운 메커니즘을 사용할 수 있습니다. 자세한 내용은AWS Glue 2.0+에서 pip를 사용하여 추가 Python 모듈 설치를 참조하세요.
- AWS Glue 4.0은 Apache YARN에서 실행되지 않으므로 YARN 설정이 적용되지 않습니다.
- AWS Glue 4.0에는 Hadoop 분산 파일 시스템(HDFS)이 없습니다.
- 기존 AWS Glue 1.0 작업에 제공된 추가 JAR 파일은 1.0에서 4.0의 여러 종속성에서 업그레이드가 있었기 때문에 종속성 충돌을 일으킬 수 있습니다. 이 문제를 방지하기 위해 기본적으로 `--user-jars-first` AWS Glue 작업 파라미터를 사용하여 AWS Glue 4.0을 활성화합니다.
- AWS Glue 4.0은 Auto Scaling을 지원합니다. 따라서 auto Scaling이 활성화된 경우 ExecutorAllocationManager 지표를 사용할 수 있습니다.

- AWS Glue 버전 4.0 작업에서는 작업자 수와 작업자 유형을 지정하지만 `maxCapacity`를 지정하지 않습니다.
- AWS Glue 4.0은 아직 기계 학습 변환을 지원하지 않습니다.
- 특정 커넥터를 마이그레이션하려면 AWS Glue 4.0용 커넥터 및 JDBC 드라이버 마이그레이션 섹션을 참조하세요.
- AWS 암호화 SDK가 1.x에서 2.x로 업그레이드되었습니다. AWS Glue보안 구성을 사용하는 AWS Glue 작업과 런타임에 제공된 AWS 암호화 SDK 종속성에 종속되는 작업이 영향을 받습니다. AWS Glue 작업 마이그레이션 지침을 참조하세요.
  - AWS Glue 0.9/1.0 작업을 AWS Glue 4.0 작업으로 직접 마이그레이션할 수 없습니다. 이는 버전 2.x 이상으로 직접 업그레이드하고 모든 새 기능을 즉시 활성화하면 AWS Encryption SDK가 이전 버전의 AWS Encryption SDK에서 암호화된 사이퍼텍스트를 해독할 수 없기 때문입니다.
  - 안전하게 업그레이드하려면 먼저 AWS Encryption SDK 브리지 버전이 포함된 AWS Glue 2.0/3.0 작업으로 마이그레이션하는 것이 좋습니다. 작업을 한 번 실행하여 AWS Encryption SDK 브리지 버전을 활용합니다.
  - 완료되면 AWS Glue 2.0/3.0 작업을 AWS Glue 4.0으로 안전하게 마이그레이션할 수 있습니다.

Spark 마이그레이션 설명서를 참조하세요.

- Spark SQL 2.4에서 3.0으로 업그레이드
- Spark SQL 3.0에서 3.1로 업그레이드
- Spark SQL 3.1에서 3.2로 업그레이드
- Spark SQL 3.2에서 3.3으로 업그레이드
- Spark 3.0 이후 예상되는 Datetime 동작의 변경 사항

AWS Glue 0.9에서 AWS Glue 4.0으로 마이그레이션

마이그레이션할 때 다음 변경 사항에 유의합니다.

- AWS Glue 0.9는 오픈 소스 Spark 2.2.1을 사용하고 AWS Glue 4.0은 Amazon EMR 최적화 Spark 3.3.0을 사용합니다.
  - 몇 가지 Spark 변경만으로도 제거된 기능이 참조되지 않도록 스크립트를 수정해야 할 수 있습니다.
  - 예를 들어 Spark 3.3.0은 Scala 유형이 지정되지 않은 UDF를 사용하지 않지만 Spark 2.2는 이를 허용합니다.

- AWS Glue 4.0의 모든 작업은 시작 시간이 크게 개선되어 실행됩니다. Spark 작업은 시작 대기 시간이 최대 10분에서 1분으로 단축되어 최소 청구 기간이 10배 더 짧으며 1초 단위로 청구됩니다.
- AWS Glue 4.0 이후 로깅 동작이 크게 변경되었습니다. Spark 3.3.0에는 여기서(https://spark.apache.org/docs/latest/core-migration-guide.html#upgrading-from-core-32-to-33) 언급한 것처럼 Log4j2의 최소 요구 사항이 있습니다.
- 부록에서 강조 표시된 여러 종속성 업데이트
- Scala도 2.11에서 2.12로 업데이트되었으며 Scala 2.12는 Scala 2.11과 역호환되지 않습니다.
- Python 3.10은 Python 스크립트에 사용되는 기본 버전이기도 합니다. AWS Glue 0.9는 Python 2만 사용했기 때문입니다.
  - Python 2.7은 Spark 3.3.0에서 지원되지 않습니다. 작업 구성에서 Python 2를 요청하는 작업은 IllegalArgumentException과 함께 실패합니다.
  - pip를 통해 추가 Python 모듈을 설치하는 새로운 메커니즘을 사용할 수 있습니다.
- AWS Glue 4.0은 Apache YARN에서 실행되지 않으므로 YARN 설정이 적용되지 않습니다.
- AWS Glue 4.0에는 Hadoop 분산 파일 시스템(HDFS)이 없습니다.
- 기존 AWS Glue 0.9 작업에 제공된 추가 JAR 파일은 0.9에서 3.0의 여러 종속성에서 업그레이드가 있었기 때문에 종속성 충돌을 일으킬 수 있습니다. AWS Glue 3.0에서 `--user-jars-first` AWS Glue 작업 파라미터를 사용하여 클래스 경로 충돌을 피할 수 있습니다.
- AWS Glue 4.0은 Auto Scaling을 지원합니다. 따라서 auto Scaling이 활성화된 경우 ExecutorAllocationManager 지표를 사용할 수 있습니다.
- AWS Glue 버전 4.0 작업에서는 작업자 수와 작업자 유형을 지정하지만 `maxCapacity`를 지정하지 않습니다.
- AWS Glue 4.0은 아직 기계 학습 변환을 지원하지 않습니다.
- 특정 커넥터를 마이그레이션하려면 AWS Glue 4.0용 커넥터 및 JDBC 드라이버 마이그레이션 섹션을 참조하세요.
- AWS 암호화 SDK가 1.x에서 2.x로 업그레이드되었습니다. AWS Glue보안 구성을 사용하는 AWS Glue 작업과 런타임에 제공된 AWS 암호화 SDK 종속성에 종속되는 작업이 영향을 받습니다. AWS Glue 작업 마이그레이션 지침을 참조하세요.
  - AWS Glue 0.9/1.0 작업을 AWS Glue 4.0 작업으로 직접 마이그레이션할 수 없습니다. 이는 버전 2.x 이상으로 직접 업그레이드하고 모든 새 기능을 즉시 활성화하면 AWS Encryption SDK가 이전 버전의 AWS Encryption SDK에서 암호화된 사이퍼텍스트를 해독할 수 없기 때문입니다.
  - 안전하게 업그레이드하려면 먼저 AWS Encryption SDK 브리지 버전이 포함된 AWS Glue 2.0/3.0 작업으로 마이그레이션하는 것이 좋습니다. 작업을 한 번 실행하여 AWS Encryption SDK 브리지 버전을 활용합니다.

- 완료되면 AWS Glue 2.0/3.0 작업을 AWS Glue 4.0으로 안전하게 마이그레이션할 수 있습니다.

Spark 마이그레이션 설명서를 참조하세요.

- Spark SQL 2.2에서 2.3으로 업그레이드
- Spark SQL 2.3에서 2.4로 업그레이드
- Spark SQL 2.4에서 3.0으로 업그레이드
- Spark SQL 3.0에서 3.1로 업그레이드
- Spark SQL 3.1에서 3.2로 업그레이드
- Spark SQL 3.2에서 3.3으로 업그레이드
- Spark 3.0 이후 예상되는 Datetime 동작의 변경 사항

AWS Glue 4.0용 커넥터 및 JDBC 드라이버 마이그레이션

업그레이드된 JDBC 및 데이터 레이크 커넥터 버전은 다음을 참조하세요.

- 부록 B: JDBC 드라이버 업그레이드
- 부록 C: 커넥터 업그레이드

Hudi

- Spark SQL 지원 개선 사항:
  - `Call Procedure` 명령을 통해 업그레이드, 다운그레이드, 부트스트랩, 정리 및 복구에 대한 지원이 추가되었습니다. Spark SQL에서는 `Create/Drop/Show/Refresh Index` 구문이 가능합니다.
  - Spark SQL과 달리 Spark DataSource를 통한 사용 간 성능 격차가 좁혀졌습니다. 과거에는 데이터 소스 쓰기가 SQL보다 빨랐습니다.
  - 모든 기본 제공 키 생성기는 성능이 우수한 Spark 전용 API 작업을 구현합니다.
  - 대량 `insert` 작업에서 UDF 변환을 RDD 변환으로 대체하여 SerDe 사용 비용을 줄였습니다.
  - Hudi에서 Spark SQL을 사용하려면 SQL 문에서 `primaryKey`를 `tblproperites` 또는 옵션으로 지정해야 합니다. 업데이트 및 삭제 작업의 경우에도 `preCombineField`가 필요합니다.
- `primaryKey` 없이 버전 0.10.0 이전에 생성한 Hudi 테이블은 버전 0.10.0 이후의 `primaryKey` 필드를 사용하여 다시 생성해야 합니다.

PostgreSQL

- 여러 취약점(CVE)이 해결되었습니다.
- Java 8은 기본적으로 지원됩니다.
- 작업에서 바이트 배열을 제외한 배열 배열을 사용하는 경우 이 시나리오를 다차원 배열로 간주할 수 있습니다.

MongoDB

- 현재 MongoDB 커넥터는 Spark 버전 3.1 이상과 MongoDB 버전 4.0 이상을 지원합니다.
- 커넥터 업그레이드로 인해 몇 가지 속성 이름이 변경되었습니다. 예를 들어 URI 속성 이름이 `connection.uri`로 변경되었습니다. 현재 옵션에 대한 자세한 내용은 MongoDB Spark 커넥터 블로그를 참조하세요.
- Amazon DocumentDB에서 호스팅하는 MongoDB 4.0을 사용하면 기능상 몇 가지 차이점이 있습니다. 자세한 내용은 다음 주제를 참조하십시오.
  - 기능적 차이: Amazon DocumentDB 및 MongoDB
  - 지원되는 MongoDB API, 작업 및 데이터 형식
- "파티셔너" 옵션은 `ShardedPartitioner`, `PaginateIntoPartitionsPartitioner` 및 `SinglePartitionPartitioner`로 제한됩니다. 스테이지 오퍼레이터가 MongoDB API를 지원하지 않기 때문에 Amazon DocumentDB에는 기본값 `SamplePartitioner` 및 `PaginateBySizePartitioner`를 사용할 수 없습니다. 자세한 내용은 지원되는 MongoDB API, 작업 및 데이터 형식을 참조하세요.

Delta Lake

- Delta Lake는 이제 SQL에서 시간 여행을 지원하여 오래된 데이터를 쉽게 쿼리할 수 있습니다. 이번 업데이트를 통해 이제 Spark SQL과 DataFrame API를 통해 시간 여행을 사용할 수 있습니다. SQL에서 현재 버전의 TIMESTAMP에 대한 Support 추가되었습니다.
- Spark 3.3에는 배치 쿼리를 위한 `Trigger.Once`와 동일한 방식으로 스트리밍 쿼리를 실행할 수 있는 Triggger.AvailableNow가 도입되었습니다. 이 지원은 Delta 테이블을 스트리밍 소스로 사용하는 경우에도 사용할 수 있습니다.
- 테이블의 열 목록을 반환하는 SHOW COLUMNS를 지원합니다.
- Scala 및 Python DeltaTable API에서 DESCRIBE DETAIL을 지원합니다. DeltaTable API 또는 Spark SQL을 사용하여 Delta 테이블에 대한 세부 정보를 검색합니다.

- SQL Delete, Merge 및 Update 명령에서 작업 지표 반환을 지원합니다. 이전에는 이러한 SQL 명령이 빈 DataFrame을 반환했지만 이제 수행된 작업에 대한 유용한 지표가 포함된 DataFrame을 반환합니다.
- 성능 개선 최적화:
  - 여러 개의 작은 파일을 압축할 때 성능을 높이려면 Optimize 명령에서 `coalesce(1)` 대신 `repartition(1)`을 사용하도록 `spark.databricks.delta.optimize.repartition.enabled=true` 구성 옵션을 설정합니다.
  - 큐 기반 접근 방식을 사용하여 압축 작업을 병렬화하여 성능을 개선했습니다.
- 기타 주요 변경 사항:
  - VACUUUM 및 OPTIMIZE SQL 명령에서 변수 사용을 지원합니다.
  - 카탈로그 테이블을 통한 CONVERT TO DELTA 관련 개선 사항은 다음과 같습니다.
    - 제공되지 않은 경우 카탈로그에서 파티션 스키마를 자동으로 채웁니다.
    - 전체 디렉터리 스캔을 수행하는 대신 카탈로그의 파티션 정보를 사용하여 커밋할 데이터 파일을 찾습니다. 테이블 디렉터리의 모든 데이터 파일을 커밋하는 대신 활성 파티션의 디렉터리에 있는 데이터 파일만 커밋됩니다.
  - DROP COLUMN 및 RENAME COLUMN을 사용하지 않은 경우 열 매핑이 가능한 테이블에 대한 CDF(Change Data Feed) 배치 읽기를 지원합니다. 자세한 내용은 Delta Lake 설명서를 참조하세요.
  - 첫 번째 통과에서 스키마 프루닝을 활성화하여 Update 명령 성능을 개선합니다.

Apache Iceberg

- 스캔 계획 및 Spark 쿼리에 대한 몇 가지 성능 개선 사항이 추가되었습니다.
- 변경 기반 커밋을 사용하여 서비스 측의 커밋 충돌을 해결하는 일반 REST 카탈로그 클라이언트를 추가했습니다.
- `AS OF` SQL 시간 여행 쿼리 구문이 지원됩니다.
- MERGE 및 UPDATE 쿼리에 대한 읽기 중 병합 지원을 추가했습니다.
- Z축을 통한 파티션 재작성 지원을 추가했습니다.
- Theta 스케치 또는 블룸 필터와 같은 대형 통계 및 인덱스 blob을 위한 형식인 Puffin의 사양 및 구현을 추가했습니다.
- 데이터를 점진적으로 소비하기 위한 새 인터페이스를 추가했습니다(추가 및 변경 로그 스캔 모두).
- 대량 작업 및 FileIO 인터페이스에 대한 범위 읽기 지원이 추가되었습니다.

- 메타데이터 트리에서 삭제 파일을 표시하는 메타데이터 테이블을 더 추가했습니다.
- 드롭 테이블 동작이 변경되었습니다. Iceberg 0.13.1에서 `DROP TABLE`을 실행하면 카탈로그에서 테이블이 제거되고 테이블 내용도 삭제됩니다. Iceberg 1.0.0에서 `DROP TABLE`은 카탈로그에서만 테이블을 제거합니다. 테이블 내용을 삭제하려면 `DROP TABLE PURGE`를 사용합니다.
- Parquet 벡터화된 읽기는 Iceberg 1.0.0에서 기본적으로 활성화됩니다. 벡터화된 읽기를 비활성화하려면 `read.parquet.vectorization.enabled`를 `false`로 설정합니다.

Oracle

변경 사항은 미미합니다.

MySQL

변경 사항은 미미합니다.

Amazon Redshift

AWS Glue 4.0에는 새로운 JDBC 드라이버가 포함된 새로운 Amazon Redshift 커넥터가 있습니다. 향상된 기능 및 이전 AWS Glue 버전에서 마이그레이션하는 방법에 대한 자세한 내용은 the section called “Redshift 연결” 섹션을 참조하세요.

부록 A: 중요한 종속성 업그레이드

다음은 종속성 업그레이드입니다.

종속성	AWS Glue 4.0 버전	AWS Glue 3.0 버전	AWS Glue 2.0 버전	AWS Glue 1.0 버전
Spark	3.3.0-amzn-1	3.1.1-amzn-0	2.4.3	2.4.3
Hadoop	3.3.3-amzn-0	3.2.1-amzn-3	2.8.5-amzn-5	2.8.5-amzn-1
Scala	2.12	2.12	2.11	2.11
Jackson	2.13.3	2.10.x	2.7.x	2.7.x
Hive	2.3.9-amzn-2	2.3.7-amzn-4	1.2	1.2
EMRFS	2.54.0	2.46.0	2.38.0	2.30.0

종속성	AWS Glue 4.0 버전	AWS Glue 3.0 버전	AWS Glue 2.0 버전	AWS Glue 1.0 버전
Json4s	3.7.0-M11	3.6.6	3.5.x	3.5.x
화살표	7.0.0	2.0.0	0.10.0	0.10.0
AWS Glue 데이터 카탈로그 클라이언트	3.7.0	3.0.0	1.10.0	N/A
Python	3.10	3.7	2.7 및 3.6	2.7 및 3.6
Boto	1.26	1.18	1.12	N/A

부록 B: JDBC 드라이버 업그레이드

다음은 JDBC 드라이버 업그레이드입니다.

드라이버	과거 AWS Glue 버전의 JDBC 드라이버 버전	AWS Glue 3.0의 JDBC 드라이버 버전	AWS Glue 4.0의 JDBC 드라이버 버전
MySQL	5.1	8.0.23	8.0.23
Microsoft SQL Server	6.1.0	7.0.0	9.4.0
Oracle Database	11.2	21.1	21.7
PostgreSQL	42.1.0	42.2.18	42.3.6
MongoDB	2.0.0	4.0.0	4.7.2
Amazon Redshift	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017	redshift-jdbc42-2.1.0.16

부록 C: 커넥터 업그레이드

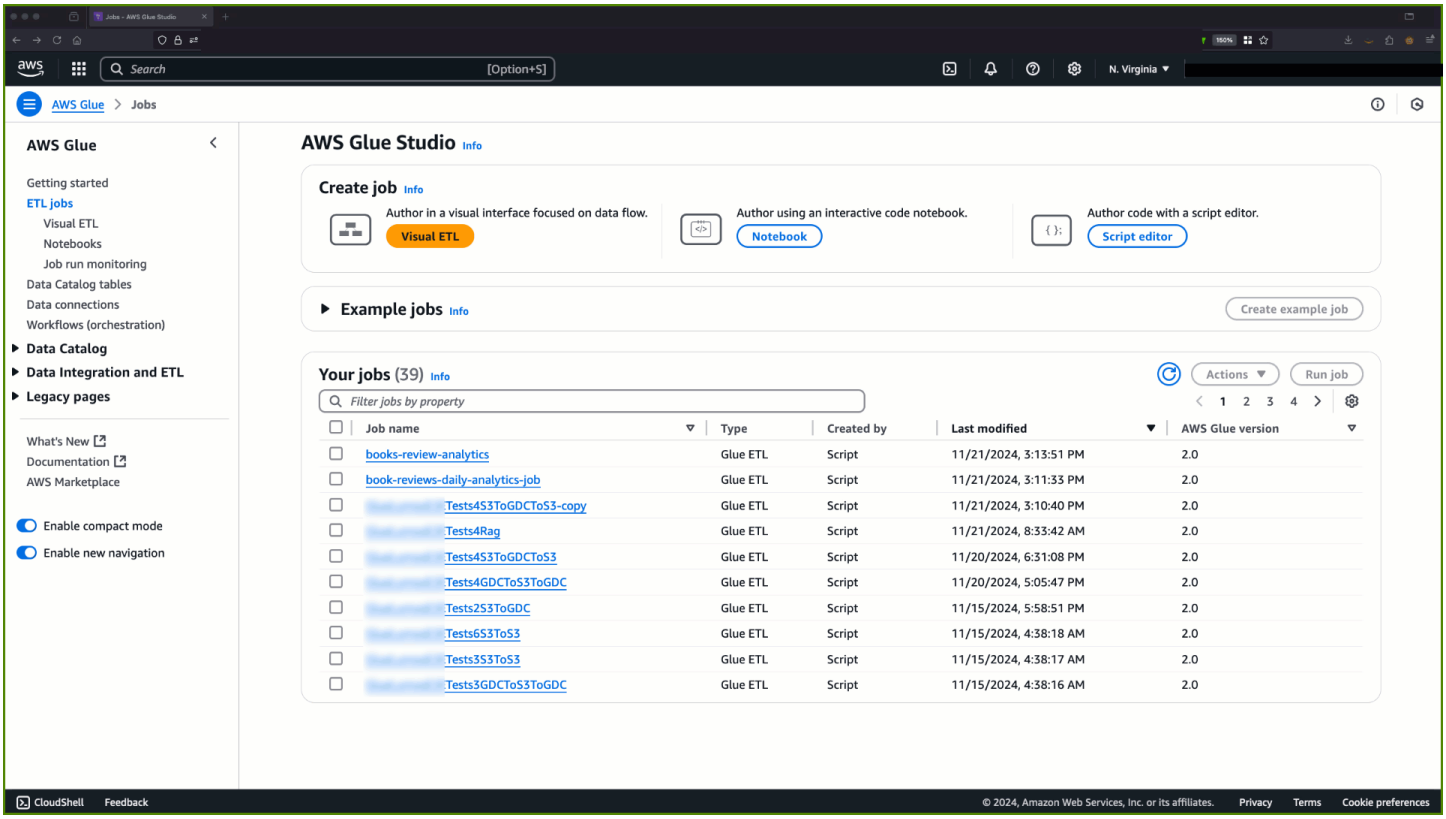
다음은 커넥터 업그레이드입니다.

드라이버	AWS Glue 3.0의 커넥터 버전	AWS Glue 4.0의 커넥터 버전
MongoDB	3.0.0	10.0.4
Hudi	0.10.1	0.12.1
Delta Lake	1.0.0	2.1.0
Iceberg	0.13.1	1.0.0
DynamoDB	1.11	1.12

AWS Glue에서 Apache Spark에 대한 생성형 AI 업그레이드

Apache Spark 평가판의 생성형 AI 업그레이드는 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오리건), 아시아 태평양(도쿄), 아시아 태평양(시드니)과 같은 AWS 리전에서 AWS Glue 버전 4.0 및 5.0에 대해 사용할 수 있습니다. 평가판 기능은 변경될 수 있습니다.

AWS Glue에서 Spark 업그레이드를 통해 데이터 엔지니어와 개발자는 생성형 AI를 사용하여 기존 AWS Glue Spark 작업을 최신 Spark 릴리스로 업그레이드 및 마이그레이션할 수 있습니다. 데이터 엔지니어는 이를 사용하여 AWS Glue Spark 작업을 스캔하고, 업그레이드 계획을 생성하며, 계획을 실행하고, 출력을 검증할 수 있습니다. Spark 스크립트, 구성, 종속 항목, 메서드 및 기능을 식별하고 업데이트하는 차별화되지 않은 작업을 자동화하여 Spark 업그레이드의 시간과 비용을 줄입니다.



작동 방법

업그레이드 분석을 사용할 때 AWS Glue는 작업 코드에서 버전 및 구성 간 차이를 식별하여 업그레이드 계획을 생성합니다. 업그레이드 계획에서는 모든 코드 변경 사항과 필요한 마이그레이션 단계를 자세히 설명합니다. 다음으로, AWS Glue는 환경에서 업그레이드된 애플리케이션을 빌드하고 실행하여 변경 사항을 검증하고 작업을 마이그레이션할 수 있는 코드 변경 사항 목록을 생성합니다. 제안된 변경 사항을 자세히 설명하는 요약과 함께 업데이트된 스크립트를 볼 수 있습니다. 자체 테스트를 실행한 후 변경 사항을 수락하면 AWS Glue 작업이 새 스크립트를 사용하여 최신 버전으로 자동 업데이트됩니다.

업그레이드 분석 프로세스는 워크로드 및 작업의 복잡성에 따라 완료하는 데 다소 시간이 걸릴 수 있습니다. 업그레이드 분석 결과는 지정된 Amazon S3 경로에 저장되며, 이를 검토하여 업그레이드 및 잠재적 호환성 문제를 파악할 수 있습니다. 업그레이드 분석 결과를 검토한 후 업그레이드하기 전에 실제 업그레이드를 진행할지 아니면 필요에 따라 작업을 변경할지 결정할 수 있습니다.

사전 조건

생성형 AI를 사용하여 AWS Glue에서 작업을 업그레이드하려면 다음 사전 조건이 필요합니다.

- AWS Glue 2 PySpark 작업 - AWS Glue 2 작업만 AWS Glue 5로 업그레이드할 수 있습니다.

- 분석을 시작하고, 결과를 검토하며, 작업을 업그레이드하려면 IAM 권한이 필요합니다. 자세한 내용은 아래 권한 섹션의 예제를 참조하세요.
- AWS KMS를 사용하여 분석 아티팩트 또는 서비스를 암호화하거나 분석에 사용되는 데이터를 암호화하는 경우 추가 AWS KMS 권한이 필요합니다. 자세한 내용은 아래 AWS KMS 정책 섹션의 예제를 참조하세요.

권한

새 업그레이드 분석을 시작하려면 다음 권한이 필요합니다.

1. 다음 권한으로 직접 호출자의 IAM 정책을 업데이트하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["glue:StartJobUpgradeAnalysis",
                 "glue:StartJobRun",
                 "glue:GetJobRun",
                 "glue:GetJob",
                 "glue:BatchStopJobRun"
                 ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:job/jobName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": [
        "<s3 script location associated with the job>"
      ]
    },
    {
      "Effect": "Allow",
      "Action": ["s3:PutObject"],
      "Resource": [
        "<result s3 path provided on API>"
      ]
    },
    {
```

```
    "Effect": "Allow",
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey",
      ],
     "Resource": "<key-arn-passed in the API>"
    }
  ]
}
```

> **Note**
>
> 결과 아티팩트 암호화에 대한 키 및 서비스 메타데이터 암호화에 대한 키와 같은 두 가지 서로 다른 AWS KMS 키를 사용하는 경우 두 키에 대해 유사한 정책이 정책에 포함되어야 합니다.

2. 다음 인라인 정책을 포함하도록 업그레이드하려는 작업의 실행 역할을 업데이트하세요.

```
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": [
        "ARN of the Amazon S3 path provided on API",
        "ARN of the Amazon S3 path provided on API/*"
      ]
    }
```

예를 들어 Amazon S3 경로(`s3://amzn-s3-demo-bucket/upgraded-result`)를 사용하는 경우 정책은 다음과 같습니다.

```
{
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/upgraded-result/",
        "arn:aws:s3:::amzn-s3-demo-bucket/upgraded-result/*"
      ]
    }
```

분석 세부 정보를 검색하려면 다음 권한이 필요합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["glue:GetJobUpgradeAnalysis"],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:job/jobName"
      ]
    }
  ]
}
```

진행 중인 분석을 중지하려면 다음 권한이 필요합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["glue:StopJobUpgradeAnalysis",
                 "glue:BatchStopJobRun"
                 ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:job/jobName"
      ]
    }
  ]
}
```

특정 작업에 대해 제출된 모든 분석을 나열하려면 다음 권한이 필요합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["glue:ListJobUpgradeAnalyses"],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:job/jobName"
```

```
      ]
    }
  ]
}
```

분석의 변경 사항을 수락하고 작업을 업그레이드하려면 다음 권한이 필요합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["glue:UpdateJob",
                 "glue:UpgradeJob"
                 ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:job/jobName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:PassRole"],
      "Resource": [
        "<Role arn associated with the job>"
      ]
    }
  ]
}
```

AWS KMS 정책

분석을 시작할 때 사용자 지정 AWS KMS 키를 전달하려면 다음 섹션을 참조하여 AWS KMS 키에 대한 적절한 권한을 구성하세요.

서비스 메타데이터 암호화 AWS KMS 키 구성:

키를 전달하려면 권한(암호화/복호화)이 필요합니다. 아래 정책 예제에서 `<IAM Customer caller ARN>`에 의해 지정된 AWS 계정 또는 역할이 허용된 작업을 수행할 수 있습니다.

- kms:Decrypt는 지정된 AWS KMS 키를 사용하여 복호화를 허용합니다.
- kms:GenerateDataKey는 지정된 AWS KMS 키를 사용하여 데이터 키 생성을 허용합니다.

```
{
    "Effect": "Allow",
    "Principal":{
        "AWS": "<IAM Customer caller ARN>"
    },
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey",
    ],
    "Resource": "<key-arn-passed-on-start-api>"
}
```

키의 암호화 및 복호화 모두에 AWS KMS 키를 사용할 수 있는 권한을 AWS Glue에 부여해야 합니다.

```
{
    "Effect": "Allow",
    "Principal":{
       "Service": "glue.amazonaws.com"
     },
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey",
      ],
     "Resource": "<key-arn>",
     "Condition": {
       "StringLike": {
         "aws:SourceArn": "arn:aws:glue:<region>:<aws_account_id>:job/job-name"
       }
     }
 }
```

AWS KMS 키를 사용하여 결과 아티팩트 암호화 구성:

이 정책을 통해 AWS KMS 키에 대한 암호화 및 복호화 권한을 모두 보유합니다.

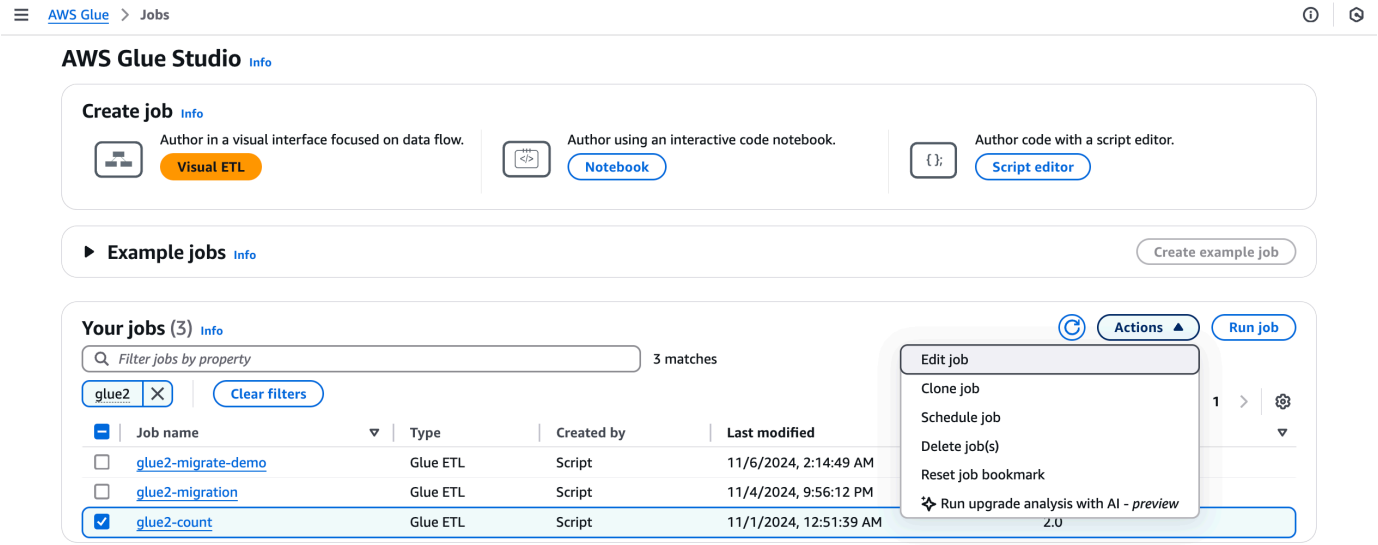
```
{
    "Effect": "Allow",
    "Principal":{
        "AWS": "<IAM Customer caller ARN>"
    },
    "Action": [
      "kms:Decrypt",
```

```
      "kms:GenerateDataKey",
    ],
    "Resource": "<key-arn-passed-on-start-api>"
}
```

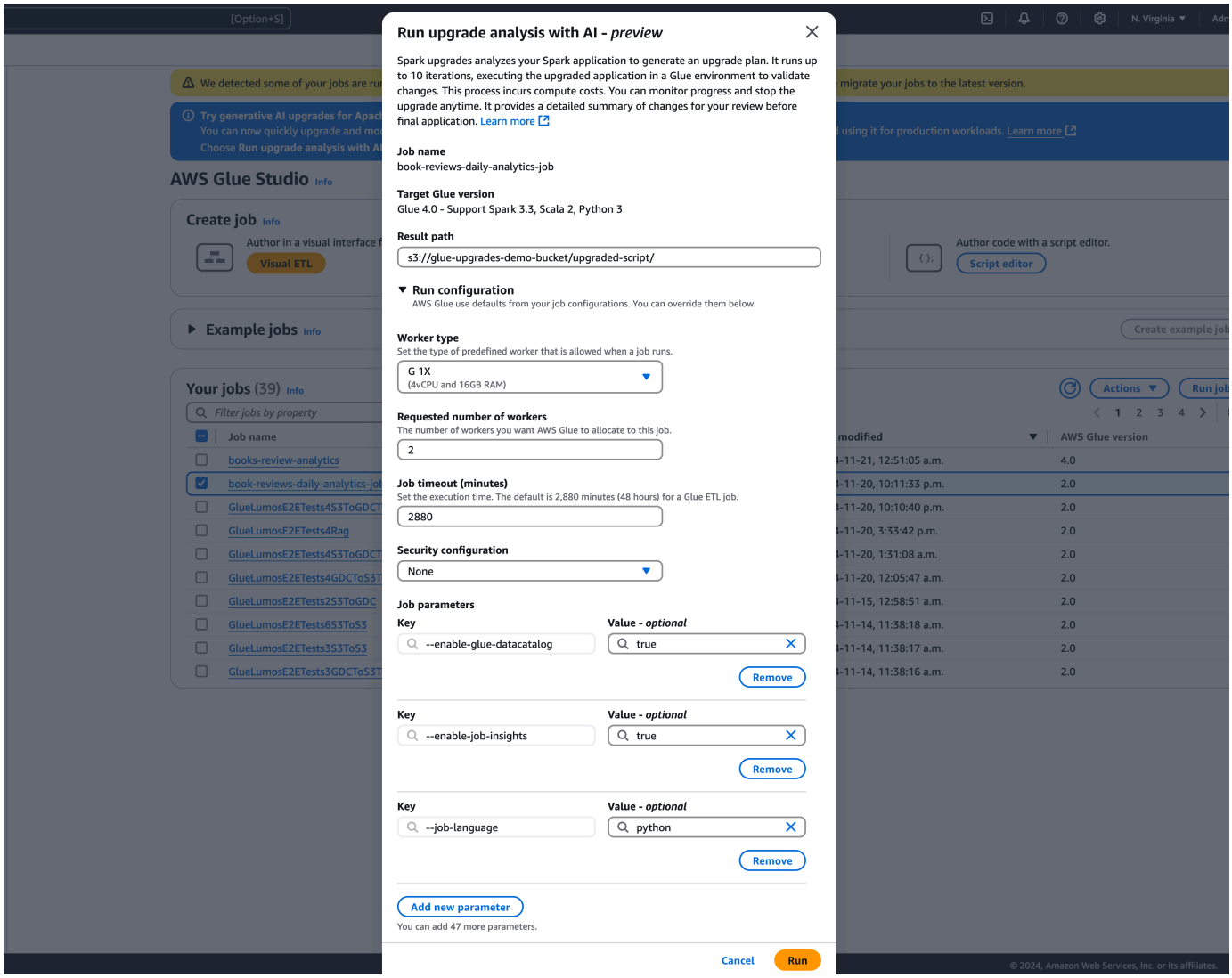
업그레이드 분석 실행 및 업그레이드 스크립트 적용

업그레이드 분석을 실행할 수 있습니다. 그러면 작업 보기에서 선택한 작업에서 업그레이드 계획이 생성됩니다.

1. 작업에서 AWS Glue 2.0 작업을 선택한 다음, 작업 메뉴에서 업그레이드 분석 실행을 선택하세요.



2. 모달에서 생성된 업그레이드 계획을 결과 경로에 저장할 경로를 선택하세요. 액세스하고 쓸 수 있는 Amazon S3 버킷이어야 합니다.



3. 필요한 경우 다음과 같은 추가 옵션을 구성하세요.

- 실행 구성 - 선택 사항: 실행 구성은 업그레이드 분석 중에 수행된 검증 실행의 다양한 측면을 사용자 지정할 수 있는 선택적 설정입니다. 이 구성은 업그레이드된 스크립트를 실행하는 데 사용되며 여기에서 컴퓨팅 환경 속성(작업자 유형, 작업자 수 등)을 선택할 수 있습니다. 변경 사항을 검토 및 수락하고 프로덕션 환경에 적용하기 전에 비프로덕션 개발자 계정을 사용하여 샘플 데이터세트에 대한 검증을 실행해야 합니다. 실행 구성에는 다음과 같은 사용자 지정 가능한 파라미터가 포함됩니다.
  - 작업자 유형: 검증 실행에 대해 사용할 작업자 유형을 지정하여 요구 사항에 따라 적절한 컴퓨팅 리소스를 선택할 수 있습니다.
  - 작업자 수: 검증 실행에 대해 프로비저닝할 작업자 수를 정의하여 워크로드 요구 사항에 따라 리소스 규모를 조정할 수 있습니다.

- 작업 제한 시간(분): 이 파라미터를 사용하면 검증 실행의 시간 제한을 설정할 수 있으므로 지정된 기간 후에 작업이 종료되어 과도한 리소스 소비를 방지할 수 있습니다.
- 보안 구성: 검증 실행 중에 데이터와 리소스를 보호하도록 암호화 및 액세스 제어와 같은 보안 설정을 구성할 수 있습니다.
- 추가 작업 파라미터: 필요한 경우 새 작업 파라미터를 추가하여 검증 실행을 위한 실행 환경을 추가로 사용자 지정할 수 있습니다.

실행 구성을 활용하여 특정 요구 사항에 맞게 검증 실행을 조정할 수 있습니다. 예를 들어 더 작은 데이터세트를 사용하도록 검증 실행을 구성할 수 있습니다. 그러면 분석을 더 빠르게 완료하고 비용을 최적화할 수 있습니다. 이 접근 방식을 사용하면 검증 단계 중에 리소스 사용률과 관련 비용을 최소화하면서 업그레이드 분석을 효율적으로 수행할 수 있습니다.

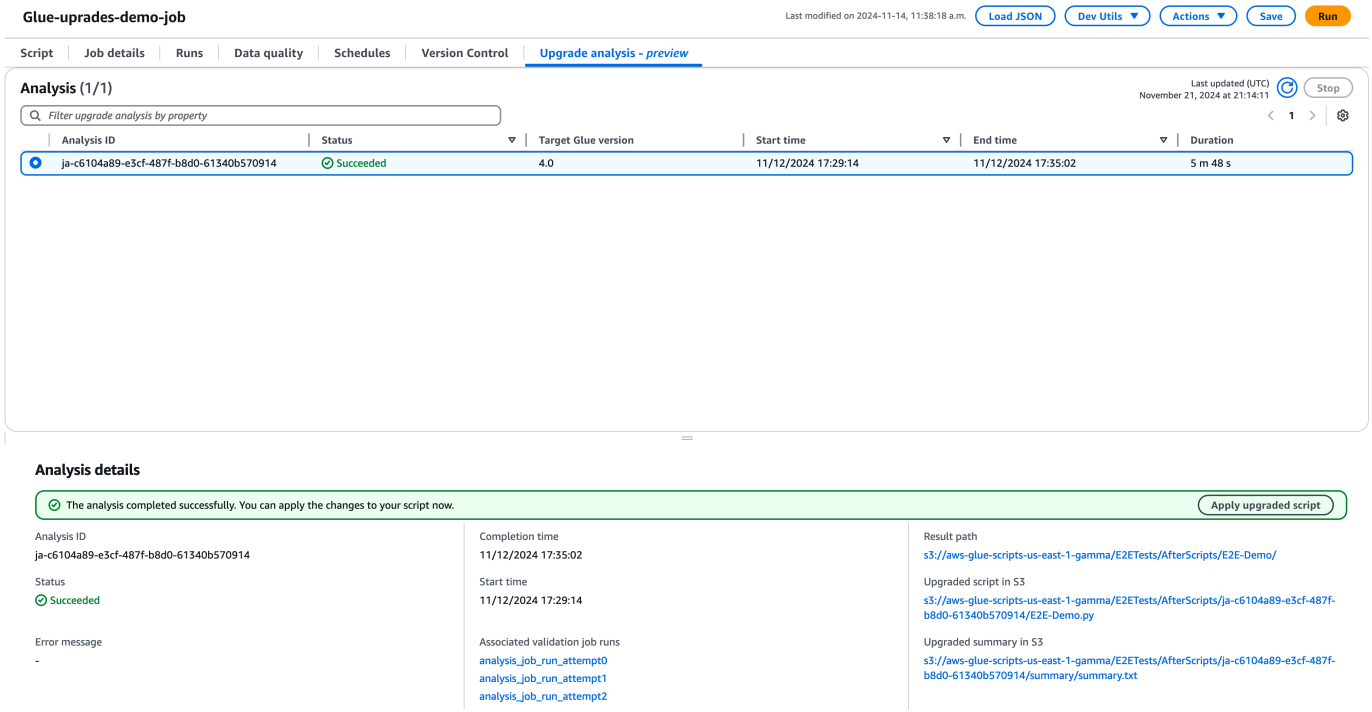
- 암호화 구성 - 선택 사항:
  - 업그레이드 아티팩트 암호화 활성화: 결과 경로에 데이터를 쓸 때 저장 데이터 암호화를 활성화합니다. 업그레이드 아티팩트를 암호화하지 않으려면 이 옵션을 선택하지 않은 상태로 둡니다.
  - 서비스 메타데이터 암호화 사용자 지정: 서비스 메타데이터는 기본적으로 AWS 소유 키를 사용하여 암호화됩니다. 암호화에 자체 키를 사용하려면이 옵션을 선택하세요.

4. 실행을 선택하여 업그레이드 분석을 시작하세요. 분석이 실행되는 동안 업그레이드 분석 탭에서 결과를 볼 수 있습니다. 분석 세부 정보 창에는 분석에 대한 정보와 업그레이드 아티팩트에 대한 링크가 표시됩니다.

   - 결과 경로 - 결과 요약 및 업그레이드 스크립트가 저장되는 위치입니다.
   - Amazon S3에서의 업그레이드된 스크립트 - Amazon S3에서 업그레이드 스크립트 위치. 업그레이드를 적용하기 전에 스크립트를 볼 수 있습니다.
   - Amazon S3에서의 업그레이드 요약 - Amazon S3에서 업그레이드 요약 위치. 업그레이드를 적용하기 전에 업그레이드 요약을 볼 수 있습니다.

5. 업그레이드 분석이 완료되면 업그레이드된 스크립트 적용을 선택하여 업그레이드 스크립트를 적용하여 작업을 자동으로 업그레이드할 수 있습니다.

   적용 후에는 AWS Glue 버전이 4.0으로 업데이트됩니다. 스크립트 탭에서 새 스크립트를 볼 수 있습니다.



업그레이드 요약 이해

이 예제에서는 AWS Glue 작업을 버전 2.0에서 버전 4.0으로 업그레이드하는 프로세스를 보여줍니다. 샘플 작업은 Amazon S3 버킷에서 제품 데이터를 읽고 Spark SQL을 사용하여 여러 변환을 데이터에 적용한 다음, 변환된 결과를 Amazon S3 버킷에 다시 저장합니다.

원본 코드(AWS Glue 2.0) - 업그레이드 전

```
from awsglue.transforms import *
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from pyspark.sql.types import *
from pyspark.sql.functions import *
from awsglue.job import Job
import json
from pyspark.sql.types import StructType

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
```

```
gdc_database = "s3://aws-glue-scripts-us-east-1-gamma/demo-database/"
schema_location = (
    "s3://aws-glue-scripts-us-east-1-gamma/DataFiles/"
)

products_schema_string = spark.read.text(
    f"{schema_location}schemas/products_schema"
).first()[0]

product_schema = StructType.fromJson(json.loads(products_schema_string))

products_source_df = (
    spark.read.option("header", "true")
    .schema(product_schema)
    .option(
        "path",
        f"{gdc_database}products/",
    )
    .csv(f"{gdc_database}products/")
)

products_source_df.show()
products_temp_view_name = "spark_upgrade_demo_product_view"
products_source_df.createOrReplaceTempView(products_temp_view_name)

query = f"select {products_temp_view_name}.*, format_string('%0$s-%0$s', category,
 subcategory) as unique_category from {products_temp_view_name}"
products_with_combination_df = spark.sql(query)
products_with_combination_df.show()

products_with_combination_df.createOrReplaceTempView(products_temp_view_name)
product_df_attribution = spark.sql(
    f"""
SELECT *,
unbase64(split(product_name, ' ')[0]) as product_name_decoded,
unbase64(split(unique_category, '-')[1]) as subcategory_decoded
FROM {products_temp_view_name}
"""
)
product_df_attribution.show()


product_df_attribution.write.mode("overwrite").option("header", "true").option(
    "path", f"{gdc_database}spark_upgrade_demo_product_agg/"
```

```
).saveAsTable("spark_upgrade_demo_product_agg", external=True)

spark_upgrade_demo_product_agg_table_df = spark.sql(
    f"SHOW TABLE EXTENDED in default like 'spark_upgrade_demo_product_agg'"
)
spark_upgrade_demo_product_agg_table_df.show()
job.commit()
```

새 코드(Glue 4.0) - 업그레이드 후

```
from awsglue.transforms import *
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from pyspark.sql.types import *
from pyspark.sql.functions import *
from awsglue.job import Job
import json
from pyspark.sql.types import StructType

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
# change 1
spark.conf.set("spark.sql.adaptive.enabled", "false")
# change 2
spark.conf.set("spark.sql.legacy.pathOptionBehavior.enabled", "true")
job = Job(glueContext)

gdc_database = "s3://aws-glue-scripts-us-east-1-gamma/demo-database/"
schema_location = (
    "s3://aws-glue-scripts-us-east-1-gamma/DataFiles/"
)

products_schema_string = spark.read.text(
    f"{schema_location}schemas/products_schema"
).first()[0]

product_schema = StructType.fromJson(json.loads(products_schema_string))

products_source_df = (
    spark.read.option("header", "true")
    .schema(product_schema)
    .option(
```

```
        "path",
        f"{gdc_database}products/",
    )
    .csv(f"{gdc_database}products/")
)

products_source_df.show()
products_temp_view_name = "spark_upgrade_demo_product_view"
products_source_df.createOrReplaceTempView(products_temp_view_name)

# change 3
query = f"select {products_temp_view_name}.*, format_string('%1$s-%1$s', category,
 subcategory) as unique_category from {products_temp_view_name}"
products_with_combination_df = spark.sql(query)
products_with_combination_df.show()

products_with_combination_df.createOrReplaceTempView(products_temp_view_name)
# change 4
product_df_attribution = spark.sql(
    f"""
SELECT *,
try_to_binary(split(product_name, ' ')[0], 'base64') as product_name_decoded,
try_to_binary(split(unique_category, '-')[1], 'base64') as subcategory_decoded
FROM {products_temp_view_name}
"""
)
product_df_attribution.show()


product_df_attribution.write.mode("overwrite").option("header", "true").option(
    "path", f"{gdc_database}spark_upgrade_demo_product_agg/"
).saveAsTable("spark_upgrade_demo_product_agg", external=True)

spark_upgrade_demo_product_agg_table_df = spark.sql(
    f"SHOW TABLE EXTENDED in default like 'spark_upgrade_demo_product_agg'"
)
spark_upgrade_demo_product_agg_table_df.show()
job.commit()
```

분석 요약 설명

```
In Spark 3.2, spark.sql.adaptive.enabled is enabled by default. To restore the behavior before
Spark 3.2,
you can set spark.sql.adaptive.enabled to false.


In Spark 3.1, path option cannot coexist when the following methods are called with path
parameter(s): DataFrameReader.load(), DataFrameWriter.save(), DataStreamReader.load(), or
DataStreamWriter.start(). In addition, paths option cannot coexist for DataFrameReader.load().
For example, spark.read.format(csv).option(path, /tmp).load(/tmp2) or spark.read.option(path, /
tmp).csv(/tmp2) will throw org.apache.spark.sql.AnalysisException. In Spark version 3.0 and
below, path option is overwritten if one path parameter is passed to above methods; path option
is added to the overall paths if multiple path parameters are passed to DataFrameReader.load().
To restore the behavior before Spark 3.1, you can set
spark.sql.legacy.pathOptionBehavior.enabled to true.


Since Spark 3.3, the `strfmt` in `format_string(strfmt, obj, ...)` and `printf(strfmt,
obj, ...)` will no longer support to use 0$ to specify the first argument, the first argument
should always reference by 1$ when use argument index to indicating the position of the argument
in the argument list.


Since Spark 3.3, the unbase64 function throws error for a malformed str input. Use
try_to_binary(<str>, 'base64') to tolerate malformed input and return NULL instead. In Spark 3.2
and earlier, the unbase64 function returns a best-efforts result for a malformed str input.
```

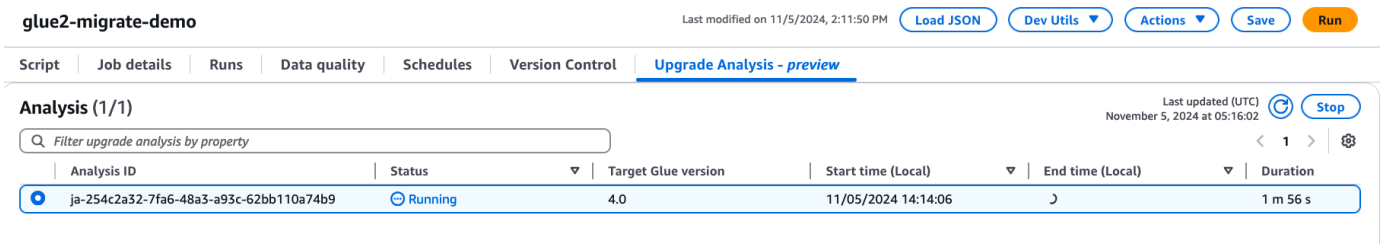
요약에 따르면 스크립트를 AWS Glue 2.0에서 AWS Glue 4.0으로 업그레이드하기 위해 AWS Glue에서 제안하는 네 가지 변경 사항이 있습니다.

1. Spark SQL 구성(spark.sql.adaptive.enabled): 이 변경 사항은 Spark SQL 적응형 쿼리 실행을 위한 새 기능으로 애플리케이션 동작을 복원합니다. Spark 3.2부터 도입됩니다. 이 구성 변경을 검사하고 기본 설정에 따라 추가로 활성화하거나 비활성화할 수 있습니다.
2. DataFrame API 변경: 경로 옵션은 `load()`와 같은 다른 DataFrameReader 작업과 공존할 수 없습니다. 이전 동작을 유지하기 위해 AWS Glue는 스크립트를 업데이트하여 새 SQL 구성(spark.sql.legacy.pathOptionBehavior.enabled)을 추가했습니다.
3. Spark SQL API 변경: `format_string(strfmt, obj, ...)`에서 `strfmt`의 동작이 첫 번째 인수로 `0$`를 허용하지 않도록 업데이트되었습니다. 호환성을 보장하기 위해 AWS Glue는 대신 첫 번째 인수로 `1$`를 사용하도록 스크립트를 수정했습니다.
4. Spark SQL API 변경: `unbase64` 함수에서는 잘못된 형식의 문자열 입력을 허용하지 않습니다. 이전 동작을 유지하기 위해 AWS Glue는 `try_to_binary` 함수를 사용하도록 스크립트를 업데이트했습니다.

진행 중인 업그레이드 분석 중지

진행 중인 업그레이드 분석을 취소하거나 분석을 중지할 수 있습니다.

1. 업그레이드 분석 탭을 선택하세요.
2. 실행 중인 작업을 선택한 다음, 중지를 선택하세요. 그러면 분석이 중지됩니다. 그런 다음, 동일한 작업에서 다른 업그레이드 분석을 실행할 수 있습니다.



고려 사항

평가판 기간에 Spark 업그레이드를 사용하기 시작하면 서비스의 최적 사용을 위해 고려할 몇 가지 중요한 측면이 있습니다.

- 서비스 범위 및 제한 사항: 평가판 릴리스에서는 AWS Glue 버전 2.0에서 버전 5.0으로 PySpark 코드를 업그레이드하는 데 중점을 둡니다. 현재 서비스는 추가 라이브러리 종속 항목에 의존하지 않는 PySpark 코드를 처리합니다. AWS 계정에서 동시에 최대 10개의 작업에 대해 자동화된 업그레이드를 실행할 수 있으므로 시스템 안정성을 유지하면서 여러 작업을 효율적으로 업그레이드할 수 있습니다.
  - PySpark 작업만 지원됩니다.
  - 업그레이드 분석은 24시간 후에 제한 시간이 초과됩니다.
  - 한 작업에 대해 한 번에 하나의 활성 업그레이드 분석만 실행할 수 있습니다. 계정 수준에서는 최대 10개의 활성 업그레이드 분석을 동시에 실행할 수 있습니다.
- 업그레이드 프로세스 중 비용 최적화: Spark 업그레이드는 생성형 AI를 사용하여 여러 반복을 통해 업그레이드 계획을 검증합니다. 이대 계정에서 각 반복이 AWS Glue 작업으로 실행되므로 비용 효율성을 위해 검증 작업 실행 구성을 최적화하는 것이 중요합니다. 이를 위해 업그레이드 분석을 시작할 때 다음과 같이 실행 구성을 지정하는 것이 좋습니다.
  - 비프로덕션 개발자 계정을 사용하고, Spark 업그레이드를 사용하는 검증에 대해 프로덕션 데이터를 나타내지만 크기가 더 작은 샘플 모의 데이터세트를 선택합니다.
  - G.1X 작업자와 같은 적절한 크기의 컴퓨팅 리소스를 사용하고 샘플 데이터를 처리하기 위해 적절한 수의 작업자를 선택합니다.

- 워크로드에 따라 리소스를 자동으로 조정하려면 해당하는 경우 AWS Glue 작업 오트 스케일링을 활성화합니다.

  예를 들어 프로덕션 작업에서 20G.2X 작업자를 사용하여 테라바이트 단위의 데이터를 처리하는 경우 검증을 위해 2G.2X 작업자를 사용하고 오토 스케일링이 활성화된 몇 기가바이트 단위의 대표 데이터를 처리하도록 업그레이드 작업을 구성할 수 있습니다.
- 평가판 모범 사례: 평가판 기간에 비프로덕션 작업으로 업그레이드 여정을 시작하는 것이 좋습니다. 이 접근 방식을 사용하면 업그레이드 워크플로를 숙지하고 서비스에서 다양한 유형의 Spark 코드 패턴을 처리하는 방법을 이해할 수 있습니다.
- 경보 및 알림: 작업에서 생성형 AI 업그레이드 기능을 사용하는 경우 실패한 작업 실행에 대한 경보/알림이 꺼져 있는지 확인합니다. 업그레이드 프로세스 중에는 업그레이드된 아티팩트가 제공되기 전에 계정에서 최대 10회의 작업 실행에 실패할 수 있습니다.
- 이상 탐지 규칙: 업그레이드 중인 작업에서 이상 탐지 규칙도 끕니다. 업그레이드 검증이 진행되는 동안 중간 작업 실행 중에 출력 폴더에 기록된 데이터가 예상되는 형식이 아닐 수 있습니다.
- 멱등성 작업에 업그레이드 분석 사용: 멱등성 작업에 업그레이드 분석을 사용하여 각 후속 검증 작업 실행 시도가 이전 시도와 유사하고 문제가 발생하지 않도록 합니다. 멱등성 작업은 동일한 입력 데이터로 여러 번 실행할 수 있는 작업이며 매번 동일한 출력을 생성합니다. AWS Glue에서 Apache Spark용 생성형 AI 업그레이드를 사용하는 경우 서비스는 검증 프로세스의 일부로 작업을 여러 번 반복합니다. 각 반복 중에 Spark 코드 및 구성을 변경하여 업그레이드 계획을 검증합니다. Spark 작업이 멱등성이 아닌 경우 동일한 입력 데이터로 여러 번 실행하면 문제가 발생할 수 있습니다.

Spark 업그레이드의 교차 리전 추론

Spark 업그레이드는 Amazon Bedrock으로 구동되며 교차 리전 추론(CRIS)을 활용합니다. CRIS를 사용하면 Spark 업그레이드는 추론 요청을 처리하고 사용 가능한 컴퓨팅 리소스 및 모델 가용성을 극대화하며 최상의 고객 경험을 제공하기 위해 사용자의 지역 내에서 최적의 리전을 자동으로 선택합니다(여기에서 자세한 설명 참조). 교차 리전 추론을 사용하는 데 드는 추가 비용은 없습니다.

교차 리전 추론 요청은 데이터가 원래 상주하는 지역의 일부인 AWS 리전 내에 보관됩니다. 예를 들어, 미국 내에서 이루어진 요청은 미국의 AWS 리전 내에 보관됩니다. 데이터는 기본 리전에만 저장되어 있지만 교차 리전 추론을 사용하는 경우 입력 프롬프트와 출력 결과가 기본 리전 외부로 이동할 수 있습니다. 모든 데이터는 Amazon의 보안 네트워크를 통해 암호화되어 전송됩니다.

AWS Glue에서 Spark 작업 사용

AWS Glue에서 Spark ETL 작업에 대한 정보를 제공합니다.

주제

- AWS Glue 작업에서 작업 파라미터 사용
- AWS Glue Spark 및 PySpark 작업
- AWS Glue에서 스트리밍 ETL 작업
- AWS Lake Formation FindMatches로 레코드 매칭
- Apache Spark 프로그램을 AWS Glue로 마이그레이션

AWS Glue 작업에서 작업 파라미터 사용

AWS Glue 작업을 생성할 때, `Role` 및 `WorkerType`과 같은 몇 가지 표준 필드를 설정합니다. `Argument` 필드(콘솔의 작업 파라미터)를 통해 추가 구성 정보를 제공할 수 있습니다. 이 필드에서 이 주제에 나열된 인수(파라미터)를 AWS Glue 작업에 제공할 수 있습니다.

AWS Glue 작업 API에 대한 자세한 내용은 the section called “작업” 섹션을 참조하세요.

> **Note**
>
> 작업 인수 크기는 최대 260KB로 제한됩니다. 인수 크기가 260KB보다 크면 검증 검사에서 오류가 발생합니다.

작업 파라미터 설정

Job Parameters(작업 파라미터) 제목 아래의 Job details(작업 세부 정보) 탭에서 콘솔을 통해 작업을 구성할 수 있습니다. 작업에서 `DefaultArguments` 또는 `NonOverridableArguments`를 설정하거나 작업 실행에서 `Arguments`를 설정하여 AWS CLI를 통해 작업을 구성할 수도 있습니다. 작업에 설정된 인수는 작업이 실행될 때마다 전달되지만, 작업 실행에 설정된 인수는 해당 개별 실행에 대해서만 전달됩니다.

예를 들어 다음은 작업 파라미터를 설정하기 위해 `--arguments`를 사용하여 작업을 실행하는 구문입니다.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py"'
```

작업 파라미터 액세스

AWS Glue 스크립트를 작성할 때 자체 코드의 동작을 변경하기 위해 작업 파라미터 값에 액세스하려고 합니다. 라이브러리에서 이를 수행하기 위한 도우미 메서드가 제공됩니다. 이러한 메서드는 작업 파라미터 값을 재정의하는 작업 실행 파라미터 값을 해석합니다. 여러 위치에 설정된 파라미터를 해석하는 경우 `NonOverridableArguments` 작업은 `Arguments` 작업 실행을 재정의하고 이 작업 실행은 `DefaultArguments` 작업을 재정의합니다.

Python에서:

Python 작업에서는 `getResolvedParameters` 함수가 제공됩니다. 자세한 내용은 the section called "getResolvedOptions 옵션 " 섹션을 참조하세요. 작업 파라미터는 `sys.argv` 변수에서 사용할 수 있습니다.

Scala에서:

Scala 작업에서는 `GlueArgParser` 객체가 제공됩니다. 자세한 내용은 the section called "GlueArgParser" 섹션을 참조하세요. 작업 파라미터는 `sysArgs` 변수에서 사용할 수 있습니다.

작업 파라미터 참조

AWS Glue에서는 작업 및 작업 실행에 대한 스크립트 환경을 설정하는 데 사용할 수 있는 다음과 같은 인수 이름을 인식합니다.

`--additional-python-modules`

설치할 Python 패키지 세트를 나타내는 쉼표로 구분된 목록입니다. PyPI에서 패키지를 설치하거나 사용자 지정 배포를 제공할 수 있습니다. PyPI 패키지 항목은 *`package`*==*`version`* 형식(대상 패키지의 PyPI 이름 및 버전)입니다. 사용자 지정 배포 항목은 배포에 대한 S3 경로입니다.

항목은 Python 버전 일치를 사용하여 패키지와 버전을 일치시킵니다. 즉, 2개의 등호(예: ==)를 사용해야 합니다. 다른 버전 일치 연산자도 있습니다. 자세한 내용을 알아보려면 PEP 440을 참조하세요.

모듈 설치 옵션을 `pip3`에 전달하려면 --python-modules-installer-option 파라미터를 사용합니다.

`--auto-scale-within-microbatch`

기본값은 true입니다. 이 파라미터는 스트리밍 데이터를 일련의 마이크로 배치로 처리하는 AWS Glue 스트리밍 작업에만 사용할 수 있으며, 이때 Auto Scaling을 활성화해야 합니다. 이 값을 false로 설정한 경우 완료된 마이크로 배치에 대한 배치 지속 시간의 지수 이동 평균을 계산하고 이 값을

창 크기와 비교하여 실행기 수의 스케일 업 또는 스케일 다운 여부를 결정합니다. 마이크로 배치가 완료될 때만 규모가 조정됩니다. 이 값을 true로 설정한 경우 마이크로 배치 중에 Spark 작업 수가 30초 동안 동일하게 유지되거나 현재 배치 처리가 창 크기보다 클 때 스케일 업됩니다. 실행기가 60초 넘게 유휴 상태이거나 배치 지속 시간의 지수 이동 평균이 낮으면 실행기 수가 감소합니다.

`--class`

Scala 스크립트 진입점으로써 Scala 클래스. 이는 `--job-language`가 `scala`로 설정된 경우에만 적용됩니다.

`--continuous-log-conversionPattern`

연속 로깅에 대해 활성화된 작업의 사용자 지정 변환 로그 패턴을 지정합니다. 변환 패턴은 드라이버 로그와 실행기 로그에만 적용됩니다. AWS Glue 진행률 표시줄에는 영향을 주지 않습니다.

`--continuous-log-logGroup`

연속 로깅에 사용되는 활성화된 작업의 사용자 정의 Amazon CloudWatch 로그 그룹 이름을 지정합니다.

`--continuous-log-logStreamPrefix`

연속 로깅에 사용되는 작업의 사용자 정의 CloudWatch 로그 스트림 접두사를 지정합니다.

`--customer-driver-env-vars` 및 **`--customer-executor-env-vars`**

이러한 파라미터는 각 작업자(드라이버 또는 실행자)에 대해 각각 운영 체제의 환경 변수를 설정합니다. AWS Glue 위에 플랫폼과 사용자 지정 프레임워크를 구축할 때 이러한 파라미터를 사용하면 사용자가 그 위에 작업을 작성할 수 있습니다. 이 두 플래그를 활성화하면 작업 스크립트 자체에 동일한 논리를 삽입하지 않고도 드라이버와 실행기에 각각 다른 환경 변수를 설정할 수 있습니다.

사용 예

다음은 이러한 파라미터를 사용하는 예제입니다.

```
"–customer-driver-env-vars", "CUSTOMER_KEY1=VAL1,CUSTOMER_KEY2=\"val2,val2 val2\"",
"–customer-executor-env-vars", "CUSTOMER_KEY3=VAL3,KEY4=VAL4"
```

작업 실행 인수에서 이를 설정하는 것은 다음 명령을 실행하는 것과 동일합니다.

드라이버에서:

- export CUSTOMER_KEY1=VAL1
- export CUSTOMER_KEY2="val2,val2 val2"

실행기에서:

- export CUSTOMER_KEY3=VAL3

그런 다음 작업 스크립트 자체에서 `os.environ.get("CUSTOMER_KEY1")` 또는 `System.getenv("CUSTOMER_KEY1")`을 사용하여 환경 변수를 검색할 수 있습니다.

적용된 구문

환경 변수를 정의할 때는 다음 표준을 준수하세요.

- 각 키에는 `CUSTOMER_ prefix`가 있어야 합니다.

  예를 들어 `"CUSTOMER_KEY3=VAL3,KEY4=VAL4"`의 경우 `KEY4=VAL4`는 무시되고 설정되지 않습니다.
- 각 키와 값 쌍은 단일 쉼표로 구분되어야 합니다.

  예: `"CUSTOMER_KEY3=VAL3,CUSTOMER_KEY4=VAL4"`
- '값'에 공백이나 쉼표가 있는 경우 따옴표 안에 정의해야 합니다.

  예: `CUSTOMER_KEY2=\"val2,val2 val2\"`

이 구문은 bash 환경 변수 설정의 표준에 가깝게 모델링합니다.

`--datalake-formats`

AWS Glue 3.0 이상 버전에서 지원됨

사용할 데이터 레이크 프레임워크를 지정합니다. AWS Glue는 지정한 프레임워크에 필요한 JAR 파일을 `classpath`에 추가합니다. 자세한 내용은 AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용 섹션을 참조하세요.

다음 값 중 하나 이상을 쉼표로 구분하여 지정할 수 있습니다.

- `hudi`
- `delta`
- `iceberg`

예를 들어 다음 인수를 전달하여 세 프레임워크를 모두 지정합니다.

```
'--datalake-formats': 'hudi,delta,iceberg'
```

--disable-proxy-v2

서비스 프록시를 비활성화하여 Amazon S3, CloudWatch, VPC를 통해 스크립트에서 시작하는 AWS Glue에 대한 AWS 서비스 호출을 허용합니다. 자세한 내용은 VPC를 통과하도록 AWS 호출 구성 을 참조하세요. 서비스 프록시를 비활성화하려면 이 파라미터 값을 `true`로 설정합니다.

--enable-auto-scaling

이 값을 `true`로 설정할 경우 Auto Scaling과 작업자별 청구를 사용하는 기능을 켭니다.

--enable-continuous-cloudwatch-log

AWS Glue 작업에 대한 실시간 연속 로깅을 사용합니다. CloudWatch에서 실시간 Apache Spark 작업 로그를 볼 수 있습니다.

--enable-continuous-log-filter

지속 로깅을 활성화하여 작업을 생성하거나 편집할 때 표준 필터(`true`) 또는 필터 없음(`false`)을 지정합니다. 표준 필터를 선택하면 유용하지 않은 Apache Spark 드라이버/실행기 및 Apache Hadoop YARN 하트비트 로그 메시지가 제거됩니다. 필터 없음을 선택하면 모든 로그 메시지가 제공됩니다.

--enable-glue-datacatalog

AWS Glue 데이터 카탈로그를 Apache Spark Hive 메타스토어로 사용할 수 있습니다. 이 기능을 활성화하려면 값을 `true`로 설정합니다.

--enable-job-insights

AWS Glue 작업 실행 인사이트로 추가 오류 분석 모니터링을 활성화합니다. 자세한 내용은 the section called “AWS Glue 작업 실행 인사이트를 사용한 모니터링”을 참조하세요. 기본적으로 이 값은 `true`로 설정되고 작업 실행 인사이트가 활성화됩니다.

이 옵션은 AWS Glue 버전 2.0과 3.0에서 사용할 수 있습니다.

--enable-lakeformation-fine-grained-access

AWS Glue 작업에 대한 세분화된 액세스 제어를 활성화합니다. 자세한 내용은 the section called “FGAC를 위한 Lake Formation” 섹션을 참조하세요.

--enable-metrics

이 작업 실행을 위해 작업 프로파일링용 측정치 수집을 활성화합니다. 이러한 지표는 AWS Glue콘솔 및 Amazon CloudWatch 콘솔에서 사용할 수 있습니다. 이 파라미터의 값은 관련이 없습니다. 이 기능을 활성화하려면 이 파라미터에 어떤 값이든 제공할 수 있지만 명확성을 위해 `true`를 권장합니다. 이 기능을 비활성화하려면 작업 구성에서 이 파라미터를 제거하세요.

--enable-observability-metrics

AWS 콘솔 및 Amazon CloudWatch 콘솔의 작업 실행 모니터링 페이지에서 각 작업 실행 내부에서 일어나는 일에 대한 인사이트를 생성하는 일련의 관찰성 지표를 사용할 수 있습니다. 이 기능을 활성화하려면 이 파라미터의 값을 true로 설정합니다. 이 기능을 비활성화하려면 `false`로 설정하거나 작업 구성에서 이 파라미터를 제거하세요.

--enable-rename-algorithm-v2

EMRFS 이름 바꾸기 알고리즘 버전을 버전 2로 설정합니다. Spark 작업에서 동적 파티션 덮어쓰기 모드를 사용하는 경우 중복 파티션이 생성될 가능성이 있습니다. 예를 들어, `s3://bucket/table/location/p1=1/p1=1`과 같은 중복 파티션으로 끝날 수 있습니다. 여기서 P1은 덮어쓰는 파티션입니다. 이름 바꾸기 알고리즘 버전 2가 이 문제를 해결합니다.

이 옵션은 AWS Glue 버전 1.0에서만 사용할 수 있습니다.

--enable-s3-parquet-optimized-committer

Amazon S3에 Parquet 데이터를 쓸 수 있도록 EMRFS S3 최적화 커미터를 사용합니다. AWS Glue 작업을 생성하거나 업데이트할 때 AWS Glue 콘솔을 통해 파라미터/값 페어를 제공할 수 있습니다. 값을 **`true`**로 설정하면 커미터가 활성화됩니다. 기본적으로 이 플래그는 AWS Glue 3.0에서 켜지고 AWS Glue 2.0에서는 꺼집니다.

자세한 내용은 EMRFS S3 최적화 커미터 사용을 참조하십시오.

--enable-spark-ui

`true`(으)로 설정하면 Spark UI를 사용하여 AWS Glue ETL 작업을 모니터링하고 디버그하는 기능을 켭니다.

--executor-cores

병렬로 실행할 수 있는 Spark 작업 수입니다. 이 옵션은 AWS Glue 3.0 이상에서 지원됩니다. 값은 작업자 유형에 있는 vCPU 수의 2배(`G.1X`의 경우 8, `G.2X`의 경우 16, `G.4X`의 경우 32, `G.8X`의 경우 64)를 초과해서는 안 됩니다. 이 구성을 업데이트할 때는 주의해야 합니다. 병렬 처리 성능을 높이면 메모리 및 디스크 부담이 커질 뿐 아니라 소스 및 대상 시스템에서 제한이 발생할 수 있으므로 작업 성능에 영향을 미칠 수 있습니다(예: Amazon RDS에서 동시 연결이 늘어남).

--extra-files

AWS Glue가 스크립트를 실행하기 전에 드라이버 노드의 스크립트 작업 디렉터리에 복사하는 구성 파일과 같은 추가 파일에 대한 Amazon S3 경로입니다. 여러 값은 쉼표(`,`)로 구분된 완전한 경로여야 합니다. 개별 파일만 지원되며 디렉터리 경로는 지원되지 않습니다. 이 옵션은 Python 쉘 작업 유형에서 지원되지 않습니다.

--extra-jars

Amazon S3는 AWS Glue가 드라이버 및 실행자에 복사하는 추가 파일에 대한 경로입니다. AWS 또한 Glue는 스크립트를 실행하기 전에 이러한 파일을 Java 클래스 경로에 추가합니다. 여러 값은 쉼표(,)로 구분된 완전한 경로여야 합니다. 확장자는 .jar와(과) 같을 필요는 없습니다

--extra-py-files

스크립트를 실행하기 전에 AWS Glue가 드라이버 노드의 Python 경로에 추가하는 추가 Python 모듈에 대한 Amazon S3 경로입니다. 여러 값은 쉼표(,)로 구분된 완전한 경로여야 합니다. 개별 파일만 지원되면 디렉터리 경로는 지원되지 않습니다.

--job-bookmark-option

작업 북마크 동작을 제어합니다. 다음 옵션 값을 설정할 수 있습니다.

-- job-bookmark-option 값	설명
job-bookmark-enable	이전에 처리된 데이터를 계속 추적합니다. 작업이 실행되면 마지막 체크포인트 이후의 새로운 데이터를 진행합니다.
job-bookmark-disable	항상 전체 데이터셋을 진행합니다. 이전에 실행된 작업의 출력값 관리는 여러분이 직접 수행해야 합니다.
job-bookmark-pause	마지막 북마크의 상태를 업데이트하지 않고 마지막으로 성공한 실행 이후의 증분 데이터 또는 다음 하위 옵션으로 식별된 범위의 데이터를 처리합니다. 이전에 실행된 작업의 출력값 관리는 여러분이 직접 수행해야 합니다. 두 하위 옵션은 다음과 같습니다. • **job-bookmark-from** <from-value> 는 이전에 마지막으로 성공한 실행까지 처리된 모든 입력을 나타내는 실행 ID이며, 지정된 실행 ID를 포함합니다. 해당하는 입력은 무시됩니다. • **job-bookmark-to** <to-value> 는 이전에 마지막으로 성공한 실행까지 처리된 모든 입력을 나타내는 실행 ID이며, 지정된 실행 ID를 포함합니다. <from-value> 에 의해 식별된 입력을 제외한 해당하는 입력이 작업에 의해 처리됩니다. 이 입력 이후의 입력도 처리에서 제외됩니다.

-- job-bookmark-option 값	설명
	이 옵션 세트를 지정하면 작업 북마크 상태가 업데이트되지 않습니다. 하위 옵션은 선택 사항입니다. 그러나 사용할 경우 두 하위 옵션 모두 제공해야 합니다.

예를 들어, 작업 북마크를 활성화하려면 다음 인수를 전달합니다.

```
'--job-bookmark-option': 'job-bookmark-enable'
```

`--job-language`

스크립트 프로그래밍 언어. 이 값은 `scala` 또는 `python`이어야 합니다. 이 파라미터가 존재하지 않을 경우 기본값은 `python`입니다.

`--python-modules-installer-option`

--additional-python-modules로 모듈을 설치할 때 `pip3`에 전달할 옵션을 정의하는 일반 텍스트 문자열입니다. 명령줄에서와 같이 공백으로 구분되고 접두사가 대시로 된 옵션을 제공합니다. 사용에 대한 자세한 내용을 알아보려면 the section called “pip를 사용하여 AWS Glue 2.0 이상에 추가 Python 모듈 설치”를 참조하세요.

> **Note**
>
> Python 3.9를 사용하는 경우 AWS Glue 작업에는 이 옵션이 지원되지 않습니다.

`--scriptLocation`

ETL 스크립트가 있는 Amazon Simple Storage Service(Amazon S3) 위치입니다(`s3://path/to/my/script.py` 형식). 이 파라미터는 `JobCommand` 객체에 설정된 스크립트 위치를 재정의합니다.

`--spark-event-logs-path`

Amazon S3 경로를 지정합니다. Spark UI 모니터링 기능을 사용할 때 AWS Glue는 Spark UI 이벤트를 저장하기 위한 임시 디렉터리로 사용할 수 있는 버킷으로 30초마다 이 Amazon S3 경로에 대한 Spark 이벤트 로그를 플러시합니다.

--TempDir

작업의 임시 디렉터리로 사용될 수 있는 버킷에 대한 Amazon S3 경로를 지정합니다.

예를 들어, 임시 디렉터리를 설정하려면 다음 인수를 전달합니다.

```
'--TempDir': 's3-path-to-directory'
```

> **Note**
>
> 리전에 버킷이 아직 없는 경우 AWS Glue에서 작업에 대한 임시 버킷을 생성합니다. 이 버킷은 퍼블릭 액세스를 허용할 수 있습니다. Amazon S3에서 버킷을 수정하여 퍼블릭 액세스 차단을 설정하거나 나중에 해당 리전의 모든 작업이 완료된 후 버킷을 삭제할 수 있습니다.

--use-postgres-driver

이 값을 `true`로 설정하면 Amazon Redshift JDBC 드라이버와의 충돌을 피하기 위해 클래스 경로에서 Postgres JDBC 드라이버의 우선순위를 지정합니다. 이 옵션은 AWS Glue 버전 2.0에서만 사용할 수 있습니다.

--user-jars-first

이 값을 `true`로 설정하면 클래스 경로에 있는 고객의 추가 JAR 파일의 우선순위를 지정합니다. 이 옵션은 AWS Glue 버전 2.0 이상에서만 사용할 수 있습니다.

--conf

Spark 구성 파라미터를 제어합니다. 고급 사용 사례에 해당됩니다.

--encryption-type

레거시 파라미터. 보안 구성을 사용하여 해당 동작을 구성해야 합니다. 보안 구성에 대한 자세한 내용은 the section called “AWS Glue에서 작성한 데이터 암호화” 섹션을 참조하세요.

다음 인수는 AWS Glue에서 내부적으로 사용하므로 절대 사용해서는 안 됩니다.

- `--debug` - AWS Glue 내부에서 사용됩니다. 설정해서는 안 됩니다.
- `--mode` - AWS Glue 내부에서 사용됩니다. 설정해서는 안 됩니다.

- `--JOB_NAME` - AWS Glue 내부에서 사용됩니다. 설정해서는 안 됩니다.
- `--endpoint` - AWS Glue 내부에서 사용됩니다. 설정해서는 안 됩니다.

AWS Glue에서는 사이트별 사용자 지정을 수행하기 위해 `sitecustomize`을(를) 사용하여 Python `site` 모듈에서 환경을 부트스트래핑할 수 있도록 지원합니다. 자체 초기화 함수의 부트스트래핑은 고급 사용 사례에만 권장되며, AWS Glue 4.0에서 효과가 가장 뛰어납니다.

환경 변수 접두사(`GLUE_CUSTOMER`)는 고객 전용으로 예약되어 있습니다.

AWS Glue Spark 및 PySpark 작업

AWS Glue은(는) Spark 및 PySpark 작업을 지원합니다. Spark 작업은 AWS Glue에서 관리하는 Apache Spark 환경에서 실행됩니다. 데이터를 배치로 처리합니다. 스트리밍 ETL 작업은 데이터 스트림에 대해 ETL을 수행한다는 점을 제외하고 Spark 작업과 유사합니다. 이는 Apache Spark Structured Streaming 프레임워크를 사용합니다. 스트리밍 ETL 작업에 일부 Spark 작업 기능을 사용할 수 없습니다.

다음 섹션에서는 AWS Glue Spark 및 PySpark 작업에 관한 정보를 제공합니다.

주제

- AWS Glue에서 Spark 작업에 대한 작업 속성 구성
- AWS Glue 콘솔에서 Spark 스크립트 편집
- 작업(레거시)
- 처리된 데이터를 작업 북마크로 추적
- Spark 셔플 데이터 저장
- AWS Glue Spark 작업 모니터링
- AWS Glue에서 Apache Spark에 대한 생성형 AI 문제 해결

AWS Glue에서 Spark 작업에 대한 작업 속성 구성

작업을 AWS Glue 콘솔에서 정의할 경우 AWS Glue 런타임 환경을 제어하기 위한 속성 값을 제공합니다.

Spark 작업에 대한 작업 속성 정의

다음 목록에서는 Spark 작업의 속성을 설명합니다. Python 셸 작업의 속성은 Python 셸 작업에 대한 작업 속성 정의 단원을 참조하십시오. 스트리밍 ETL 작업의 속성은 the section called “스트리밍 ETL 작업에 대한 작업 속성 정의” 단원을 참조하십시오.

속성은 AWS Glue 콘솔의 [작업 추가(Add job)] 마법사에 나타나는 순서대로 나열됩니다.

명칭

최대 255자 길이의 UTF-8 문자열을 제공합니다.

설명

최대 2,048자의 설명을 입력합니다(선택 사항).

IAM 역할

작업을 실행하고 데이터 스토어에 액세스하는 데 사용되는 리소스에 대한 권한 부여용 IAM 역할을 지정합니다. AWS Glue 작업을 실행하는 권한에 대한 자세한 내용은 AWS Glue의 Identity and Access Management 단원을 참조하십시오.

유형

ETL 작업의 유형입니다. 이는 선택한 데이터 소스 유형에 따라 자동으로 설정됩니다.

- Spark는 작업 명령 `glueetl`로 Apache Spark ETL 스크립트를 실행합니다.
- Spark Streaming은 `gluestreaming` 작업 명령으로 Apache Spark 스트리밍 ETL 스크립트를 실행합니다. 자세한 내용은 the section called “스트리밍 ETL 작업” 섹션을 참조하세요.
- Python 쉘에서 작업 명령 `pythonshell`로 Python 스크립트를 실행합니다. 자세한 내용은 AWS Glue에서 Python 쉘 작업에 대한 작업 속성 구성 섹션을 참조하세요.

AWS Glue 버전

AWS Glue 버전이 다음 표에 지정된 대로 작업에 사용할 수 있는 Apache Spark 및 Python의 버전을 결정합니다.

AWS Glue 버전	지원되는 Spark 및 Python 버전
5.0	• Spark 3.5.4 • Python 3.11
4.0	• Spark 3.3.0

AWS Glue 버전	지원되는 Spark 및 Python 버전
	• Python 3.10
3.0	• Spark 3.1.1 • Python 3.7

Language

ETL 스크립트의 코드는 작업 논리를 정의합니다. 이 스크립트는 Python 또는 Scala에 코딩될 수 있습니다. 작업이 실행하는 스크립트가 AWS Glue가 생성했는지 여러분이 제공했는지 선택할 수 있습니다. Amazon Simple Storage Service(Amazon S3)에서 스크립트 이름과 위치를 제공합니다. 경로의 스크립트 디렉터리와 동일한 이름의 파일이 없다는 것을 확인합니다. 스크립트 작성에 대한 자세한 내용은 AWS Glue 프로그래밍 안내서 단원을 참조하십시오.

작업자 유형

다음 작업자 유형을 사용할 수 있습니다.

AWS Glue 작업자에서 사용할 수 있는 리소스는 DPU 단위로 측정됩니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다.

- **`G.1X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 94GB 디스크에서 1개의 DPU(vCPU 4개, 메모리 16GB)에 매핑됩니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
- **`G.2X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 138GB 디스크에서 2개의 DPU(vCPU 8개, 메모리 32GB)에 매핑됩니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
- **`G.4X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 256GB 디스크(약 235GB의 여유 공간)에서 4개의 DPU(vCPU 16개, 메모리 64GB)에 매핑됩니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다.
- **`G.8X`** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 512GB 디스크(약 487GB의 여유 공간)에서 8개의 DPU(vCPU 32개, 메모리 128GB)에 매핑됩니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다.

- **G.025X** – 이 유형을 선택할 경우 Number of workers(작업자 수) 값도 제공합니다. 각 작업자는 84GB 디스크(약 34GB의 여유 공간)에서 0.25개의 DPU(vCPU 2개, 메모리 4GB)에 매핑됩니다. 볼륨이 낮은 스트리밍 작업에 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 AWS Glue 버전 3.0 이상 스트리밍 작업에만 사용할 수 있습니다.

ETL 작업을 실행하는 데 사용된 DPU의 개수에 따라 시간당 비용이 부과됩니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

AWS Glue 버전 1.0 이전 작업의 경우 콘솔을 사용하여 작업을 구성하고 [작업자 유형(Worker type)]을 [표준(Standard)]로 지정하면, [최대 용량(Maximum capacity)]이 설정되고 [작업자 수(Number of workers)]는 [최대 용량(Maximum capacity)] - 1이 됩니다. AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하는 경우, [최대 용량(Max capacity)] 파라미터를 지정하거나, [작업자 유형(Worker type)]과 [작업자 수(Number of workers)]를 모두 지정할 수 있습니다.

AWS Glue 버전 2.0 이상 작업의 경우 최대 용량을 지정할 수 없습니다. 대신 [작업자 유형(Worker type)] 및 [작업자 수(Number of workers)]를 지정해야 합니다.

G.4X 및 **G.8X** 작업자 유형은 다음 AWS 리전의 AWS Glue 버전 3.0 이상 Spark ETL 작업에만 사용할 수 있습니다. 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오리건), 캐나다(중부), 아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 유럽(스페인), 유럽(스톡홀름), 남아메리카(상파울루).

요청된 작업자 수

대부분의 작업자 유형에서 작업이 실행될 때 할당되는 작업자 수를 지정해야 합니다.

작업 북마크

작업 실행 시 AWS Glue가 상태 정보를 진행하는 방법을 지정합니다. 이전에 처리된 데이터를 기억하거나, 상태 정보를 업데이트하거나, 상태 정보를 무시할 수 있습니다. 자세한 내용은 the section called “처리된 데이터를 작업 북마크로 추적” 섹션을 참조하세요.

작업 실행 대기열

서비스 할당량 때문에 즉시 실행할 수 없는 경우 나중에 실행하기 위해 작업 실행을 대기열에 넣을지 여부를 지정합니다.

선택하면 해당 작업 실행에 대해 작업 실행 대기열이 활성화됩니다. 채워지지 않은 경우 작업 실행은 대기열에 포함되는 것으로 간주되지 않습니다.

이 설정이 작업 실행에 설정된 값과 일치하지 않으면 작업 실행 필드의 값이 사용됩니다.

유연한 실행

AWS Studio 또는 API를 사용하여 작업을 구성하면 표준 또는 유연한 작업 실행 클래스를 지정할 수 있습니다. 작업의 우선순위와 시간 민감도는 다를 수 있습니다. 표준 실행 클래스는 빠른 작업 시작 및 전용 리소스가 필요한 시간에 민감한 워크로드에 적합합니다.

유연한 실행 클래스는 사전 프로덕션 작업, 테스트 및 일회성 데이터 로드와 같은 긴급하지 않은 작업에 적합합니다. 유연한 작업 실행은 AWS Glue 버전 3.0 이상 및 `G.1X` 또는 `G.2X` 작업자 유형을 사용하는 작업에 지원됩니다.

유연한 작업 실행은 한 번에 실행하는 작업자 수를 기준으로 요금이 청구됩니다. 유연한 작업 실행에 대한 작업자 수를 추가하거나 제거할 수 있습니다. `Max Capacity*Execution Time`의 간단한 계산으로 청구하는 대신 각 작업자는 작업 실행 중에 실행된 시간으로 요금이 청구됩니다. 청구 금액은 `(Number of DPUs per worker*time each worker ran)`의 합입니다.

자세한 내용은 AWS Studio 또는 작업 및 작업 실행의 도움말 패널을 참조하십시오.

재시도 횟수

AWS Glue가 자동적으로 작업을 재시작하지 못하면 재시작하도록 시간 수를 0~10사이에서 지정합니다. 시간 제한에 도달한 작업은 다시 시작되지 않습니다.

작업 제한 시간

최대 실행 제한을 분으로 설정합니다. 최대 설정은 7일 또는 10,080분입니다. 그렇지 않으면 작업에서 예외가 발생합니다.

값을 비워 두면 제한 시간은 기본적으로 2,880분으로 설정됩니다.

제한 시간 값이 7일을 초과하는 기존 AWS Glue 작업은 기본적으로 7일로 설정됩니다. 예를 들어 배치 작업에 20일의 제한 시간을 지정했다면 7일째 되는 날에 작업이 중지됩니다.

> **작업 시간 초과에 대한 모범 사례**
>
> 작업은 실행 시간을 기준으로 요금이 청구됩니다. 예상치 못한 요금이 부과되지 않도록 하려면 작업의 예상 실행 시간에 적합한 제한 시간 값을 구성하세요.

고급 속성

스크립트 파일 이름

작업의 고유한 스크립트 이름입니다. Untitled job으로 지정할 수 없습니다.

스크립트 경로

스크립트의 Amazon S3 위치입니다. 경로는 `s3://bucket/prefix/path/` 형식이어야 합니다. 슬래시(/)로 끝나야 하며 파일을 포함하지 않아야 합니다.

작업 지표

이 작업이 실행될 때 Amazon CloudWatch 지표 생성을 설정하거나 해제합니다. 프로파일링 데이터를 보려면 이 옵션을 활성화해야 합니다. 지표를 설정하고 시각화하는 방법에 대한 자세한 내용은 작업 모니터링 및 디버깅 섹션을 참조하세요.

작업 관찰성 지표

이 작업이 실행될 때 추가 통합 가시성 CloudWatch 지표 생성을 사용하도록 설정합니다. 자세한 내용은 the section called “AWS Glue 관찰성 메트릭을 사용한 모니터링” 섹션을 참조하세요.

연속 로깅

Amazon CloudWatch에 대한 연속 로깅을 설정합니다. 이 옵션이 활성화되지 않으면 작업이 완료된 후에만 로그를 사용할 수 있습니다. 자세한 내용은 the section called “AWS Glue 작업의 로깅” 섹션을 참조하세요.

Spark UI

이 작업 모니터링에 Spark UI 사용을 설정합니다. 자세한 내용은 AWS Glue 작업을 위한 Apache Spark 웹 UI 사용 설정 섹션을 참조하세요.

Spark UI 로그 경로

Spark UI를 사용하도록 설정한 경우 로그를 작성할 경로입니다.

Spark UI 로깅 및 모니터링 구성

다음 옵션 중 하나를 선택하세요.

- 표준: AWS Glue 작업 실행 ID를 파일 이름으로 사용하여 로그를 작성합니다. AWS Glue 콘솔에서 Spark UI 모니터링을 켜세요.
- 레거시: 'spark-application-{timestamp}'를 파일 이름으로 사용하여 로그를 작성합니다. Spark UI 모니터링을 켜지 마세요.
- 표준 및 레거시: 표준 위치와 레거시 위치 모두에 로그를 작성합니다. AWS Glue 콘솔에서 Spark UI 모니터링을 켜세요.

최대 동시성

이 작업에 허용된 최대 동시 실행 수를 설정합니다. 기본 값은 1입니다. 이 임곗값에 도달하면 오류가 반환됩니다. 지정할 수 있는 최대값은 서비스 제한에 따라 통제됩니다. 예를 들어, 새로운

인스턴스가 시작된 후에도 이전 작업이 여전히 실행되고 있다면 오류를 반환하여 동일한 작업의 두 개의 인스턴스를 동시에 실행하지 않도록 방지합니다.

임시 경로

AWS Glue가 스크립트를 실행할 때 임시 중간 결과가 작성된 Amazon S3의 작업 디렉터리의 위치를 제공합니다. 경로의 임시 디렉터리와 동일한 이름의 파일이 없다는 것을 확인합니다. 이 디렉터리는 AWS Glue가 Amazon Redshift에 읽고 쓸 때 및 일부 AWS Glue 변환에 사용됩니다.

> **Note**
>
> 리전에 버킷이 아직 없는 경우 AWS Glue에서 작업에 대한 임시 버킷을 생성합니다. 이 버킷은 퍼블릭 액세스를 허용할 수 있습니다. Amazon S3에서 버킷을 수정하여 퍼블릭 액세스 차단을 설정하거나 나중에 해당 리전의 모든 작업이 완료된 후 버킷을 삭제할 수 있습니다.

지연 알림 임계값(분)

지연 알림이 전송되기 전까지의 임계값(분)을 설정합니다. `RUNNING`, `STARTING` 또는 `STOPPING` 작업 실행이 예상 시간(분)을 초과할 때 알림을 전송하도록 이 임계값을 설정할 수 있습니다.

보안 구성

목록에서 보안 구성을 선택합니다. 보안 구성은 암호화 없음, AWS KMS 관리형 키(SSE-KMS)를 사용한 서버 측 암호화, Amazon S3 관리형 암호화 키(SSE-S3) 등 Amazon S3 대상의 데이터 암호화 방법을 지정합니다.

서버 측 암호화

이 옵션을 선택할 경우, ETL 작업을 Amazon S3에 작성하면 SSE-S3 암호화를 사용하여 데이터는 암호화합니다. Amazon S3 임시 디렉터리에 쓴 모든 데이터와 Amazon S3 데이터 대상 모두 암호화됩니다. 이 옵션은 작업 파라미터로 전달됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 관리형 암호화 키(SSE-S3)로 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

> **Important**
>
> 보안 구성이 지정된 경우 이 옵션은 무시됩니다.

Glue 데이터 카탈로그를 Hive 메타스토어로 사용

이를 선택하여 AWS Glue 데이터 카탈로그를 Hive 메타스토어로 사용합니다. 작업에 사용되는 IAM 역할에는 `glue:CreateDatabase` 권한이 있어야 합니다. "기본값"이라는 데이터베이스가 없는 경우에 데이터 카탈로그에서 생성됩니다.

연결

Virtual Private Cloud(VPC)에 있는 Amazon S3 데이터 소스에 액세스하려면 VPC 구성을 선택합니다. AWS Glue에서 네트워크 연결을 생성하고 관리할 수 있습니다. 자세한 내용은 데이터에 연결 섹션을 참조하세요.

Libraries

Python 라이브러리 경로, 종속 JAR 경로 및 참조된 파일 경로

스크립트에 필요한 경우 이러한 옵션을 지정합니다. 작업을 정의할 때 이러한 옵션에 대해 쉼표로 구분된 Amazon S3 경로를 정의할 수 있습니다. 작업을 실행할 때 이 경로를 무시할 수 있습니다. 자세한 내용은 사용자 지정 스크립트 제공 섹션을 참조하세요.

작업 파라미터

이름이 지정된 파라미터로 스크립트에 전달되는 키-값 쌍의 집합입니다. 이러한 값은 스크립트가 실행될 때 사용되는 기본값이지만 트리거에서 또는 작업을 실행할 때 재정의할 수 있습니다. 키 이름 앞에 `--`를 붙여야 합니다(예: `--myKey`). AWS Command Line Interface를 사용할 때 맵으로 작업 파라미터를 전달합니다 .

예를 들어 AWS Glue에서 Python 파라미터 전달 및 액세스의 Python 파라미터를 참조하십시오.

Tags

태그 키와 선택 사항인 태그 값으로 작업에 태그를 지정합니다. 생성된 태그 키는 읽기 전용입니다. 일부 리소스에서 태그를 이용하면 리소스를 정리하고 식별하는 데 도움이 됩니다. 자세한 내용은 AWS Glue의 AWS 태그 섹션을 참조하세요.

Lake Formation 관리형 테이블에 액세스하는 작업에 대한 제한 사항

AWS Lake Formation에서 관리되는 테이블에서 읽거나 쓰는 작업을 생성하는 경우 다음 참고 사항과 제한 사항에 유의하세요.

- 셀 수준 필터가 있는 테이블에 액세스하는 작업에서는 다음 기능이 지원되지 않습니다.
  - 작업 북마크 및 제한된 실행
  - 푸시다운 조건자

- 서버 측 카탈로그 파티션 조건자
- enableUpdateCatalog

AWS Glue 콘솔에서 Spark 스크립트 편집

스크립트는 소스에서 데이터를 추출하고 데이터를 변환한 다음 대상으로 로드하는 스크립트를 실행하는 코드를 포함합니다. AWS Glue는 작업 시작 시 스크립트를 실행합니다.

AWS Glue ETL 스크립트는 Python 또는 Scala에 코딩될 수 있습니다. Python 스크립트는 추출, 변환 및 로드(ETL) 작업 스크립트의 PySpark Python의 확장 언어를 사용합니다. 스크립트는 ETL 변환을 다루는 확장된 구조를 포함합니다. 자동적으로 작업의 소스 코드 논리를 생성하면 스크립트가 생성됩니다. 스크립트를 편집하거나 자체 스크립트를 제공하여 ETL 작업을 실행할 수 있습니다.

AWS Glue에서 스크립트를 정의하고 편집하는 방법에 대한 자세한 내용은 AWS Glue 프로그래밍 안내서 섹션을 참조하세요.

추가 라이브러리 또는 파일

스크립트가 추가 라이브러리 혹은 파일은 요구하면 다음과 같이 지정합니다.

Python 라이브러리 경로

스크립트에 필요한 Python 라이브러리로 이동하는, 쉼표로 구분된 Amazon Simple Storage Service(Amazon S3) 경로입니다.

> **Note**
>
> 순수 Python 라이브러리만 사용할 수 있습니다. pandas Python 데이터 분석 라이브러리 등 C 확장을 활용하는 라이브러리는 아직 지원되지 않습니다.

종속된 jars 경로

스크립트에 필요한 JAR 파일로 이동하는, 쉼표로 구분된 Amazon S3 경로입니다.

> **Note**
>
> 현재 순수 Java 또는 Scala(2.11) 라이브러리만 사용할 수 있습니다.

참조된 파일 경로

스크립트에 필요한 추가 파일(예: 구성 파일)로 이동하는, 쉼표로 구분된 Amazon S3 경로입니다.

작업(레거시)

스크립트는 작업을 추출, 변환 및 로드(ETL)하는 코드를 포함합니다. 사용자가 스크립트를 제공하거나 AWS Glue가 사용자의 지시에 따라 스크립트를 생성할 수 있습니다. 스크립트 생성에 대한 자세한 내용은 사용자 지정 스크립트 제공 섹션을 참조하십시오.

AWS Glue 콘솔의 스크립트를 편집할 수 있습니다. 스크립트를 편집하려면 원본, 대상 및 변환을 추가할 수 있습니다.

스트립트 편집

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다. 그런 다음 [Jobs(작업)] 탭을 선택합니다.
2. 목록에서 작업을 선택한 다음 [Action(작업)]과 [Edit script(스크립트 편집)]을 선택하여 스크립트 에디터를 엽니다.

   작업 상세 정보 페이지에서 스크립트 에디터로 액세스할 수 있습니다. [스크립트(Script)] 탭을 선택한 다음 [스크립트 편집(Edit Script)]을 선택합니다.

스크립트 에디터

AWS Glue 스크립트 편집기를 사용하여 스트립트에서 원본, 대상 및 변환을 삽입, 수정 및 삭제할 수 있습니다. 스크립트 편집기는 데이터 흐름을 시각화하여 스크립트와 다이어그램을 모두 보여줍니다.

스크립트용 다이어그램을 생성하려면 다이어그램 생성을 선택합니다. AWS Glue는 ##로 시작하는 스크립트 문장을 사용하여 다이어그램을 만듭니다. 다이어그램에서 스크립트를 정확하게 나타내기 위해서 파라미터를 주석에 있도록 하고 Apache Spark 코드와 동기화되도록 합니다.

스크립트 편집기는 커서가 스크립트 어디에 있든지 코드 템플릿을 추가합니다. 편집기 상단에서 다음 옵션을 선택합니다.

- [Source(원본)]을 선택하여 원본 테이블을 스크립트에 추가합니다.
- [Target(대상)]을 선택하여 대상 테이블을 스크립트에 추가합니다.
- [Target location(대상 위치)]을 선택하여 대상 위치를 스크립트에 추가합니다.

- [Transform(변환)]을 선택하여 변환을 스크립트에 추가합니다. 스크립트에 호출된 함수에 대한 자세한 내용은 PySpark의 AWS Glue ETL 스크립트 프로그래밍 단원을 참조하십시오.
- [Spigot(스피곳)]을 선택하여 스피곳 변환을 스크립트에 추가합니다.

삽입된 코드의 경우, 주석과 Apache Spark 코드 모두에서 `parameters`를 수정합니다. 예를 들어, [Spigot(스피곳)] 변환을 추가하면 `path`가 `@args` 주석과 `output` 코드 라인으로 바뀌었는지 확인합니다.

[Logs(로그)] 탭은 작업이 실행되면서 작업과 관련된 로그를 보여줍니다. 최신 1,000 라인이 표시됩니다.

[스키마(Schema)] 탭에는 선택한 원본 및 대상의 스키마가 표시됩니다(Data Catalog에서 사용 가능한 경우).

처리된 데이터를 작업 북마크로 추적

AWS Glue는 작업 실행의 상태 정보를 유지하여 이전에 ETL 작업을 실행할 때 이미 처리된 데이터를 추적합니다. 이와 같은 지속 상태 정보를 작업 북마크라고 합니다. AWS Glue는 작업 북마크로 상태 정보를 유지하고 이전 데이터의 재처리를 방지합니다. 작업 북마크를 사용하면 예약된 간격으로 재실행 중인 새 데이터를 처리할 수 있습니다. 작업 북마크는 원본, 변환 및 대상과 같은 다양한 작업 요소의 상태로 구성됩니다. 예를 들어, ETL 작업에서 Amazon S3 파일의 새 파티션을 읽어야 할 수 있습니다. AWS Glue는 그 작업에서 처리한 파티션을 추적하여 중복 실행을 방지하고 작업 대상인 데이터 스토어에 데이터를 복제합니다.

작업 북마크는 JDBC 데이터 원본, Relationalize 변환 및 일부 Amazon Simple Storage Service(Amazon S3) 소스에 대해 구현됩니다. 다음 테이블에는 AWS Glue가 작업 북마크에 대해 지원하는 Amazon S3 소스 포맷이 나열됩니다.

AWS Glue 버전	Amazon S3 소스 포맷
버전 0.9	JSON, CSV, Apache Avro, XML
버전 1.0 이상	JSON, CSV, Apache Avro, XML, Parquet, ORC

AWS Glue 버전에 대한 자세한 내용은 Spark 작업에 대한 작업 속성 정의 섹션을 참조하세요.

AWS Glue 스크립트를 통해 액세스할 경우 작업 북마크 기능에는 추가 기능이 있습니다. 생성된 스크립트를 찾아보면 이 기능과 관련된 변환 컨텍스트가 표시될 수 있습니다. 자세한 내용은 the section called “작업 북마크 사용” 단원을 참조하십시오.

주제

- AWS Glue에서 작업 북마크 사용
- 작업 북마크 기능의 운영 세부 정보

AWS Glue에서 작업 북마크 사용

작업이 시작되면 작업 북마크 옵션을 파라미터로 전달합니다. 다음 표에 AWS Glue 콘솔에서 작업 북마크를 설정하기 위한 옵션이 설명되어 있습니다.

작업 북마크	설명
Enable	이전에 처리된 데이터를 추적하기 위해 작업 실행 후 상태를 업데이트하도록 합니다. 작업 북마크가 지원되는 소스의 작업이라면 이미 처리된 데이터를 추적하고, 작업 실행 시 마지막 체크포인트 이후에 받은 새 데이터를 처리합니다.
비활성화	작업 북마크를 사용하지 않고, 작업에서 항상 전체 데이터 세트를 처리합니다. 이전에 실행된 작업의 출력값 관리는 여러분이 직접 수행해야 합니다. 이 값이 기본값입니다.
일시 중지	마지막 북마크의 상태를 업데이트하지 않고 마지막으로 성공한 실행 이후의 증분 데이터 또는 다음 하위 옵션으로 식별된 범위의 데이터를 처리합니다. 이전에 실행된 작업의 출력값 관리는 여러분이 직접 수행해야 합니다. 두 개의 하위 옵션은 다음과 같습니다. • job-bookmark-from<시작 값>은 마지막으로 성공한 실행 전까지 처리된 모든 입력을 나타내는 실행 ID이며, 지정된 실행 ID를 포함합니다. 해당하는 입력은 무시됩니다. • job-bookmark-to<종료 값>는 마지막으로 성공한 실행 전까지 처리된 모든 입력을 나타내는 실행 ID이며, 지정된 실행 ID를 포함합니다. <시작 값>에 의해 식별된 입력을 제외한 해당하는 입력이 작업에서 처리됩니다. 이 입력 이후의 입력도 처리에서 제외됩니다.

작업 북마크	설명
	이 옵션 세트를 지정하면 작업 북마크 상태가 업데이트되지 않습니다. 하위 옵션은 선택 사항이지만, 두 하위 옵션이 모두 사용되는 경우에는 제공해야 합니다.

명령줄에서 작업에 전달된 파라미터, 특히 작업 북마크에 대한 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용 단원을 참조하십시오.

Amazon S3 입력 소스의 경우 AWS Glue 작업 북마크는 객체의 마지막 수정 시간을 검사해 어떤 객체를 재처리해야 하는지 확인합니다. 마지막 작업 실행 이후로 입력 소스 데이터가 수정된 경우, 작업을 다시 실행하면 그 파일이 재처리됩니다.

JDBC 소스의 경우 다음 규칙이 적용됩니다.

- 각 테이블에 대해 AWS Glue는 하나 이상의 열을 북마크 키로 사용하여 새 데이터와 처리된 데이터를 결정합니다. 북마크 키가 결합되어 단일 복합 키를 형성합니다.
- AWS Glue에서는 기본적으로 프라이머리 키를 북마크 키로 사용합니다. 단, 해당 키는 간격 없이 순차적으로 증가하거나 감소합니다.
- AWS Glue 스크립트에서 북마크 키로 사용할 열을 지정할 수 있습니다. AWS Glue 스크립트에서 작업 북마크를 사용하는 방법에 대한 자세한 내용은 the section called “작업 북마크 사용” 섹션을 참조하세요.
- AWS Glue에서는 대소문자를 구분하는 열을 작업 북마크 키로 사용하는 것을 지원하지 않습니다.

AWS Glue Spark ETL 작업에 대한 작업 북마크를 이전 작업 실행으로 되돌릴 수 있습니다. 작업 북마크를 이전의 작업 실행으로 되돌릴 수 있어 후속 작업 실행 시 북마크로 지정된 작업 실행의 데이터만 다시 처리하므로 데이터 다시 채우기 시나리오를 훨씬 효율적으로 지원할 수 있습니다.

모든 데이터를 동일한 작업에서 재처리하려는 경우에는 작업 북마크를 재설정합니다. 작업 북마크 상태를 재설정하려면 AWS Glue 콘솔, ResetJobBookmark 작업(Python: reset_job_bookmark) API 작업 또는 AWS CLI를 사용합니다. 예를 들어, AWS CLI에서는 다음 명령을 입력합니다.

```
aws glue reset-job-bookmark --job-name my-job-name
```

북마크를 되감거나 재설정할 때 여러 대상이 있을 수 있고 대상이 작업 북마크로 추적되지 않기 때문에 AWS Glue는 대상 파일을 정리하지 않습니다. 소스 파일만 북마크로 추적됩니다. 출력에서 중복 데이터를 피하기 위해 소스 파일을 되감고 다시 처리할 때 다른 출력 대상을 생성할 수 있습니다.

AWS Glue는 작업 북마크를 작업별로 추적합니다. 작업을 삭제하면 작업 북마크가 삭제됩니다.

AWS Glue 작업 북마크를 활성화했는데도 ETL 작업이 앞선 실행에서 이미 처리된 데이터를 재처리하는 경우가 있을 수 있습니다. 이러한 오류의 일반적인 원인 해결에 대한 자세한 내용은 Spark 오류 문제 해결 단원을 참조하십시오.

작업 북마크 기능의 운영 세부 정보

이 단원에서는 작업 북마크 사용의 실무적 측면에 대해 좀 더 자세히 설명합니다.

작업 북마크는 작업의 상태를 저장합니다. 각각의 상태 인스턴스를 작업 이름과 버전 번호로 입력합니다. 스크립트가 `job.init`를 호출하면 상태를 검색해 항상 최신 버전을 가져옵니다. 상태 안에는 여러 가지 상태 요소가 있는데, 이러한 요소는 해당 스크립트에서 소스, 변환 및 싱크 인스턴스별로 고유합니다. 이러한 상태 요소는 스크립트의 해당 요소(소스, 변환 또는 싱크)에 연결된 변환 컨텍스트로 식별할 수 있습니다. 상태 요소는 사용자 스크립트에서 `job.commit`를 호출할 때 기본적으로 저장됩니다. 이 스크립트는 인수에서 작업 이름과 작업 북마크에 대한 제어 옵션을 가져옵니다.

작업 북마크의 상태 요소는 소스, 변환 또는 싱크 관련 데이터입니다. 예를 들어, 업스트림 작업 또는 프로세스에서 끊임 없이 쓰는 Amazon S3 위치에서 증분 데이터를 읽으려 한다고 가정해 보겠습니다. 이 경우, 스크립트는 지금까지 처리된 항목을 확인해야 합니다. Amazon S3 소스에 대해 작업 북마크를 구현하면 정보를 저장하기 때문에 작업을 다시 실행할 때 저장된 정보를 사용해 새 객체만 필터링하여 작업의 다음 실행을 위해 상태를 재컴퓨팅할 수 있습니다. 타임스탬프는 새 파일을 필터링하는 데 사용됩니다.

작업 북마크에는 상태 요소 외에도 실행 횟수, 시도 횟수 및 버전 번호가 포함되어 있습니다. 실행 횟수는 작업의 실행을 추적하고, 시도 횟수는 작업 실행을 시도한 횟수를 기록합니다. 작업 실행 횟수는 실행에 성공할 때마다 단순하게 늘어나는 숫자입니다. 시도 횟수는 각각의 실행 시도를 추적하며, 시도 실패 후 실행하는 경우에만 숫자가 늘어납니다. 버전 번호는 단순히 증가하고, 작업 북마크에 대한 업데이트를 추적합니다.

AWS Glue 서비스 데이터베이스에서 모든 변환에 대한 북마크 상태는 키-값 쌍으로 함께 저장됩니다.

```
{
  "job_name" : ...,
  "run_id": ...,
  "run_number": ..,
```

```
  "attempt_number": ...
  "states": {
    "transformation_ctx1" : {
      bookmark_state1
    },
    "transformation_ctx2" : {
      bookmark_state2
    }
  }
}
```

모범 사례

다음은 작업 북마크 사용에 관한 모범 사례입니다.

- 북마크가 사용 설정된 상태에서 데이터 원본 속성을 변경하지 마세요. 예를 들어, Amazon S3 입력 경로 A를 가리키는 datasource0가 있으며 작업은 북마크가 사용 설정된 상태에서 여러 라운드 동안 실행 중인 소스에서 읽어오고 있습니다. `transformation_ctx`를 변경하지 않고 datasource0의 입력 경로를 Amazon S3 경로 B로 변경하는 경우, AWS Glue 작업은 저장된 이전 북마크 상태를 사용합니다. 그러면 AWS Glue가 해당 파일이 이전 실행에서 처리되었다고 가정하여 입력 경로 B에서 파일이 누락되거나 파일을 건너뛰게 됩니다.
- 더 효과적인 파티션 관리를 위해 북마크가 있는 카탈로그 테이블을 사용하세요. 책갈피는 Data Catalog 또는 옵션의 데이터 원본에 모두 작동합니다. 그러나 옵션 접근 방식을 사용하면 새 파티션을 제거/추가하는 것은 어렵습니다. 크롤러로 카탈로그 테이블을 사용하면 새로 추가된 파티션을 추적하는 더 향상된 자동화를 제공할 수 있으며, 푸시다운 조건자로 특정 파티션을 선택할 수 있는 유연성이 확보됩니다.
- 대용량 데이터 집합에 대해 AWS Glue Amazon S3 파일 리스터를 사용하세요. 북마크는 각 입력 파티션 아래의 모든 파일을 나열하고 필터링을 수행하므로 단일 파티션 아래에 파일이 너무 많으면 북마크가 드라이버 OOM으로 실행될 수 있습니다. 메모리에 있는 모든 파일을 한 번에 나열하지 않도록 AWS Glue Amazon S3 파일 리스터를 사용하세요.

Spark 셔플 데이터 저장

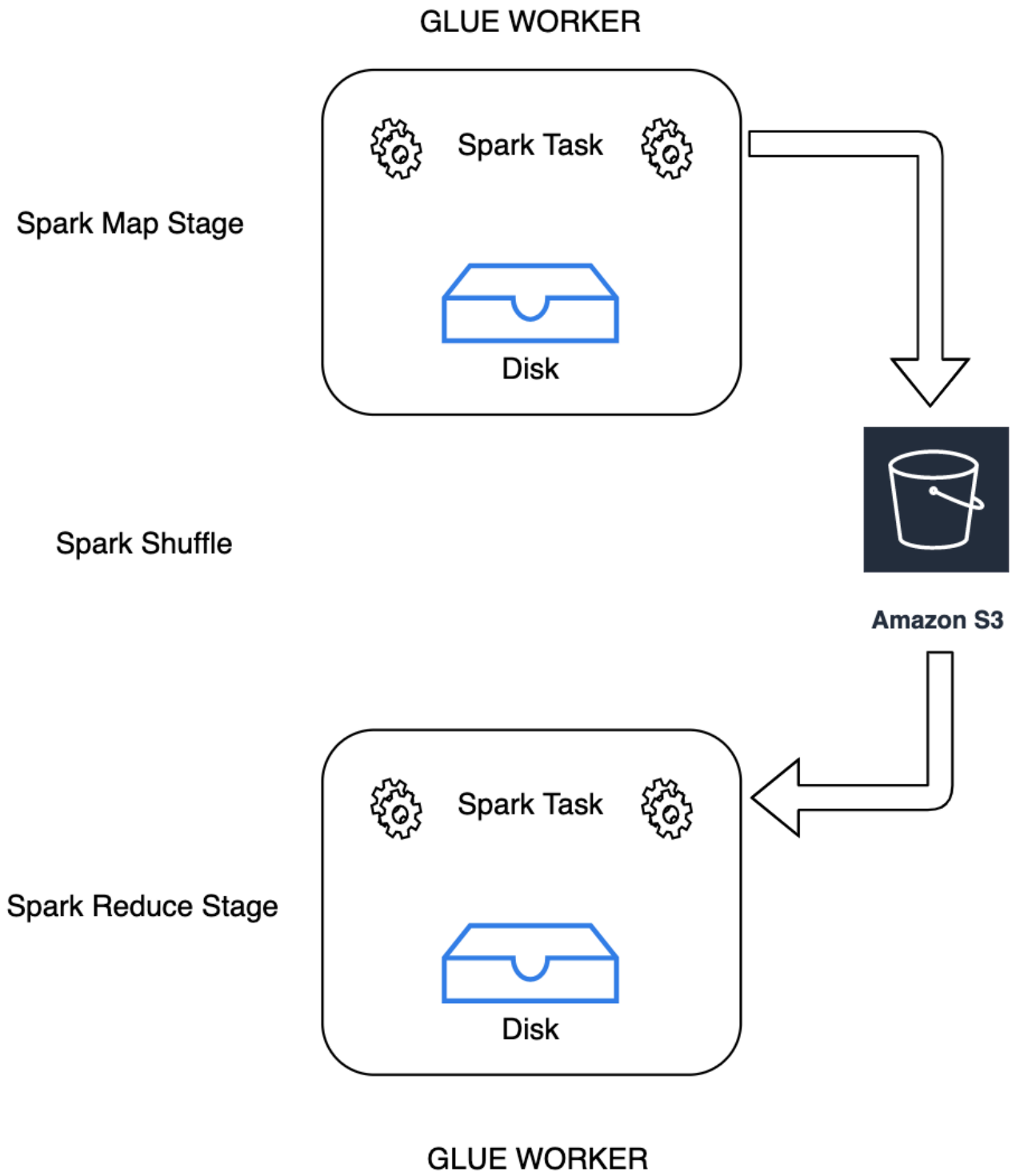
셔플링은 데이터가 파티션 간에 재배열될 때마다 Spark 작업에서 중요한 단계입니다. 이는 `join`, `groupByKey`, `reduceByKey`, `repartition` 등의 광범위한 변환이 처리를 완료하기 위해 다른 파티션의 정보를 필요로 하기 때문에 필요합니다. Spark는 각 파티션에서 필요한 데이터를 수집하여 새 파티션으로 결합합니다. 셔플 중에 데이터가 디스크에 기록되고 네트워크를 통해 전송됩니다. 결과적으로 셔플 작업은 로컬 디스크 용량에 바인딩됩니다. Spark는 실행기에 디스크 공간이 충분하지 않고 복

구가 없을 때 `No space left on device` 또는 `MetadataFetchFailedException` 오류를 발생시킵니다.

> **Note**
>
> Amazon S3의 AWS Glue Spark 셔플 플러그인은 AWS Glue ETL 작업에서만 지원됩니다.

Solution

AWS Glue에서는 이제 Amazon S3를 사용하여 Spark 셔플 데이터를 저장할 수 있습니다. Amazon S3는 업계 최고의 확장성, 데이터 가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스입니다. 이 솔루션은 Spark 작업에 대한 컴퓨팅 및 스토리지를 분해하고 완전한 탄력성과 저비용 셔플 스토리지를 제공하여 가장 셔플 집약적인 워크로드를 안정적으로 실행할 수 있도록 합니다.



Amazon S3를 사용하기 위한 새로운 Apache Spark용 클라우드 셔플 스토리지 플러그인을 소개합니다. 대규모 셔플 작업을 위한 로컬 디스크 용량에 의해 제한되는 것으로 알려진 경우 Amazon S3 셔플링을 켜서 AWS Glue 작업을 실패 없이 안정적으로 실행할 수 있습니다. 경우에 따라 Amazon S3에 기

록된 작은 파티션이나 셔플 파일이 많은 경우 Amazon S3로의 셔플링은 로컬 디스크(또는 EBS)보다 약간 느립니다.

클라우드 셔플 스토리지 플러그인 사용을 위한 필수 조건

클라우드 셔플 스토리지 플러그인을 AWS Glue ETL 작업에서 사용하려면 다음이 필요합니다.

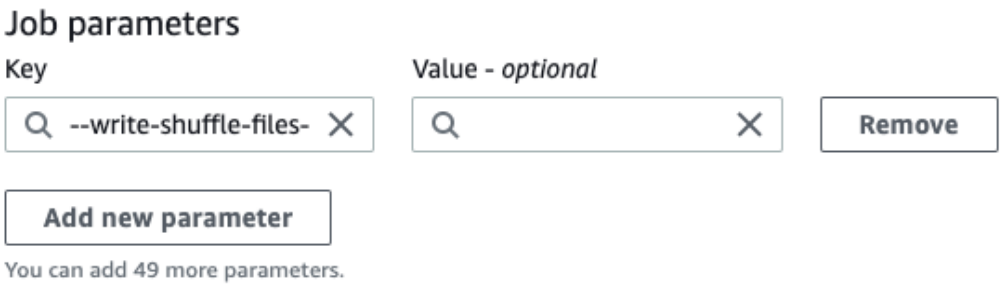
- 중간 셔플 및 유출된 데이터를 저장할, 작업 실행과 동일한 리전에 위치한 Amazon S3 버킷. 셔플 스토리지의 Amazon S3 접두사는 다음 예제와 같이 `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket`*`/`*`prefix`*`/`으로 지정될 수 있습니다.

  ```
  --conf spark.shuffle.glue.s3ShuffleBucket=s3://glue-shuffle-123456789-us-east-1/glue-shuffle-data/
  ```

- 셔플 관리자는 작업이 완료된 후 파일을 정리하지 않으므로 접두사(예: `glue-shuffle-data`)에서 Amazon S3 스토리지 수명 주기 정책을 설정합니다. 중간 셔플 및 유출된 데이터는 작업 완료 후 삭제해야 합니다. 사용자는 접두사에 간단한 수명 주기 정책을 설정할 수 있습니다. Amazon S3 수명 주기 정책 설정 지침은 Amazon Simple Storage Service Console 사용 설명서의 버킷에서 수명 주기 구성 설정을 참조하세요.

AWS 콘솔에서 AWS Glue Spark 셔플 관리자 사용

작업을 구성할 때 AWS Glue 콘솔 또는 AWS Glue Studio를 사용하여 AWS Glue Spark 셔플 관리자를 설정하려면 --write-shuffle-files-to-s3 작업 파라미터를 선택하여 작업에 대해 Amazon S3 셔플을 설정합니다.



AWS Glue Spark 셔플 플러그인 사용

다음 작업 파라미터는 AWS Glue 셔플 매니저를 설정하고 조정합니다. 이러한 파라미터는 플래그이므로 제공된 값은 고려되지 않습니다.

- `--write-shuffle-files-to-s3` - AWS Glue Spark 셔플 관리자가 Amazon S3 버킷을 사용하여 셔플 데이터를 쓰고 읽을 수 있도록 하는 기본 플래그입니다. 플래그가 지정되지 않은 경우 셔플 관리자가 사용되지 않습니다.
- `--write-shuffle-spills-to-s3` - (AWS Glue 버전 2.0에서만 지원됨). 유출 파일을 Amazon S3 버킷으로 오프로드하도록 허용하는 선택적 플래그로, Spark 작업에 복원력을 추가로 제공합니다. 이는 많은 데이터를 디스크로 유출하는 대규모 워크로드에만 필요합니다. 플래그를 지정하지 않으면 중간 유출 파일이 작성되지 않습니다.
- `--conf spark.shuffle.glue.s3ShuffleBucket=s3://<shuffle-bucket>` - 셔플 파일을 작성하는 Amazon S3 버킷을 지정하는 또 다른 선택적 플래그입니다. 기본값은 `--TempDir` /shuffle-data입니다. AWS Glue 3.0 이상에서는 `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket-1`*`/`*`prefix`*`,s3://`*`shuffle-bucket-2`*`/`*`prefix`*`/`에서와 같이 쉼표 구분자를 사용해 버킷을 지정함으로써 여러 버킷에 셔플 파일을 쓸 수 있도록 지원합니다. 버킷을 여러 개 사용하면 성능이 향상됩니다.

셔플 데이터의 유휴 암호화를 활성화하려면 보안 구성 설정을 제공해야 합니다. 보안 구성에 대한 자세한 내용은 the section called “암호화 설정” 섹션을 참조하세요. AWS Glue는 Spark에서 제공하는 다른 모든 셔플 관련 구성을 지원합니다.

클라우드 셔플 스토리지 플러그인을 위한 소프트웨어 바이너리

또한 Apache 2.0 라이선스에 따라 Apache Spark용 클라우드 셔플 스토리지 플러그인의 소프트웨어 바이너리를 다운로드하여 Spark 환경에서 실행할 수 있습니다. 새 플러그인은 Amazon S3에 대한 기본 지원과 함께 제공되며 Google Cloud Storage 및 Microsoft Azure Blob Storage와 같은 다른 형태의 클라우드 스토리지를 사용하도록 쉽게 구성할 수도 있습니다. 자세한 내용은 Apache Spark용 클라우드 셔플 스토리지 플러그인을 참조하세요.

참고 및 제한 사항

다음은 AWS Glue 셔플 관리자에 대한 참고 또는 제한 사항입니다.

- AWS Glue 셔플 관리자는 작업이 완료된 후 Amazon S3 버킷에 저장된 (임시) 셔플 데이터 파일을 자동으로 삭제하지 않습니다. 데이터를 보호하려면 클라우드 셔플 스토리지 플러그인을 활성화하기 전에 클라우드 셔플 스토리지 플러그인 사용을 위한 필수 조건의 지침을 따릅니다.
- 데이터가 왜곡된 경우 이 기능을 사용할 수 있습니다.

Apache Spark용 클라우드 셔플 스토리지 플러그인

클라우드 셔플 스토리지 플러그인은 `ShuffleDataIO API`와 호환되는 Apache Spark 플러그인으로, 셔플 데이터를 클라우드 스토리지 시스템(예: Amazon S3)에 저장할 수 있습니다. Spark 애플리케이션에서 일반적으로 `join`, `reduceByKey`, `groupByKey`, `repartition`과 같은 변환에 의해 트리거되는 대규모 셔플 작업을 위해 로컬 디스크 스토리지 용량을 보충하거나 교체하여 서버리스 데이터 분석 작업 및 파이프라인의 일반적인 장애 또는 가격 대비 성능 문제를 줄일 수 있습니다.

AWS Glue

AWS Glue 버전 3.0 및 4.0에는 플러그인이 미리 설치되어 있으며 추가 단계 없이 Amazon S3로 셔플링할 수 있습니다. 자세한 내용을 보려면 Amazon S3의 AWS Glue Spark 셔플 클러그인을 참조하여 Spark 애플리케이션에 대한 기능을 활성화하세요.

기타 Spark 환경

플러그인을 사용하려면 다른 Spark 환경에서 다음과 같은 Spark 구성을 설정해야 합니다.

- `--conf spark.shuffle.sort.io.plugin.class=com.amazonaws.spark.shuffle.io.cloud.Choppe`
  Shuffle IO에 이 플러그인을 사용하도록 Spark에 알려줍니다.
- `--conf spark.shuffle.storage.path=s3://`*`bucket-name`*`/`*`shuffle-file-dir`*: 셔플 파일이 저장되는 경로입니다.

> **Note**
>
> 플러그인은 Spark 코어 클래스 하나를 덮어씁니다. 따라서 플러그인 jar을 Spark jar보다 먼저 로드해야 합니다. 플러그인을 AWS Glue 외부에서 사용하는 경우 온-프레미스 YARN 환경에서 `userClassPathFirst`를 사용하여 이 작업을 수행할 수 있습니다.

Spark 애플리케이션과 플러그인 번들링

Spark 애플리케이션을 로컬에서 개발하는 동안 Maven `pom.xml`에 플러그인 종속성을 추가하여 플러그인을 Spark 애플리케이션 및 Spark 배포(버전 3.1 이상)와 함께 번들로 제공할 수 있습니다. 플러그인 및 Spark 버전에 대한 자세한 내용은 플러그인 버전 섹션을 참조하세요.

```
<repositories>
   ...
    <repository>
```

```
        <id>aws-glue-etl-artifacts</id>
        <url>https://aws-glue-etl-artifacts.s3.amazonaws.com/release/ </url>
    </repository>
</repositories>
...
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>chopper-plugin</artifactId>
    <version>3.1-amzn-LATEST</version>
</dependency>
```

또는 다음과 같이 AWS Glue Maven 아티팩트에서 바이너리를 직접 다운로드하여 Spark 애플리케이션에 포함시킬 수도 있습니다.

```
#!/bin/bash
sudo wget -v https://aws-glue-etl-artifacts.s3.amazonaws.com/release/com/amazonaws/
chopper-plugin/3.1-amzn-LATEST/chopper-plugin-3.1-amzn-LATEST.jar -P /usr/lib/spark/
jars/
```

예제 spark-submit

```
spark-submit --deploy-mode cluster \
--conf spark.shuffle.storage.s3.path=s3://<ShuffleBucket>/<shuffle-dir> \
--conf spark.driver.extraClassPath=<Path to plugin jar> \
--conf spark.executor.extraClassPath=<Path to plugin jar> \
--class <your test class name> s3://<ShuffleBucket>/<Your application jar> \
```

구성 옵션

Amazon S3 셔플 동작을 제어하는 구성 옵션 값이 있습니다.

- `spark.shuffle.storage.s3.enableServerSideEncryption`: 셔플 및 유출 파일에 대한 S3 SSE를 활성화/비활성화합니다. 기본값은 `true`입니다.
- `spark.shuffle.storage.s3.serverSideEncryption.algorithm`: 사용할 SSE 알고리즘입니다. 기본값은 `AES256`입니다.
- `spark.shuffle.storage.s3.serverSideEncryption.kms.key`: SSE aws:kms가 활성화된 경우의 KMS 키 ARN입니다.

이러한 구성과 함께 `spark.hadoop.fs.s3.enableServerSideEncryption` 및 기타 환경별 구성과 같은 구성을 설정하여 사용 사례에 적합한 암호화가 적용되도록 해야 할 수도 있습니다.

플러그인 버전

이 플러그인은 각 AWS Glue 버전과 연결된 Spark 버전에서 지원됩니다. 다음 표에는 플러그인 소프트웨어 바이너리의 AWS Glue 버전, Spark 버전 및 Amazon S3 위치와 연결된 플러그인 버전이 나와 있습니다.

AWS Glue 버전	Spark 버전	플러그인 버전	Amazon S3 위치
3.0	3.1	3.1-amzn-LATEST	s3://aws-glue-etl-artifacts/release/com/amazonaws/chopper-plugin/3.1-amzn-0/chopper-plugin-3.1-amzn-LATEST.jar
4.0	3.3	3.3-amzn-LATEST	s3://aws-glue-etl-artifacts/release/com/amazonaws/chopper-plugin/3.3-amzn-0/chopper-plugin-3.3-amzn-LATEST.jar

라이선스

이 플러그인의 소프트웨어 바이너리는 Apache-2.0 라이선스에 따라 사용이 허가됩니다.

AWS Glue Spark 작업 모니터링

주제

- AWS Glue Studio에서 사용할 수 있는 Spark 지표
- Apache Spark 웹 UI를 사용하여 작업 모니터링
- AWS Glue 작업 실행 인사이트를 사용한 모니터링
- Amazon CloudWatch를 사용한 모니터링
- 작업 모니터링 및 디버깅

AWS Glue Studio에서 사용할 수 있는 Spark 지표

[지표(Metrics)] 탭에는 작업이 실행되고 프로파일링이 활성화될 때 수집되는 지표가 표시됩니다. 다음 그래프가 Spark 작업에 표시됩니다.

- ETL 데이터 이동
- 메모리 프로필: 드라이버 및 실행기

다음 그래프를 표시하려면 View additional metrics(추가 측정치 보기)를 선택합니다.

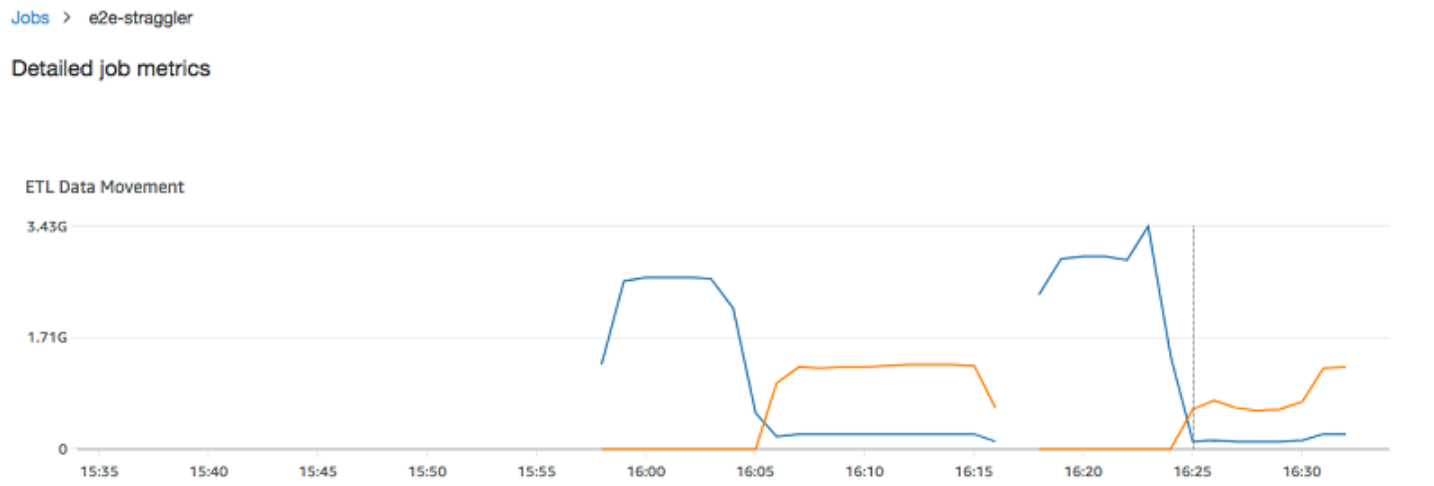
- ETL 데이터 이동
- 메모리 프로필: 드라이버 및 실행기
- 실행기 간의 데이터 셔플
- CPU 부하: 드라이버 및 실행기
- 작업 실행: 활성 실행기, 완료된 단계 및 최대 필요 실행기

지표를 수집하도록 작업이 구성된 경우 이러한 그래프에 대한 데이터가 CloudWatch 지표로 푸시됩니다. 지표를 설정하고 그래프를 해석하는 방법에 대한 자세한 내용은 작업 모니터링 및 디버깅 섹션을 참조하세요.

Example ETL 데이터 이동 그래프

ETL 데이터 이동 그래프는 다음 측정치를 표시합니다.

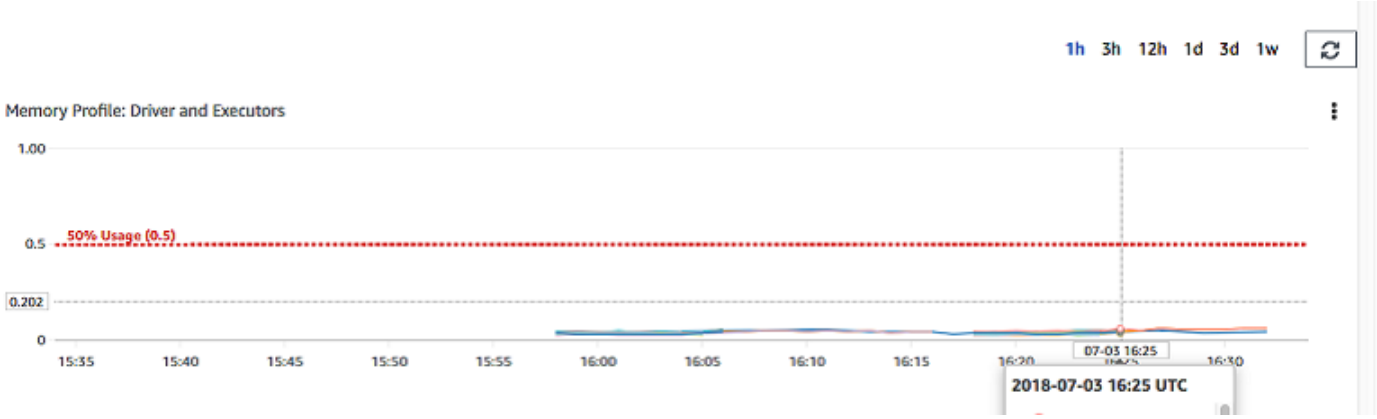
- 모든 실행기가 Amazon S3에서 읽은 바이트 수 - `glue.ALL.s3.filesystem.read_bytes`
- 모든 실행기가 Amazon S3에 쓴 바이트 수 - `glue.ALL.s3.filesystem.write_bytes`



Example 메모리 프로필 그래프

메모리 프로필 그래프는 다음 측정치를 표시합니다.

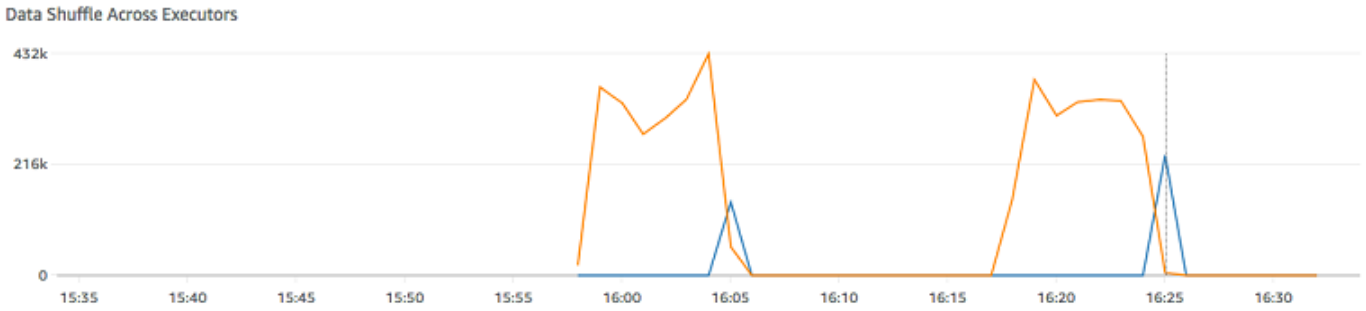
- 드라이버에 의해 이 드라이버용 JVM 힙에 사용되는 메모리 부분(규모: 0~1), executorId에 의해 식별되는 실행기, 또는 모든 실행기 -
  - `glue.driver.jvm.heap.usage`
  - `glue.executorId.jvm.heap.usage`
  - `glue.ALL.jvm.heap.usage`



Example 실행기 간의 데이터 셔플 그래프

실행기 간의 데이터 셔플 그래프는 다음 측정치를 표시합니다.

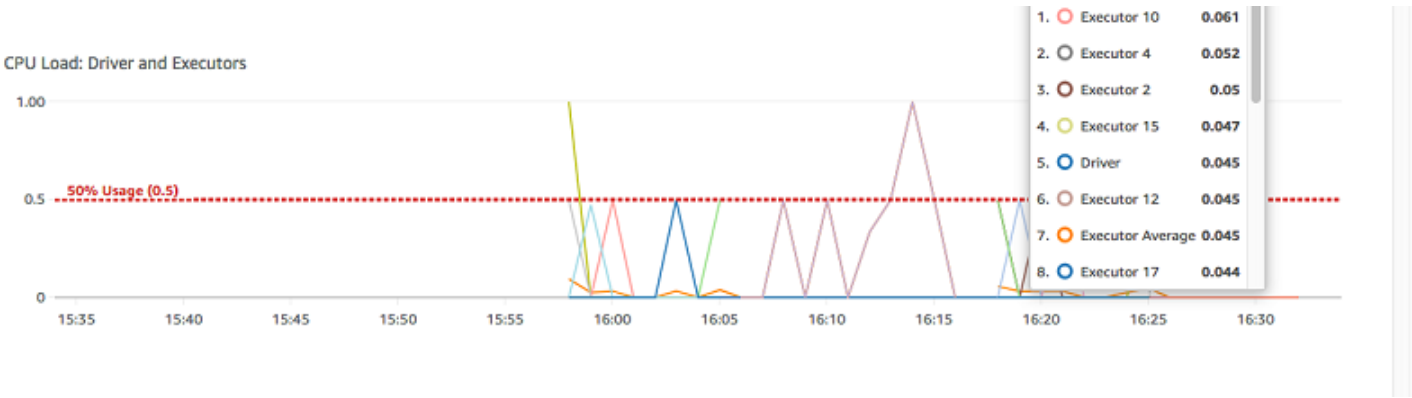
- 서로 간에 데이터를 셔플링하기 위해 모든 실행기가 읽은 바이트 수 - glue.driver.aggregate.shuffleLocalBytesRead
- 서로 간에 데이터를 셔플링하기 위해 모든 실행기가 쓴 바이트 수 - glue.driver.aggregate.shuffleBytesWritten



Example CPU 부하 그래프

CPU 부하 그래프는 다음 측정치를 표시합니다.

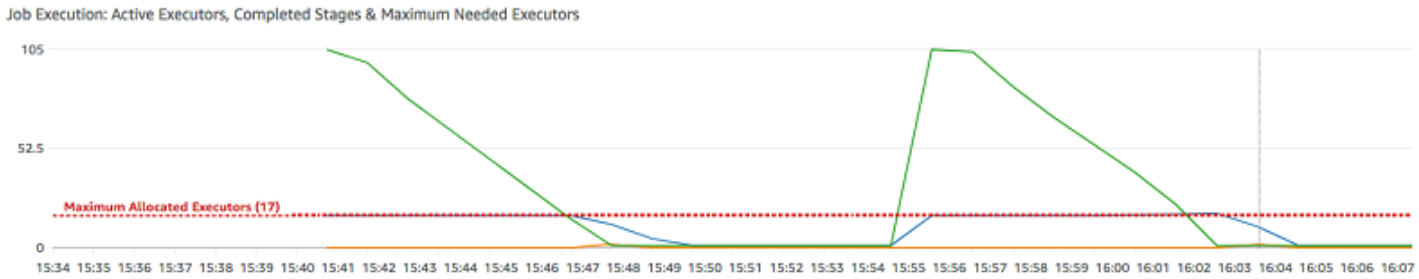
- 드라이버, executorId로 식별되는 실행기 또는 모든 실행기가 사용한 CPU 시스템 로드 부분(규모: 0~1).
  - glue.driver.system.cpuSystemLoad
  - glue.*executorId*.system.cpuSystemLoad
  - glue.ALL.system.cpuSystemLoad



Example 작업 실행 그래프

작업 실행 그래프는 다음 측정치를 표시합니다.

- 능동적으로 실행 중인 실행기 수 - glue.driver.ExecutorAllocationManager.executors.numberAllExecutors
- 완료된 단계 수 - glue.aggregate.numCompletedStages
- 최대 필요 실행기 수 - glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors



Apache Spark 웹 UI를 사용하여 작업 모니터링

Apache Spark 웹 UI를 사용하여 AWS Glue 작업 시스템에서 실행 중인 AWS Glue ETL 작업과 AWS Glue 개발 엔드포인트에서 실행 중인 Spark 애플리케이션을 모니터링하고 디버그할 수 있습니다. Spark UI를 사용하면 각 작업에 대해 다음을 확인할 수 있습니다.

- 각 Spark 단계의 이벤트 타임라인
- 작업의 방향성 비순환 그래프(DAG)
- SparkSQL 쿼리에 대한 물리적, 논리적 계획
- 각 작업에 대한 기본 Spark 환경 변수

Spark 웹 UI 사용에 대한 자세한 내용은 Spark 설명서의 웹 UI를 참조하십시오. Spark UI 결과를 해석하여 작업 성능을 개선하는 방법에 대한 지침은 AWS 권장 가이드의 Apache Spark 용 AWS Glue 작업 성능 조정 모범 사례를 참조하세요.

AWS Glue 콘솔에서 Spark UI를 확인할 수 있습니다. 이 기능은 AWS Glue 작업이 AWS Glue 3.0 이상 버전에서 실행되고 새로운 작업의 기본값인 레거시 형식이 아닌 표준 형식으로 생성된 로그가 있는 경우 사용할 수 있습니다. 로그 파일이 0.5GB를 초과하는 경우 AWS Glue 4.0 이상 버전에서 작업 실행에 대한 로그 롤링 지원을 활성화하여 로그 아카이브, 분석 및 문제 해결을 간소화할 수 있습니다.

AWS Glue 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Spark UI를 활성화할 수 있습니다. Spark UI를 사용하면 AWS Glue 개발 엔드포인트의 AWS Glue ETL 작업과 Spark 애플리케이션은 Amazon Simple Storage Service(Amazon S3)에서 지정한 위치에 Spark 이벤트 로그를 백업할

수 있습니다. 작업이 작동 중일 때와 완료된 후 모두 실시간으로 Amazon S3에서 Spark UI를 통해 백업된 이벤트 로그를 사용할 수 있습니다. 로그가 Amazon S3에 남아 있는 동안 AWS Glue 콘솔의 Spark UI에서 해당 로그를 볼 수 있습니다.

권한

AWS Glue 콘솔에서 Spark UI를 사용하려면 `UseGlueStudio`를 사용하거나 모든 개별 서비스 API를 추가할 수 있습니다. Spark UI를 완전히 사용하려면 모든 API가 필요하지만, 사용자는 세분화된 액세스를 위해 IAM 권한에 해당 서비스 API를 추가하여 SparkUI 기능에 액세스할 수 있습니다.

`RequestLogParsing`은 로그 구문 분석을 수행하므로 가장 중요합니다. 나머지 API는 각각의 구문 분석된 데이터를 읽는 데 사용됩니다. 예를 들어 `GetStages`는 Spark 작업의 모든 단계에 대한 데이터에 액세스할 수 있습니다.

`UseGlueStudio`에 매핑된 Spark UI 서비스 API 목록은 아래 샘플 정책에 나와 있습니다. 아래 정책은 Spark UI 기능만 사용할 수 있는 액세스 권한을 제공합니다. Amazon S3 및 IAM과 같은 권한을 더 추가하려면 AWS Glue Studio에 대한 사용자 지정 IAM 정책 만들기를 참조하세요.

`UseGlueStudio`에 매핑된 Spark UI 서비스 API 목록은 아래 샘플 정책에 나와 있습니다. Spark UI 서비스 API를 사용할 때는 다음 네임스페이스를 사용하세요. `glue:<ServiceAPI>`

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowGlueStudioSparkUI",
   "Effect": "Allow",
            "Action": [
                "glue:RequestLogParsing",
                "glue:GetLogParsingStatus",
                "glue:GetEnvironment",
                "glue:GetJobs",
                "glue:GetJob",
                "glue:GetStage",
                "glue:GetStages",
                "glue:GetStageFiles",
                "glue:BatchGetStageFiles",
                "glue:GetStageAttempt",
                "glue:GetStageAttemptTaskList",
                "glue:GetStageAttemptTaskSummary",
                "glue:GetExecutors",
                "glue:GetExecutorsThreads",
                "glue:GetStorage",
```

```
                "glue:GetStorageUnit",
                "glue:GetQueries",
                "glue:GetQuery"
            ],
    "Resource": [
     "*"
    ]
   }
 ]
}
```

제한 사항

- AWS Glue 콘솔의 Spark UI는 레거시 로그 형식이므로 2023년 11월 20일 이전에 발생한 작업 실행에는 사용할 수 없습니다.
- AWS Glue 콘솔의 Spark UI는 스트리밍 작업에서 기본적으로 생성되는 로그와 같이 AWS Glue 4.0에 대한 로그 롤링을 지원합니다. 생성된 모든 롤링된 로그 이벤트 파일의 최대 합계는 2GB입니다. 롤링된 로그를 지원하지 않는 AWS Glue 작업의 경우 SparkUI에 대해 지원되는 최대 로그 이벤트 파일 크기는 0.5GB입니다.
- VPC에서만 액세스할 수 있는 Amazon S3 버킷에 저장된 Spark 이벤트 로그에 대해서는 서버리스 Spark UI를 사용할 수 없습니다.

예: Apache Spark 웹 UI

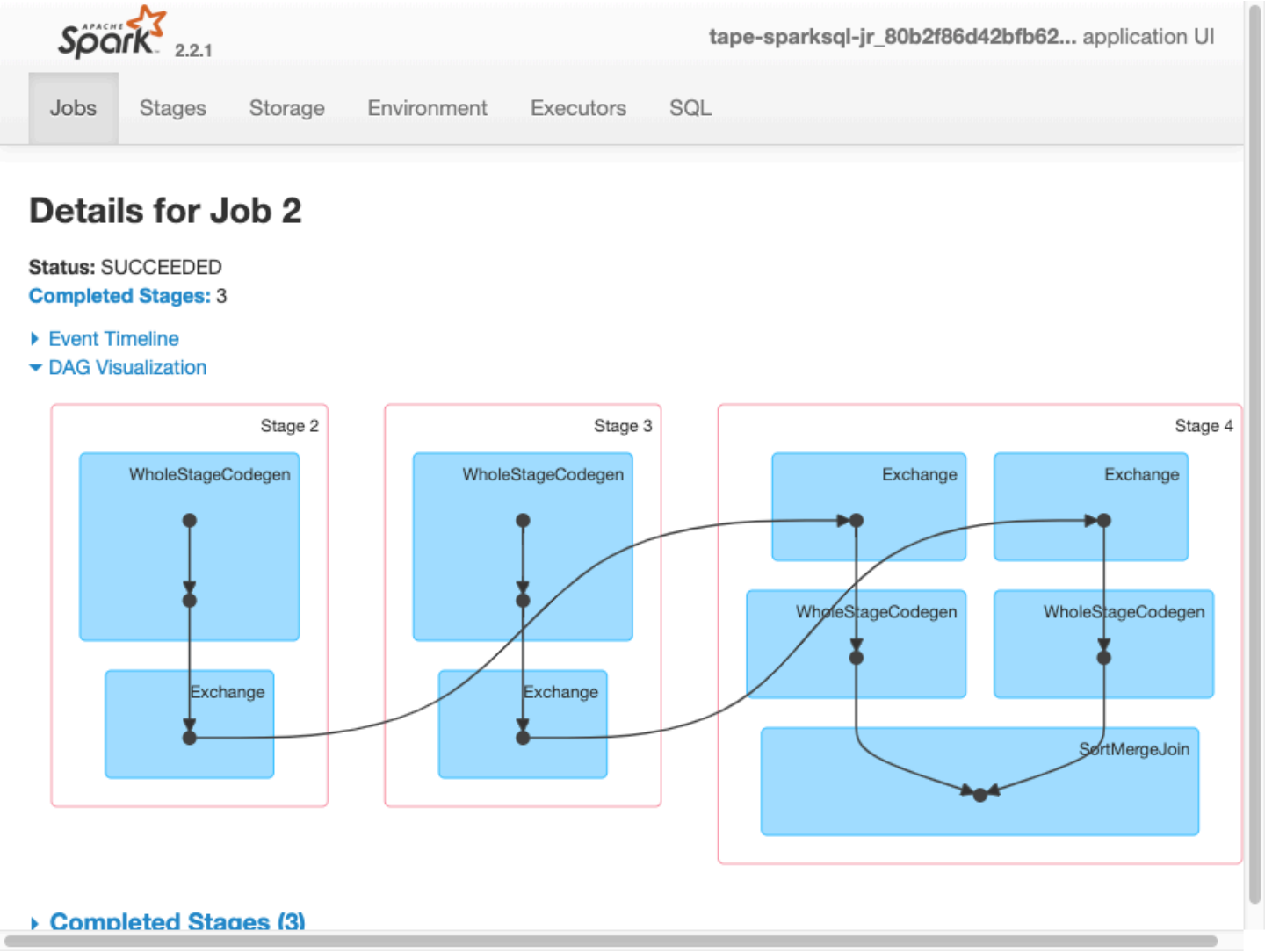
이 예제에서는 Spark UI를 사용하여 작업 성과를 이해하는 방법을 보여줍니다. 스크린샷은 자체 관리형 Spark 기록 서버에서 제공하는 Spark 웹 UI를 보여줍니다. AWS Glue 콘솔의 Spark UI에서도 비슷한 화면을 볼 수 있습니다. Spark 웹 UI 사용에 대한 자세한 내용은 Spark 설명서의 웹 UI를 참조하십시오.

다음은 Spark 애플리케이션에서 두 데이터 소스의 데이터를 읽고 조인 변환을 수행하여 Parquet 형식으로 Amazon S3에 쓰는 예제입니다.

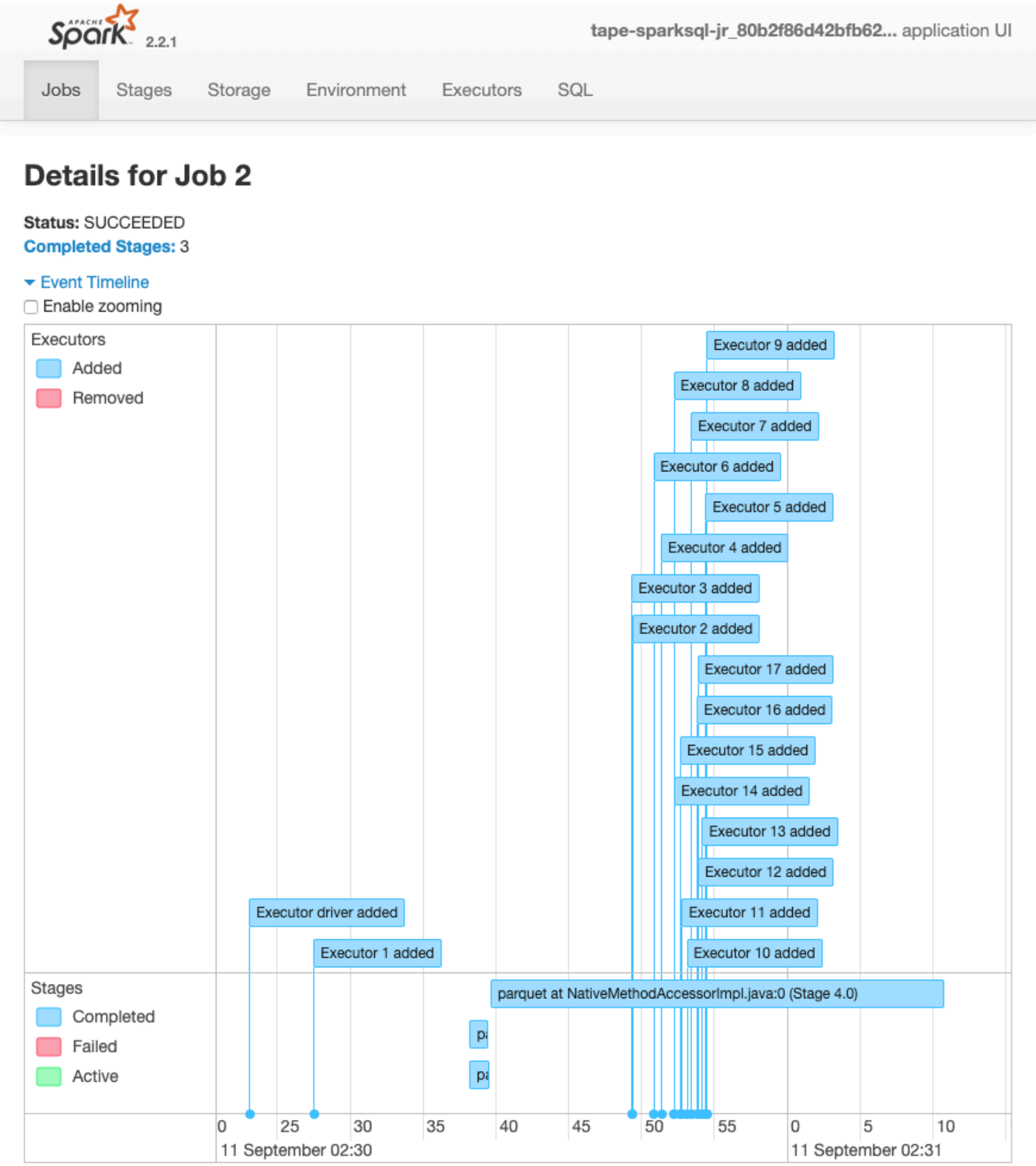
```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import count, when, expr, col, sum, isnull
```

```
from pyspark.sql.functions import countDistinct
from awsglue.dynamicframe import DynamicFrame

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'])

df_persons = spark.read.json("s3://awsglue-datasets/examples/us-legislators/all/
persons.json")
df_memberships = spark.read.json("s3://awsglue-datasets/examples/us-legislators/all/
memberships.json")

df_joined = df_persons.join(df_memberships, df_persons.id == df_memberships.person_id,
 'fullouter')
df_joined.write.parquet("s3://aws-glue-demo-sparkui/output/")

job.commit()
```

다음 DAG 시각화는 이 Spark 작업의 여러 단계를 보여줍니다.

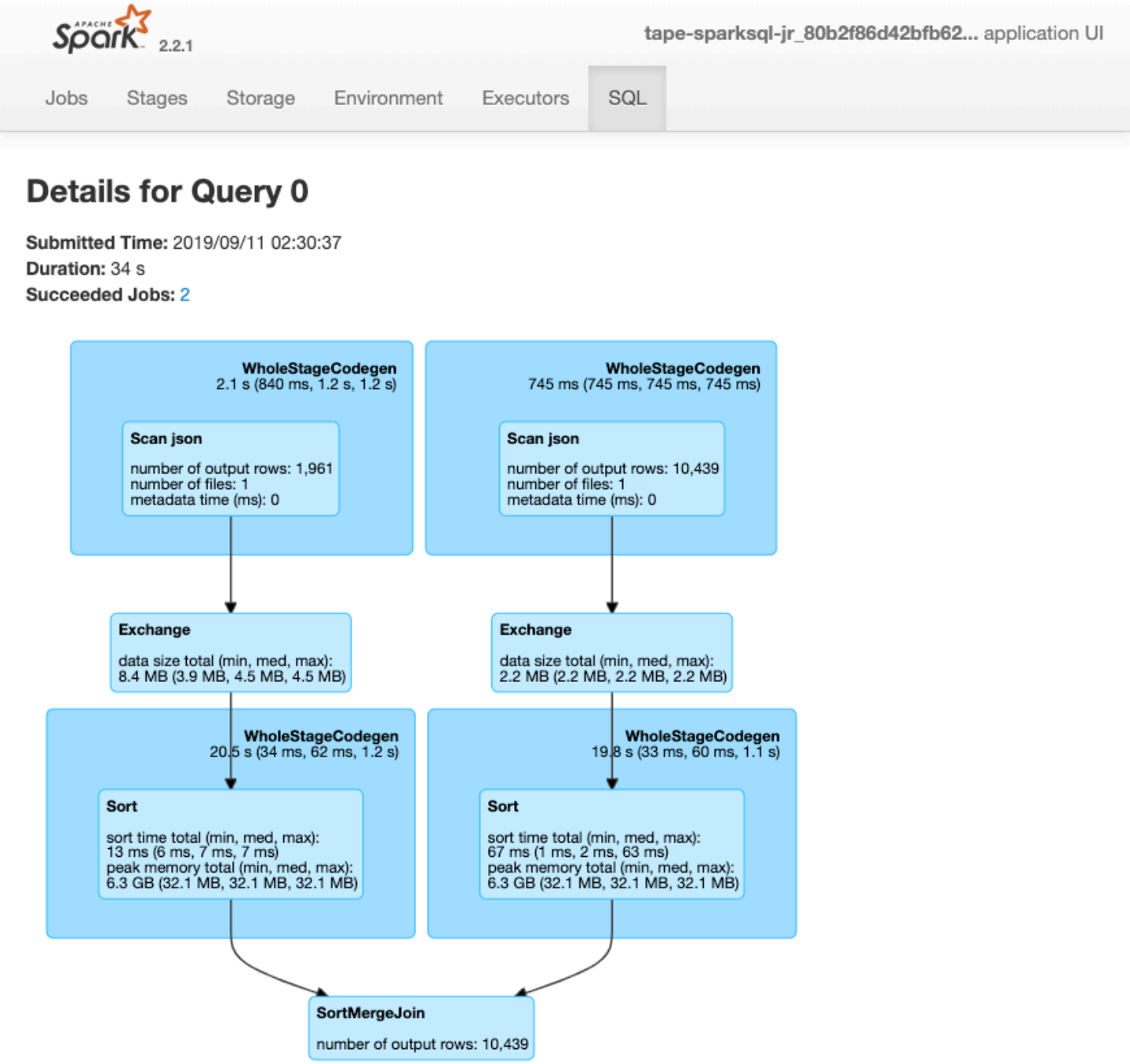


작업에 대한 다음 이벤트 타임라인은 다양한 Spark 실행기의 시작, 실행 및 종료를 보여줍니다.


Spark 2.2.1
tape-sparksql-jr_80b2f86d42bfb62... application UI
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Job 2
Status: SUCCEEDED
Completed Stages: 3
Event Timeline
Enable zooming
Executors
Added
Removed
Executor 9 added
Executor 8 added
Executor 7 added
Executor 6 added
Executor 5 added
Executor 4 added
Executor 3 added
Executor 2 added
Executor 17 added
Executor 16 added
Executor 15 added
Executor 14 added
Executor 13 added
Executor 12 added
Executor 11 added
Executor 10 added
Executor driver added
Executor 1 added
Stages
Completed
Failed
Active
parquet at NativeMethodAccessorImpl.java:0 (Stage 4.0)
0
25
30
35
40
45
50
55
0
5
10
11 September 02:30
11 September 02:31
DAG Visualization
Completed Stages (3)

다음 화면은 SparkSQL 쿼리 계획의 세부 정보를 보여줍니다.

- 구문 분석된 논리적 계획
- 분석된 논리적 계획
- 최적화된 논리적 계획
- 실행할 물리적 계획


Spark 2.2.1
tape-sparksql-jr_80b2f86d42bfb62... application UI
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Query 0
Submitted Time: 2019/09/11 02:30:37
Duration: 34 s
Succeeded Jobs: 2
WholeStageCodegen
2.1 s (840 ms, 1.2 s, 1.2 s)
Scan json
number of output rows: 1,961
number of files: 1
metadata time (ms): 0
WholeStageCodegen
745 ms (745 ms, 745 ms, 745 ms)
Scan json
number of output rows: 10,439
number of files: 1
metadata time (ms): 0
Exchange
data size total (min, med, max):
8.4 MB (3.9 MB, 4.5 MB, 4.5 MB)
Exchange
data size total (min, med, max):
2.2 MB (2.2 MB, 2.2 MB, 2.2 MB)
WholeStageCodegen
20.5 s (34 ms, 62 ms, 1.2 s)
Sort
sort time total (min, med, max):
13 ms (6 ms, 7 ms, 7 ms)
peak memory total (min, med, max):
6.3 GB (32.1 MB, 32.1 MB, 32.1 MB)
WholeStageCodegen
19.8 s (33 ms, 60 ms, 1.1 s)
Sort
sort time total (min, med, max):
67 ms (1 ms, 2 ms, 63 ms)
peak memory total (min, med, max):
6.3 GB (32.1 MB, 32.1 MB, 32.1 MB)
SortMergeJoin
number of output rows: 10,439
Details
== Parsed Logical Plan ==
Join FullOuter, (id#14 = person_id#50)
:-
Relation[birth_date#8,contact_details#9,death_date#10,family_name#11,gender#12,given_name#13,id#14,identifiers#15
,image#16,images#17,links#18,name#19,other_names#20,sort_name#21] json
+-
Relation[area_id#45,end_date#46,legislative_period_id#47,on_behalf_of_id#48,organization_id#49,person_id#50,role#
51,start_date#52] json
== Analyzed Logical Plan ==
birth_date: string, contact_details: array<struct<type:string,value:string>>, death_date: string, family_name:
string, gender: string, given_name: string, id: string, identifiers:
array<struct<identifier:string,scheme:string>>, image: string, images: array<struct<url:string>>, links:
array<struct<note:string,url:string>>, name: string, other_names:
array<struct<lang:string,name:string,note:string>>, sort_name: string, area_id: string, end_date: string,
legislative_period_id: string, on_behalf_of_id: string, organization_id: string, person_id: string, role: string,
start_date: string
Join FullOuter, (id#14 = person_id#50)

주제

- AWS Glue 작업을 위한 Apache Spark 웹 UI 사용 설정
- Spark 기록 서버 시작

AWS Glue 작업을 위한 Apache Spark 웹 UI 사용 설정

Apache Spark 웹 UI를 사용하여 AWS Glue 작업 시스템에서 실행 중인 AWS Glue ETL 작업을 모니터링하고 디버그할 수 있습니다. AWS Glue 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Spark UI를 구성할 수 있습니다.

30초마다 AWS Glue가 Spark 이벤트 로그를 지정한 Amazon S3 경로로 백업합니다.

주제

- Spark UI 구성(콘솔)
- Spark UI 구성(AWS CLI)
- 노트북을 사용하여 세션에 대한 Spark UI 구성
- 롤링 로그 활성화

Spark UI 구성(콘솔)

AWS Management Console을 사용하여 Spark UI를 구성하려면 다음 단계를 따르세요. AWS Glue 작업을 생성할 때 Spark UI는 기본으로 활성화됩니다.

작업을 생성하거나 편집할 때 Spark UI를 켜려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서, 작업을 선택합니다.
3. 작업 추가를 선택하거나 기존 작업을 선택합니다.
4. 작업 세부 정보에서 고급 속성을 엽니다.
5. Spark UI 탭에서 Amazon S3에 Spark UI 로그 쓰기를 선택합니다.
6. 작업의 Spark 이벤트 로그를 저장할 Amazon S3 경로를 지정합니다. 작업에서 보안 구성을 사용하는 경우 암호화는 Spark UI 로그 파일에도 적용됩니다. 자세한 내용은 AWS Glue에서 작성한 데이터 암호화 단원을 참조하십시오.
7. Spark UI 로깅 및 모니터링 구성에서:

- AWS Glue 콘솔에서 볼 로그를 생성하는 경우 표준을 선택합니다.
- Spark 기록 서버에서 볼 로그를 생성하는 경우 레거시를 선택합니다.
- 둘 다 생성하도록 선택할 수도 있습니다.

Spark UI 구성(AWS CLI)

AWS CLI를 사용하여 AWS Glue 콘솔에서 Spark UI로 볼 로그를 생성하려면 다음 작업 파라미터를 AWS Glue 작업에 전달합니다. 자세한 내용은 the section called “작업 파라미터” 단원을 참조하십시오.

```
'--enable-spark-ui': 'true',
'--spark-event-logs-path': 's3://s3-event-log-path'
```

기존 위치에 로그를 배포하려면 `--enable-spark-ui-legacy-path` 파라미터를 `"true"`로 설정합니다. 두 가지 형식으로 로그를 생성하지 않으려면 `--enable-spark-ui` 매개변수를 제거하십시오.

노트북을 사용하여 세션에 대한 Spark UI 구성

> **Warning**
>
> AWS Glue 대화형 세션은 현재 콘솔의 Spark UI를 지원하지 않습니다. Spark 기록 서버를 구성합니다.

AWS Glue 노트북을 사용하는 경우 세션을 시작하기 전에 SparkUI 구성을 설정합니다. 이렇게 하려면 `%%configure` 셀 매직을 사용합니다.

```
%%configure { “--enable-spark-ui”: “true”, “--spark-event-logs-path”: “s3://path” }
```

롤링 로그 활성화

AWS Glue 작업에 SparkUI 및 로그 이벤트 파일 롤링을 활성화하면 다음과 같은 몇 가지 이점이 있습니다.

- 로그 이벤트 파일 롤링 - 로그 이벤트 파일 롤링이 활성화된 상태에서 AWS Glue는 작업 실행의 각 단계에 대해 별도의 로그 파일을 생성하므로 특정 단계 또는 변환과 관련된 문제를 쉽게 식별하고 해결할 수 있습니다.

- 더 나은 로그 관리 - 로그 이벤트 파일을 롤링하면 로그 파일을 더 효율적으로 관리하는 데 도움이 됩니다. 잠재적으로 큰 단일 로그 파일을 보유하는 대신, 작업 실행 단계에 따라 더 작고 관리 가능한 파일로 로그가 분할됩니다. 이렇게 하면 로그 아카이빙, 분석 및 문제 해결을 간소화할 수 있습니다.
- 내결함성 개선 - AWS Glue 작업이 실패하거나 중단되는 경우 로그 이벤트 파일 롤링은 마지막 성공 단계에 대한 중요한 정보를 제공할 수 있으므로 처음부터 시작하는 대신 해당 시점부터 작업을 더 쉽게 재개할 수 있습니다.
- 비용 최적화 - 로그 이벤트 파일 롤링을 활성화하면 로그 파일과 관련된 스토리지 비용을 절감할 수 있습니다. 잠재적으로 큰 단일 로그 파일을 저장하는 대신 더 작고 관리 가능한 로그 파일을 저장합니다. 특히 장기 실행 또는 복잡한 작업의 경우 더 비용 효율적일 수 있습니다.

새 환경에서 사용자는 다음을 통해 로그 롤링을 명시적으로 활성화할 수 있습니다.

```
'–conf': 'spark.eventLog.rolling.enabled=true'
```

또는

```
'–conf': 'spark.eventLog.rolling.enabled=true –conf
spark.eventLog.rolling.maxFileSize=128m'
```

로그 롤링이 활성화된 경우 `spark.eventLog.rolling.maxFileSize`에서는 롤오버하기 전 이벤트 로그 파일의 최대 크기를 지정합니다. 지정하지 않으면 이 선택적 파라미터의 기본값은 128MB입니다. 최소 10MB입니다.

생성된 모든 롤링된 로그 이벤트 파일의 최대 합계는 2GB입니다. 로그 롤링을 지원하지 않는 AWS Glue 작업의 경우 SparkUI에 대해 지원되는 최대 로그 이벤트 파일 크기는 0.5GB입니다.

추가 구성을 전달하여 스트리밍 작업에 대한 롤링 로그를 끌 수 있습니다. 로그 파일이 매우 크면 유지 관리 비용이 많이 들 수 있습니다.

롤링 로그를 끄려면 다음 구성을 제공하십시오.

```
'--spark-ui-event-logs-path': 'true',
'--conf': 'spark.eventLog.rolling.enabled=false'
```

Spark 기록 서버 시작

Spark 기록 서버를 사용하여 자체 인프라에서 Spark 로그를 시각화할 수 있습니다. AWS Glue 콘솔에서는 AWS Glue 4.0 이상 버전의 AWS Glue 작업 실행에 대한 동일한 시각화를 레거시 형식이 아닌 표

준 형식으로 생성된 로그와 함께 볼 수 있습니다. 자세한 내용은 the section called “Spark UI로 모니터링” 단원을 참조하십시오.

EC2 인스턴스에서 서버를 호스팅하는 AWS CloudFormation 템플릿을 사용하여 Spark 기록 서버를 시작하거나 Docker를 사용하여 로컬로 시작할 수 있습니다.

주제

- AWS CloudFormation을 사용하여 Spark 기록 서버 시작 및 Spark UI 보기
- Docker를 사용하여 Spark 기록 서버 시작 및 Spark UI 보기

AWS CloudFormation을 사용하여 Spark 기록 서버 시작 및 Spark UI 보기

AWS CloudFormation 템플릿을 사용하여 Apache Spark 기록 서버를 시작하고 Spark 웹 UI를 볼 수 있습니다. 이 템플릿은 요구 사항에 맞게 수정해야 하는 샘플입니다.

AWS CloudFormation을 사용하여 Spark History Server를 시작하고 Spark UI를 보려면

1. 다음 표에서 스택 시작 버튼 중 하나를 선택하십시오. 그러면 AWS CloudFormation 콘솔에서 스택이 시작됩니다.

리전	시작
미국 동부(오하이오)	Launch Stack
미국 동부(버지니아 북부)	Launch Stack
미국 서부(캘리포니아 북부)	Launch Stack
미국 서부(오레곤)	Launch Stack
아프리카(케이프타운)	Launch Stack
아시아 태평양(홍콩)	Launch Stack
아시아 태평양(뭄바이)	Launch Stack

리전	시작
아시아 태평양(오사카)	Launch Stack
아시아 태평양(서울)	Launch Stack
아시아 태평양(싱가포르)	Launch Stack
아시아 태평양(시드니)	Launch Stack
아시아 태평양(도쿄)	Launch Stack
캐나다(중부)	Launch Stack
유럽(프랑크푸르트)	Launch Stack
유럽(아일랜드)	Launch Stack
유럽(런던)	Launch Stack
유럽(밀라노)	Launch Stack
유럽(파리)	Launch Stack
유럽(스톡홀름)	Launch Stack
중동(바레인)	Launch Stack
남아메리카(상파울루)	Launch Stack

2. Specify template(템플릿 지정) 페이지에서 다음을 선택합니다.

3. 스택 세부 정보 지정 페이지에서 스택 이름을 입력합니다. 파라미터(Parameters) 아래에 추가 정보를 입력합니다.

   a. Spark UI 구성

      다음 정보를 제공합니다.

      - [IP 주소 범위(IP address range)] - Spark UI를 보는 데 사용할 수 있는 IP 주소 범위입니다. 특정 IP 주소 범위의 액세스를 제한하려면 사용자 지정 값을 사용해야 합니다.
      - [기록 서버 포트(History server port)] - Spark UI의 포트입니다. 기본 값을 사용할 수 있습니다.
      - [이벤트 로그 디렉터리(Event log directory)] - AWS Glue 작업 또는 개발 엔드포인트에서 Spark 이벤트 로그가 저장되는 위치를 선택합니다. 이벤트 로그 경로 체계에 **`s3a://`**를 사용해야 합니다.
      - [Spark 패키지 위치(Spark package location)] - 기본값을 사용할 수 있습니다.
      - [키 스토어 경로(Keystore path)] - HTTPS의 SSL/TLS 키 스토어 경로입니다. 사용자 지정 키 스토어 파일을 사용하려면 여기에서 S3 경로 `s3://path_to_your_keystore_file`을 지정하면 됩니다. 이 파라미터를 비워두면 자체 서명된 인증서 기반 키 스토어가 생성되어 사용됩니다.
      - 키 스토어 암호(Keystore password) - HTTPS에 대한 SSL/TLS 키 스토어 암호를 입력합니다.

   b. EC2 인스턴스 구성

      다음 정보를 제공합니다.

      - [인스턴스 유형(Instance type)] - Spark 기록 서버를 호스팅하는 Amazon EC2 인스턴스 유형입니다. 이 템플릿은 계정에서 Amazon EC2 인스턴스를 시작하므로 계정에 별도로 Amazon EC2 비용이 청구됩니다.
      - [최신 AMI ID(Latest AMI ID)] - Spark 기록 서버 인스턴스에 대한 Amazon Linux 2의 AMI ID입니다. 기본 값을 사용할 수 있습니다.
      - [VPC ID] - Spark 기록 서버 인스턴스의 Virtual Private Cloud(VPC) ID입니다. 계정에 제공된 모든 VPC를 사용할 수 있습니다. 기본 네트워크 ACL과 함께 기본 VPC를 사용하지 않는 것이 좋습니다. 자세한 내용은 Amazon VPC 사용 설명서의 기본 VPC 및 서브넷과 VPC 생성을 참조하세요.
      - [서브넷 ID(Subnet ID)] - Spark 기록 서버 인스턴스의 ID입니다. VPC의 모든 서브넷을 사용할 수 있습니다. 클라이언트에서 서브넷으로 네트워크에 도달할 수 있어야 합니다. 인터넷

을 통해 액세스하려면 라우팅 테이블에 인터넷 게이트웨이가 있는 퍼블릭 서브넷을 사용해야 합니다.

c. Next(다음)를 선택합니다.

4. 스택 옵션 구성(Configure stack options) 페이지에서 현재 사용자 자격 증명을 사용하여 CloudFormation이 스택의 리소스를 생성, 수정 또는 삭제하는 방식을 확인하려면 다음(Next)을 선택합니다. 권한 섹션에서 현재 사용자 권한 대신 사용할 역할을 지정하고 다음을 선택할 수도 있습니다.
5. 검토 페이지에서 설정을 확인합니다.

   AWS CloudFormation에서 IAM 리소스를 생성할 수 있음을 승인합니다를 선택하고 스택 생성을 선택합니다.
6. 스택이 생성되기를 기다립니다.
7. 출력 탭을 클릭합니다.

   a. 퍼블릭 서브넷을 사용하는 경우 SparkUiPublicUrl의 URL을 복사하십시오.

   b. 프라이빗 서브넷을 사용하는 경우 SparkUiPrivateUrl의 URL을 복사하십시오.
8. 웹 브라우저를 열고 URL에 붙여 넣습니다. 지정된 포트에서 HTTPS를 사용하여 서버에 액세스할 수 있습니다. 브라우저가 서버 인증을 인식하지 못할 수 있어 서버 보호를 재정의하여 실행해야 합니다.

Docker를 사용하여 Spark 기록 서버 시작 및 Spark UI 보기

로컬 액세스를 선호하는 경우(Apache Spark 기록 서버에 EC2 인스턴스가 없는 경우) Docker를 사용하여 Apache Spark 기록 서버를 시작하고 Spark UI를 로컬로 볼 수도 있습니다. 이 Dockerfile은 요구 사항에 맞게 수정해야 하는 샘플입니다.

사전 조건

노트북에 도커를 설치하는 방법에 대한 자세한 내용은 Docker Engine community를 참조하십시오.

Docker를 사용하여 로컬로 Spark History Server를 시작하고 Spark UI를 보려면

1. GitHub에서 파일을 다운로드하십시오.

   AWS Glue 코드 샘플에서 Dockerfile과 `pom.xml`을 다운로드합니다.
2. AWS에 액세스하는 데 사용자 자격 증명을 사용할지 또는 페더레이션 사용자 자격 증명을 사용할지 결정합니다.

- 현재 사용자 자격 증명을 사용하여 AWS에 액세스하려면 `docker run` 명령의 `AWS_ACCESS_KEY_ID`와 `AWS_SECRET_ACCESS_KEY`에 사용할 값을 가져옵니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.
- SAML 2.0 페더레이션 사용자를 사용하여 AWS에 액세스하려면 `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` 및 `AWS_SESSION_TOKEN`의 값을 가져옵니다. 자세한 내용은 임시 보안 자격 증명 요청을 참조하세요.

3. `docker run` 명령에서 사용할 이벤트 로그 디렉터리의 위치를 결정합니다.
4. 이름 `glue/sparkui` 및 태그 `latest`를 사용하여 로컬 디렉터리의 파일을 통해 Docker 이미지를 구축합니다.

```
$ docker build -t glue/sparkui:latest .
```

5. Docker 컨테이너를 생성하고 시작합니다.

   다음 명령에서는 이전 2단계와 3단계에서 얻은 값을 사용합니다.

   a. 사용자 자격 증명을 사용하여 Docker 컨테이너를 생성하려면 다음과 비슷한 명령을 사용합니다.

```
docker run -itd -e SPARK_HISTORY_OPTS="$SPARK_HISTORY_OPTS -
Dspark.history.fs.logDirectory=s3a://path_to_eventlog
 -Dspark.hadoop.fs.s3a.access.key=AWS_ACCESS_KEY_ID -
Dspark.hadoop.fs.s3a.secret.key=AWS_SECRET_ACCESS_KEY"
 -p 18080:18080 glue/sparkui:latest "/opt/spark/bin/spark-class
 org.apache.spark.deploy.history.HistoryServer"
```

   b. 임시 자격 증명을 사용하여 Docker 컨테이너를 생성하려면 `org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider`를 공급자로 사용하고 2단계에서 얻은 자격 증명 값을 제공합니다. 자세한 내용은 Hadoop: Amazon Web Services와 통합 설명서의 TemporaryAWSCredentialsProvider로 세션 자격 증명 사용을 참조하세요.

```
docker run -itd -e SPARK_HISTORY_OPTS="$SPARK_HISTORY_OPTS -
Dspark.history.fs.logDirectory=s3a://path_to_eventlog
 -Dspark.hadoop.fs.s3a.access.key=AWS_ACCESS_KEY_ID -
Dspark.hadoop.fs.s3a.secret.key=AWS_SECRET_ACCESS_KEY
 -Dspark.hadoop.fs.s3a.session.token=AWS_SESSION_TOKEN
 -
Dspark.hadoop.fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.TemporaryAWSCred
```

```
-p 18080:18080 glue/sparkui:latest "/opt/spark/bin/spark-class
org.apache.spark.deploy.history.HistoryServer"
```

> **Note**
>
> 이러한 구성 파라미터는 Hadoop-AWS 모듈에서 나옵니다. 사용 사례를 바탕으로 특정 구성을 추가해야 할 수 있습니다. 예를 들어, 격리된 리전에 있는 사용자는 `spark.hadoop.fs.s3a.endpoint`를 구성해야 합니다.

6. 브라우저에서 `http://localhost:18080`을 열어 로컬로 Spark UI를 봅니다.

AWS Glue 작업 실행 인사이트를 사용한 모니터링

AWS Glue 작업 실행 인사이트는 작업 디버깅 및 AWS Glue 작업의 최적화를 간소화해 주는 AWS Glue의 기능입니다. AWS Glue는 AWS Glue 작업을 모니터링할 수 있도록 Spark UI와 CloudWatch 로그 및 지표를 제공합니다. 이 기능을 사용하면 AWS Glue 작업의 실행에 대해 다음과 같은 정보를 확인할 수 있습니다.

- 실패가 발생한 AWS Glue 작업 스크립트의 행 번호
- 작업 실패가 발생하기 직전에 Spark 쿼리 계획에서 마지막으로 실행된 Spark 작업
- 실패와 관련된 Spark 예외 이벤트의 시간순으로 된 로그 스트림
- 근본 원인 분석 및 문제 해결을 위한 권장 작업(예: 스크립트 튜닝)
- 일반적인 Spark 이벤트(Spark 작업과 관련된 로그 메시지)와 근본 원인 해결을 위한 권장 작업

이러한 인사이트는 AWS Glue 작업의 CloudWatch 로그에 추가된 2가지 로그 스트림을 사용하여 확인할 수 있습니다.

요구 사항

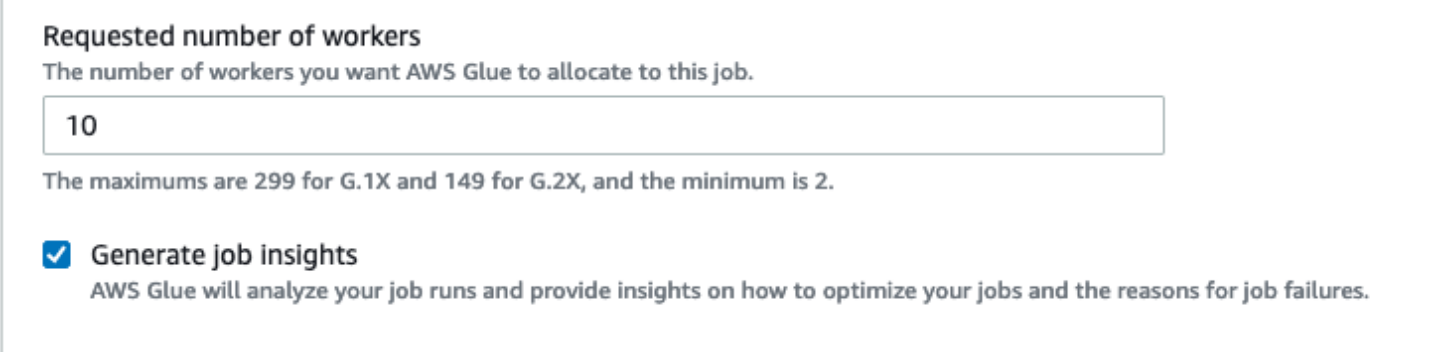
AWS Glue 작업 실행 인사이트 기능은 AWS Glue 버전 2.0, 3.0, 4.0, 5.0에서 사용할 수 있습니다. 기존 작업의 마이그레이션 가이드에 따라 이전 버전의 AWS Glue에서 업그레이드할 수 있습니다.

AWS Glue ETL 작업에 대해 작업 실행 인사이트 활성화

작업 실행 인사이트는 AWS Glue Studio 또는 CLI를 통해 활성화할 수 있습니다.

AWS Glue Studio

AWS Glue Studio를 통해 작업을 생성할 때는 Job Details(작업 세부 정보) 탭에서 작업 실행 인사이트를 활성화 또는 비활성화할 수 있습니다. 작업 인사이트 생성 상자가 선택되어 있는지 확인합니다.



명령줄

CLI를 통해 작업을 생성할 때는 새로 추가된 단일 작업 파라미터 `--enable-job-insights = true`를 사용하여 작업 실행을 시작할 수 있습니다.

기본적으로 작업 실행 인사이트 로그 스트림은 AWS Glue 연속 로깅에서 사용되는 것과 동일한 기본 로그 그룹인 `/aws-glue/jobs/logs-v2/` 아래에 생성됩니다. 연속 로깅에 사용된 것과 동일한 인수 세트를 사용하여 사용자 지정 로그 그룹 이름, 로그 필터 및 로그 그룹 구성을 설정할 수 있습니다. 자세한 내용은 Enabling Continuous Logging for AWS Glue Jobs(작업에 대해 연속 로깅 활성화)를 참조하세요.

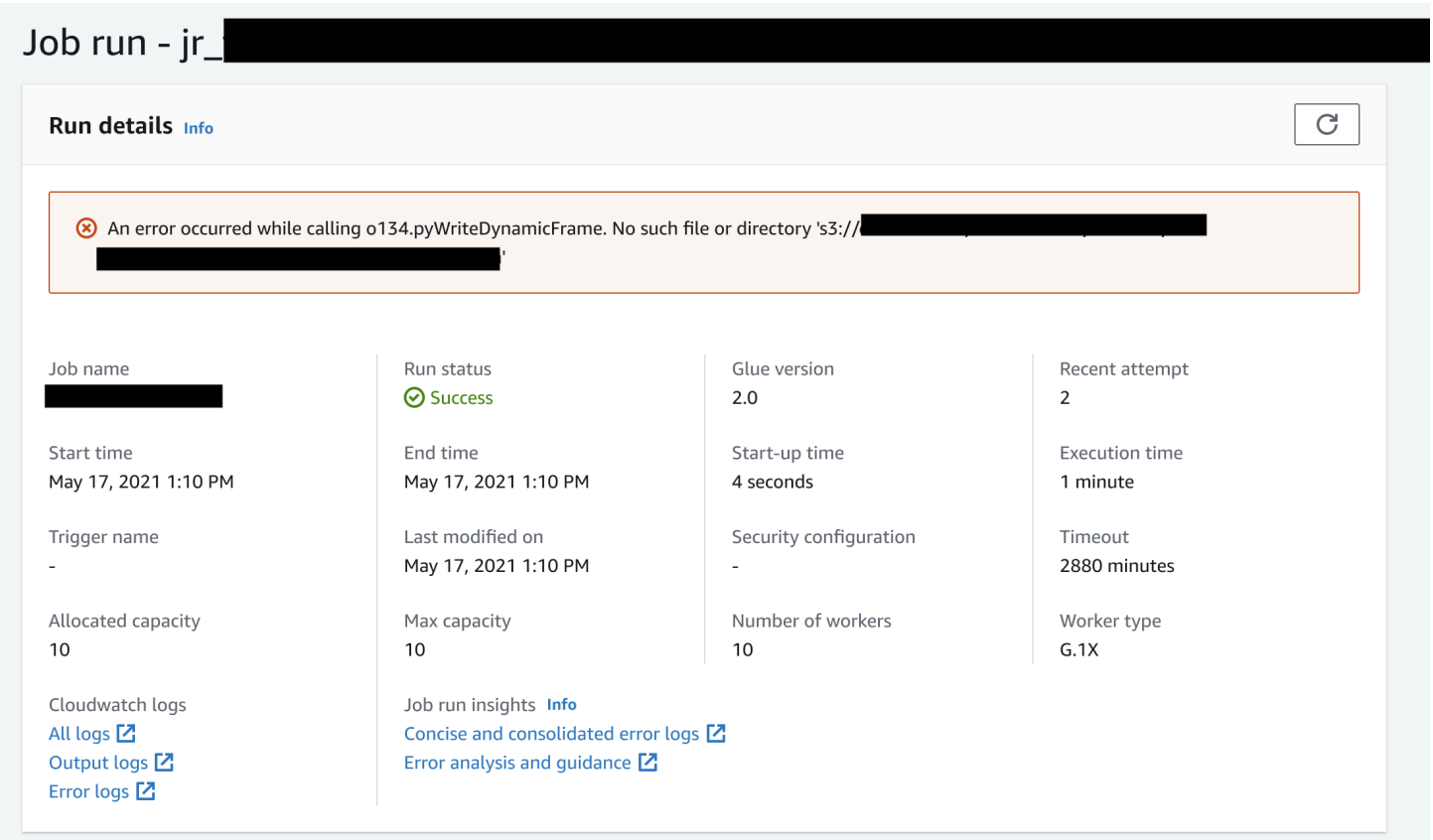
CloudWatch에서 작업 실행 인사이트 로그 스트림에 액세스

작업 실행 인사이트 기능이 활성화된 상태에서 작업 실행이 실패할 경우 2가지 로그 스트림이 생성될 수 있습니다. 작업이 성공적으로 완료되면 2가지 스트림 모두 생성되지 않습니다.

1. 예외 분석 로그 스트림: `<job-run-id>-job-insights-rca-driver`. 이 스트림은 다음과 같은 정보를 제공합니다.
   - 실패의 원인이 된 AWS Glue 작업 스크립트의 행 번호.
   - Spark 쿼리 계획(DAG)에서 마지막으로 실행된 Spark 작업.
   - Spark 드라이버 및 실행기의 예외와 관련 있는 시간순 이벤트. 필요한 경우 전체 오류 메시지, 실패한 Spark 작업과 해당 실행기 ID(특정 실행기의 로그 스트림을 살펴보는 데 사용)와 같은 세부 정보를 볼 수 있습니다.
2. 규칙 기반 인사이트 스트림:
   - 근본 원인 분석 및 오류 수정을 위한 권장 사항(예: 특정 작업 파라미터를 사용하여 성능 최적화).
   - 근본 원인 분석의 기반이 되는 관련 있는 Spark 이벤트 및 권장 작업.

> **Note**
>
> 첫 번째 스트림은 실패한 작업 실행의 예외 Spark 이벤트가 있는 경우에만 생성되고, 두 번째 스트림은 실패한 작업 실행에 대한 인사이트가 있는 경우에만 생성됩니다. 예를 들어, 작업이 성공적으로 완료되었다면 2가지 스트림 모두 생성되지 않고, 작업이 실패했으나 실패 시나리오와 일치하는 서비스 정의 규칙이 없는 경우에는 첫 번째 스트림만 생성됩니다.

작업이 AWS Glue Studio에서 생성되었다면 Job Run Details(작업 실행 세부 정보) 탭 아래의 Job run insights(작업 실행 인사이트)에 위 스트림의 링크가 각각 'Concise and consolidated error logs'(간결하고 통합된 오류 로그) 및 'Error analysis and guidance'(오류 분석 및 가이드)로 표시됩니다.



AWS Glue 작업 실행 인사이트의 예

이 섹션에서는 작업 실행 인사이트 기능을 사용하여 실패한 작업의 문제를 해결하는 방법을 예를 통해 살펴봅니다. 이 예제에서는 사용자가 AWS Glue 작업에서 데이터에 대한 기계 학습 모델을 분석 및 빌드하는 데 필요한 모듈(tensorflow)을 가져오지 않았습니다.

```
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
from pyspark.sql.types import *
from pyspark.sql.functions import udf,col

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

data_set_1 = [1, 2, 3, 4]
data_set_2 = [5, 6, 7, 8]

scoresDf = spark.createDataFrame(data_set_1, IntegerType())

def data_multiplier_func(factor, data_vector):
    import tensorflow as tf
    with tf.compat.v1.Session() as sess:
        x1 = tf.constant(factor)
        x2 = tf.constant(data_vector)
        result = tf.multiply(x1, x2)
        return sess.run(result).tolist()

data_multiplier_udf = udf(lambda x:data_multiplier_func(x, data_set_2),
 ArrayType(IntegerType(),False))
factoredDf = scoresDf.withColumn("final_value", data_multiplier_udf(col("value")))
print(factoredDf.collect())
```

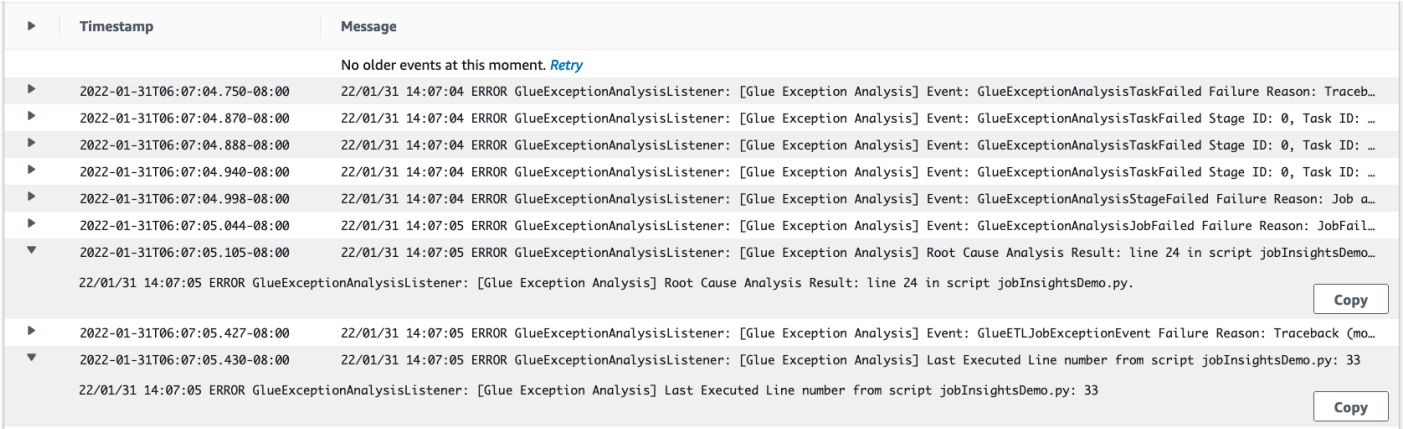
작업 실행 인사이트 기능이 없다면 작업이 실패해도 Spark에 의해 발생된 다음과 같은 메시지만 볼 수 있습니다.

```
An error occurred while calling o111.collectToPython. Traceback (most
recent call last):
```

이 메시지는 명확하지 않으므로 디버깅에 큰 도움이 되지 않습니다. 작업 실행 인사이트 기능은 다음과 같은 2가지 CloudWatch 로그 스트림을 통해 추가 인사이트를 제공합니다.

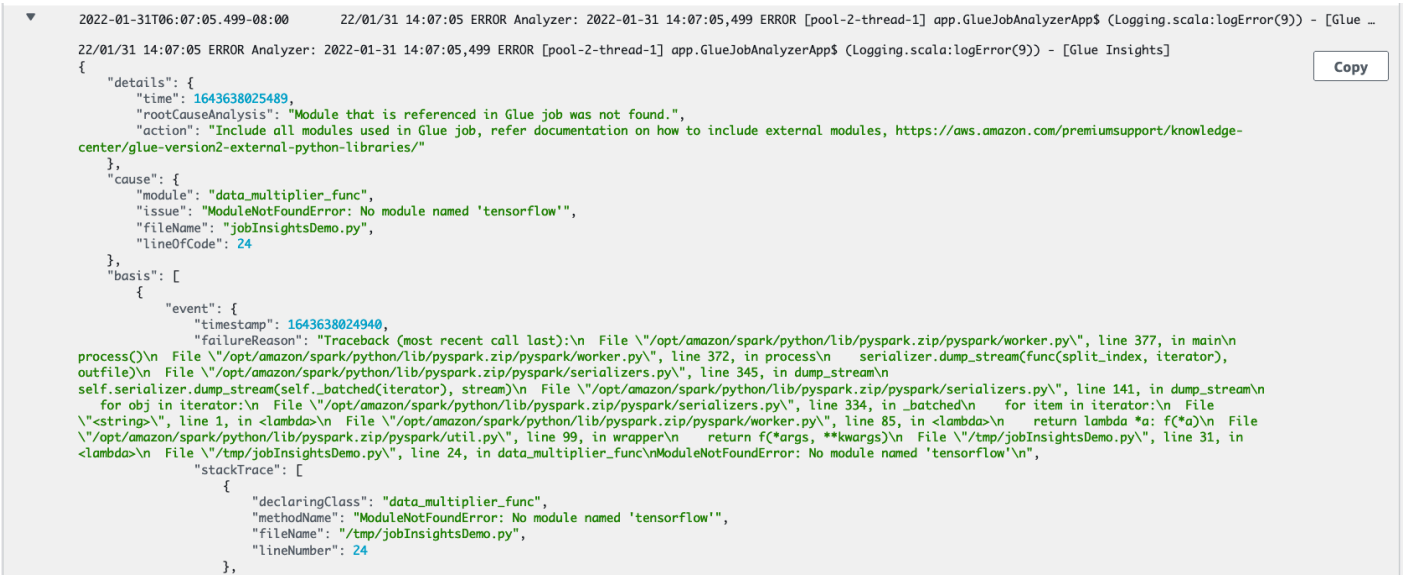
1. `job-insights-rca-driver` 로그 스트림:

- 예외 이벤트: 이 로그 스트림에서 Spark 드라이버 및 여러 분산 작업자로부터 수집된, 실패와 관련 있는 Spark 예외 이벤트를 볼 수 있습니다. 이러한 이벤트를 검토하면 여러 AWS Glue 작업자에 분산된 Spark 작업, 실행기 및 단계에서 결함 있는 코드가 실행되는 과정에서 예외가 전파된 상황을 시간순으로 살펴볼 수 있습니다.
- 행 번호: 이 로그 스트림에서 누락된 Python 모듈 가져오기를 호출하여 실패의 원인이 된 것이 21번 행이고 스크립트에서 마지막으로 실행된 Spark 작업 `collect()`의 호출은 24번 행인 것을 알 수 있습니다.



2. `job-insights-rule-driver` 로그 스트림:

- 근본 원인 분석 및 권장 사항: 스크립트에서 결함이 되는 행 번호와 마지막으로 실행된 행 번호 외에도 이 로그 스트림에서는 근본 원인 분석과 권장 사항(AWS Glue 작업에서 추가 Python 모듈을 사용하려면 AWS Glue 문서에 따라 필요한 작업 파라미터를 설정할 것)을 볼 수 있습니다.
- 기본 이벤트: 이 로그 스트림에서는 근본 원인을 유추할 수 있도록, 서비스 정의 규칙을 사용하여 평가된 Spark 예외 이벤트를 볼 수 있으며 권장 사항을 제공합니다.



Amazon CloudWatch를 사용한 모니터링

AWS Glue에서 원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리하는 Amazon CloudWatch를 통해 AWS Glue를 모니터링할 수 있습니다. 이러한 통계는 2주간 기록되므로 기록 정보에 액세스하여 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 기본적으로 AWS Glue 지표 데이터는 CloudWatch에 자동으로 전송됩니다. 자세한 내용은 AWS Glue 지표 및 Amazon CloudWatch User Guide의 What Is Amazon CloudWatch?를 참조하세요.

연속 로깅

AWS Glue에서도 AWS Glue 작업에 대한 실시간 지속 로깅을 지원합니다. 작업에 대한 지속 로깅이 활성화되어 있으면 AWS Glue 콘솔 또는 CloudWatch 콘솔 대시보드에서 실시간 로그를 볼 수 있습니다. 자세한 내용은 AWS Glue 작업의 로깅 단원을 참조하십시오.

관찰성 메트릭

작업 관찰성 지표가 활성화되면 작업이 실행될 때 추가 Amazon CloudWatch 지표가 생성됩니다. AWS Glue 관찰성 메트릭을 사용하면 AWS Glue 내부에서 일어나는 일에 대한 통찰력을 얻어 문제의 분류 및 분석을 개선할 수 있습니다.

주제

- Amazon CloudWatch 지표를 사용하여 AWS Glue 모니터링
- AWS Glue 작업 프로파일에서 Amazon CloudWatch 경보 설정
- AWS Glue 작업의 로깅
- AWS Glue 관찰성 메트릭을 사용한 모니터링

Amazon CloudWatch 지표를 사용하여 AWS Glue 모니터링

AWS Glue 작업 프로파일러를 사용하여 AWS Glue 작업을 프로파일링하고 모니터링할 수 있습니다. AWS Glue 작업에서 원시 데이터를 수집한 후 Amazon CloudWatch에 저장된 판독이 가능한 지표로 실시간에 가깝게 처리합니다. 이러한 통계는 CloudWatch에서 보관 및 집계되므로 기록 정보에 액세스하여 애플리케이션이 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다.

> **Note**
>
> 작업 지표를 활성화하고 CloudWatch 사용자 지정 지표를 생성하면 추가 요금이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

AWS Glue 지표 개요

AWS Glue는 사용자와 상호 작용할 때 지표를 CloudWatch로 전송합니다. AWS Glue 콘솔(기본 방법), CloudWatch 콘솔 대시보드 또는 AWS Command Line Interface(AWS CLI)를 사용하여 이러한 지표를 볼 수 있습니다.

AWS Glue 콘솔 대시보드를 사용하여 지표를 보려면

작업에 대한 요약이나 세부 그래프, 또는 작업 실행에 대한 세부 그래프를 볼 수 있습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 작업 실행 모니터링을 선택합니다.
3. 작업 실행에서 현재 실행 중인 작업을 중지하거나, 작업을 보거나, 작업 북마크를 되돌립니다.
4. 작업을 선택한 다음 실행 세부 정보 보기를 선택하여 작업 실행에 대한 추가 정보를 확인합니다.

CloudWatch 콘솔 대시보드를 사용하여 지표를 보려면

지표는 먼저 서비스 네임스페이스별로 그룹화된 다음 각 네임스페이스 내에서 다양한 차원 조합별로 그룹화됩니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. Glue 네임스페이스를 선택합니다.

AWS CLI을(를) 사용하여 지표를 보려면

- 명령 프롬프트에서 다음 명령을 사용합니다.

```
aws cloudwatch list-metrics --namespace Glue
```

AWS Glue는 30초마다 CloudWatch에 지표를 보고하며, CloudWatch 지표 대시보드는 1분마다 이러한 지표를 표시하도록 구성되어 있습니다. AWS Glue 지표는 이전에 보고한 값의 델타 값을 나타냅니다. 적절한 경우 지표 대시보드는 30초 값을 집계(합)하여 마지막 1분 전체에 대한 값을 얻습니다.

Spark 작업에 대한 AWS Glue 지표 동작

AWS Glue 지표는 스크립트에서 `GlueContext`를 초기화할 때 활성화되며 일반적으로 Apache Spark 작업이 끝날 때만 업데이트됩니다. 이러한 지표는 지금까지 완료된 모든 Spark 작업의 집계 값을 나타냅니다.

그러나 AWS Glue가 CloudWatch로 전달하는 Spark 지표는 일반적으로 보고되는 시점의 현재 상태를 나타내는 절대 값입니다. AWS Glue는 이 지표를 30초마다 CloudWatch로 보고하며, 일반적으로 지표 대시보드에는 마지막 1분 동안 받은 데이터 포인트의 평균이 표시됩니다.

AWS Glue 지표 이름은 모두 다음 접두사 유형 중 하나로 시작합니다.

- `glue.driver.` – 이 접두사로 시작하는 이름의 지표는 Spark 드라이버의 모든 실행기에서 집계된 AWS Glue 지표 또는 Spark 드라이버에 해당하는 Spark 지표를 나타냅니다.
- `glue.executorId.` – executorId는 특정 Spark 실행기의 번호입니다. 이는 로그에 나열된 실행기에 해당됩니다.
- `glue.ALL.` – 이름이 이 접두사로 시작하는 지표는 모든 Spark 실행기의 값을 집계합니다.

AWS Glue 지표

AWS Glue는 30초 간격으로 다음 지표를 프로파일링하여 CloudWatch로 전송하며, AWS Glue 지표 대시보드는 1분마다 이들 지표를 보고합니다.

지표	설명
`glue.driver.aggregate.bytesRead`	모든 실행기에서 실행되는 모든 완료된 Spark 태스크가 모든 데이터 원본에서 읽은 바이트 수. 유효 차원: `JobName`(AWS Glue 작업 이름), `JobRunId`(JobRun ID. 또는 `ALL`), `Type`(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 바이트 다음을 모니터링하는 데 사용: • 읽은 바이트 수.

지표	설명
	• 작업 진행 상황. • JDBC 데이터 원본. • 작업 북마크 문제. • 작업 실행 간 편차. 이 지표는 `glue.ALL.s3.filesystem.read_bytes` 지표와 동일한 방식으로 사용할 수 있으며(차이는 이 지표가 Spark 작업 종료 시 업데이트된다는 것) 비 S3 데이터 원본도 캡처합니다.
`glue.driver.aggregate.elapsedTime`	밀리초 단위의 ETL 경과 시간(작업 부트스트랩 시간은 포함하지 않음). 유효 차원: `JobName`(AWS Glue 작업 이름), `JobRunId`(JobRun ID. 또는 `ALL`), `Type`(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 밀리초 작업 실행이 평균적으로 얼마나 오래 실행되는지 측정하는 데 사용할 수 있습니다. 이 데이터는 다음과 같이 사용할 수 있습니다. • 스트래글러에 경보를 설정합니다. • 작업 실행 간 편차를 측정합니다.

지표	설명
glue.driver.aggregate.numCompletedStages	작업에서 완료된 단계 수. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 수 다음을 모니터링하는 데 사용: • 작업 진행 상황. • 다른 지표와 상관관계가 계산되는 경우, 작업 실행의 단계별 타임라인. 이 데이터는 다음과 같이 사용할 수 있습니다. • 작업 실행에서 가장 까다로운 단계를 식별합니다. • 여러 작업 실행에서 상관관계가 있는 스파이크(까다로운 단계)에 대해 경보를 설정합니다.

지표	설명
glue.driver.aggregate.numCompletedTasks	작업에서 상관관계가 있는 태스크 수. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 수 다음을 모니터링하는 데 사용: • 작업 진행 상황. • 특정 단계 내 병렬화.
glue.driver.aggregate.numFailedTasks	실패한 태스크 수. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 수 다음을 모니터링하는 데 사용: • 작업 태스크가 실패한 원인인 데이터 이상 현상. • 작업 태스크가 실패한 원인인 클러스터 이상 현상. • 작업 태스크가 실패한 원인인 스크립트 이상 현상. 이 데이터를 사용하여 데이터, 클러스터 또는 스크립트 이상을 암시하는 실패 증가에 대해 경보를 설정할 수 있습니다.

지표	설명
glue.driver.aggregate.numKilledTasks	중지된 태스크 수. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 수 다음을 모니터링하는 데 사용: • 태스크가 중지된 예외(OOM)가 발생한 데이터 스큐 이상 현상. • 태스크가 중지된 예외(OOM)가 발생한 스크립트 이상 현상. 이 데이터는 다음과 같이 사용할 수 있습니다. • 데이터 이상 현상을 나타내는 실패 증가에 대한 경보를 설정합니다. • 클러스터 이상 현상을 나타내는 실패 증가에 대한 경보를 설정합니다. • 스크립트 이상 현상을 나타내는 실패 증가에 대한 경보를 설정합니다.

지표	설명
glue.driver.aggregate.recordsRead	모든 실행기에서 실행되는 모든 완료된 Spark 작업이 모든 데이터 원본에서 읽은 레코드 수. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 수 다음을 모니터링하는 데 사용: • 읽은 레코드 수. • 작업 진행 상황. • JDBC 데이터 원본. • 작업 북마크 문제. • 며칠에 걸친 작업 실행의 스큐. 이 지표는 glue.ALL.s3.filesystem.read_bytes 지표와 동일한 방식으로 사용할 수 있으며, 차이는 이 지표가 Spark 작업 종료 시 업데이트된다는 것입니다.

지표	설명
glue.driver.aggregate.shuffleBytesWritten	이전 보고 이후 데이터 셔플을 위해 모든 실행기가 기록한 바이트 수(AWS Glue 지표 대시보드가 이 목적으로 이전 1분 동안 기록된 바이트 수로 집계). 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 바이트 다음을 모니터링하는 데 사용: 작업(대규모 joins, groupBy, repartition, coalesce) 내 데이터 셔플. 이 데이터는 다음과 같이 사용할 수 있습니다. • 대규모 입력 파일을 추가 처리 전에 재파티셔닝 또는 압축 해제합니다. • 데이터를 보다 균일하게 다시 파티셔닝하여 핫 키를 방지합니다. • joins 또는 groupBy 작업 전에 데이터를 사전 필터링합니다.

지표	설명
glue.driver.aggregate.shuffleLocalBytesRead	이전 보고 이후 데이터 셔플을 위해 모든 실행기가 읽은 바이트 수(AWS Glue 지표 대시보드가 이 목적으로 이전 1분 동안 읽은 바이트 수로 집계). 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. 단위: 바이트 다음을 모니터링하는 데 사용: 작업(대규모 joins, groupBy, repartition, coalesce) 내 데이터 셔플. 이 데이터는 다음과 같이 사용할 수 있습니다. • 대규모 입력 파일을 추가 처리 전에 재파티셔닝 또는 압축 해제합니다. • 핫 키를 사용하여 데이터를 보다 균일하게 다시 파티셔닝합니다. • joins 또는 groupBy 작업 전에 데이터를 사전 필터링합니다.

지표	설명
`glue.driver.BlockManager.disk.diskSpaceUsed_MB`	모든 실행기에서 사용된 디스크 공간의 메가바이트 수. 유효 차원: `JobName`(AWS Glue 작업 이름), `JobRunId`(JobRun ID. 또는 `ALL`), `Type`(게이지). 유효 통계: Average. 이 Spark 지표는 절대값으로 보고됩니다. 단위: 메가바이트 다음을 모니터링하는 데 사용: • 캐싱된 RDD 파티션을 나타내는 블록에 사용된 디스크 공간. • 중간 셔플 출력을 나타내는 블록에 사용된 디스크 공간. • 브로드캐스트를 나타내는 블록에 사용된 디스크 공간. 이 데이터는 다음과 같이 사용할 수 있습니다. • 디스크 사용량 증가로 인한 작업 실패를 식별합니다. • 분산 또는 셔플이 발생하는 대규모 파티션을 식별합니다. • 이 문제를 해결하려면 프로비저닝된 DPU 용량을 증가시킵니다.

지표	설명
glue.driver.ExecutorAllocationManager.executors.numberAllExecutors	능동적으로 실행 중인 실행기 수. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(게이지). 유효 통계: Average. 이 Spark 지표는 절대값으로 보고됩니다. 단위: 수 다음을 모니터링하는 데 사용: • 작업 활동. • 스트래글링 실행기(몇 개의 실행기만 실행 중) • 현재의 실행기 수준 병렬화. 이 데이터는 다음과 같이 사용할 수 있습니다. • 클러스터 활용도가 저하되기 전에 대규모 입력 파일을 재파티셔닝 또는 압축 해제합니다. • 스트래글러 시나리오로 인한 단계 또는 작업 예외 지연을 식별합니다. • • 추가 DPU를 프로비저닝해야 하는 백로그를 이해하기 위해 numberMaxNeededExecutors와 비교합니다.

지표	설명
glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors	현재 로드를 충족하는 데 필요한 최대 작업 (능동 실행 및 보류) 실행기 수. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(게이지). 유효한 통계: Maximum. 이 Spark 지표는 절대값으로 보고됩니다. 단위: 수 다음을 모니터링하는 데 사용: • 작업 활동. • DPU 용량 또는 중지/실패한 실행기로 인해 가용 실행기가 없어 아직 예약되지 않은 보류 태스크의 현재 실행기 수준 병렬화 및 백로그. 이 데이터는 다음과 같이 사용할 수 있습니다. • 예약 대기열의 보류/백로그를 식별합니다. • 스트래글러 시나리오로 인한 단계 또는 작업 예외 지연을 식별합니다. • 추가 DPU를 프로비저닝해야 하는 백로그를 이해하기 위해 numberAllExecutors와 비교합니다. • 보류 실행기 백로그를 교정하려면 프로비저닝된 DPU 용량을 증가시킵니다.

지표	설명
glue.driver.jvm.heap.usage glue.executorId.jvm.heap.usage glue.ALL.jvm.heap.usage	드라이버, executorId로 식별되는 실행기 또는 모든 실행기에 대해 이 드라이버용 JVM 힙이 사용하는 메모리 부분(규모: 0-1). 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(게이지). 유효 통계: Average. 이 Spark 지표는 절대값으로 보고됩니다. 단위: 퍼센트 다음을 모니터링하는 데 사용: • glue.driver.jvm.heap.usage 를 사용하여 드라이버 메모리 부족 상태(OOM). • glue.ALL.jvm.heap.usage 를 사용하여 실행기 메모리 부족 상태(OOM). 이 데이터는 다음과 같이 사용할 수 있습니다. • 메모리 사용 실행기 ID 및 단계를 식별합니다. • 스트래글링 실행기 ID 및 단계를 식별합니다. • 드라이버 메모리 부족(OOM) 상태를 식별합니다. • 실행기 메모리 부족(OOM) 상태를 식별하고 실행기 로그에서 스택 트레이스를 가져올 수 있도록 해당 실행기 ID를 확인합니다. • 스트래글러 또는 메모리 부족 상태(OOM)가 발생하는 데이터 스큐를 가질 수 있는 파일 또는 파티션을 식별합니다.

지표	설명
glue.driver.jvm.heap.used	드라이버, executorId로 식별되는 실행기 또는 모든 실행기에 대해 JVM 힙이 사용하는 메모리 바이트 수.
glue.executorId.jvm.heap.used	유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(게이지).
glue.ALL.jvm.heap.used	유효 통계: Average. 이 Spark 지표는 절대값으로 보고됩니다. 단위: 바이트 다음을 모니터링하는 데 사용: • 드라이버 메모리 부족 상태(OOM). • 실행기 메모리 부족 상태(OOM). 이 데이터는 다음과 같이 사용할 수 있습니다. • 메모리 사용 실행기 ID 및 단계를 식별합니다. • 스트래글링 실행기 ID 및 단계를 식별합니다. • 드라이버 메모리 부족(OOM) 상태를 식별합니다. • 실행기 메모리 부족(OOM) 상태를 식별하고 실행기 로그에서 스택 트레이스를 가져올 수 있도록 해당 실행기 ID를 확인합니다. • 스트래글러 또는 메모리 부족 상태(OOM)가 발생하는 데이터 스큐를 가질 수 있는 파일 또는 파티션을 식별합니다.

지표	설명

`glue.driver.s3.filesystem.read_bytes`
 	
`glue.executorId.s3.filesystem.read_bytes`
 	
`glue.ALL.s3.filesystem.read_bytes`
	이전 보고 이후 드라이버, executorId로 식별되는 실행기 또는 모든 실행기가 Amazon S3에서 읽은 바이트 수(AWS Glue 지표 대시보드가 이 목적으로 이전 1분 동안 읽은 바이트 수로 집계).
	
	유효 차원: `JobName`, `JobRunId`, `Type`(게이지).
	
	유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. AWS Glue 지표 대시보드에서 곡선 아래 영역은 두 작업 실행이 읽은 바이트 수를 시각적으로 비교하는 데 사용할 수 있습니다.
	
	단위: 바이트
	
	다음을 모니터링하는 데 사용:
	
	• ETL 데이터 이동.
	• 작업 진행 상황.
	• 작업 북마크 문제(처리, 재처리 및 건너뛴 데이터).
	• 외부 데이터 원본으로부터 수집 속도와 읽기를 비교.
	• 작업 실행 간 편차.
	
	결과 데이터는 다음 용도로 사용할 수 있습니다.
	
	• DPU 용량 계획.
	• 작업 실행 및 작업 단계를 위해 읽은 데이터의 대규모 스파이크 또는 딥에 대한 경보를 설정.

지표	설명
glue.driver.s3.filesystem.write_bytes glue.executorId.s3.filesystem.write_bytes glue.ALL.s3.filesystem.write_bytes	이전 보고 이후 드라이버, executorId로 식별되는 실행기 또는 모든 실행기가 Amazon S3에 쓴 바이트 수(AWS Glue 지표 대시보드가 이 목적으로 이전 1분 동안 기록된 바이트 수로 집계). 유효 차원: JobName, JobRunId, Type(게이지). 유효 통계: SUM. 이 지표는 마지막으로 보고된 값의 델타 값입니다. 그러므로 AWS Glue 지표 대시보드에서 SUM 통계가 집계에 사용됩니다. AWS Glue 지표 대시보드에서 곡선 아래 영역은 두 작업 실행이 기록한 바이트 수를 시각적으로 비교하는 데 사용할 수 있습니다. 단위: 바이트 다음을 모니터링하는 데 사용: • ETL 데이터 이동. • 작업 진행 상황. • 작업 북마크 문제(처리, 재처리 및 건너뛴 데이터). • 외부 데이터 원본으로부터 수집 속도와 읽기를 비교. • 작업 실행 간 편차. 이 데이터는 다음과 같이 사용할 수 있습니다. • DPU 용량 계획. • 작업 실행 및 작업 단계를 위해 읽은 데이터의 대규모 스파이크 또는 딥에 대한 경보를 설정.

지표	설명
glue.driver.streaming.numRecords	마이크로 배치로 수신된 레코드 수입니다. 이 지표는 AWS Glue 버전 2.0 이상의 AWS Glue 스트리밍 작업에만 사용할 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: Sum, Maximum, Minimum, Average, Percentile 단위: 수 다음을 모니터링하는 데 사용: • 읽은 레코드 수. • 작업 진행 상황.
glue.driver.streaming.batchProcessingTimeInMs	배치를 처리하는 데 걸리는 시간(밀리초)입니다. 이 지표는 AWS Glue 버전 2.0 이상의 AWS Glue 스트리밍 작업에만 사용할 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID. 또는 ALL), Type(개수). 유효 통계: Sum, Maximum, Minimum, Average, Percentile 단위: 수 다음을 모니터링하는 데 사용: • 작업 진행 상황. • 스크립트 성능입니다.

지표	설명

`glue.driver.system.cpuSystemLoad`
 	
`glue.executorId.system.cpuSystemLoad`
 	
`glue.ALL.system.cpuSystemLoad`
	드라이버, executorId로 식별되는 실행기 또는 모든 실행기가 사용한 CPU 시스템 부하 부분(규모: 0-1).
	
	유효 차원: `JobName`(AWS Glue 작업 이름), `JobRunId`(JobRun ID. 또는 `ALL`), `Type`(게이지).
	
	유효 통계: Average. 이 지표는 절대값으로 보고됩니다.
	
	단위: 퍼센트
	
	다음을 모니터링하는 데 사용:
	
	• 드라이버 CPU 부하.
	• 실행기 CPU 로드.
	• 작업에서 CPU 바운드 또는 IO 바운드 실행기를 탐지.
	
	이 데이터는 다음과 같이 사용할 수 있습니다.
	
	• IO 지표(읽은 바이트/셔플 바이트, 태스크 병렬화) 및 최대 필요 실행기 지표와 함께 DPU 용량 계획.
	• CPU/IO 바운드 비율을 식별합니다. 이를 통해 CPU 사용률이 저조한, 분리 가능 데이터 세트의 장기 실행 작업을 위해 프로비저닝된 용량을 재파티셔닝 및 증가할 수 있습니다.

AWS Glue 지표의 차원

AWS Glue 지표는 AWS Glue 네임스페이스를 사용하며 다음 차원의 지표를 제공합니다.

측정기준	설명
`JobName`	이 차원은 특정 AWS Glue 작업 내 모든 작업 실행의 지표를 필터링합니다.

측정기준	설명
JobRunId	이 차원은 JobRun ID에 의해 실행되는 특정 AWS Glue 작업 또는 ALL의 지표를 필터링합니다.
Type	이 차원은 count(집계 값) 또는 gauge(특정 시점 값) 기준 지표를 필터링합니다.

자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

AWS Glue 작업 프로파일에서 Amazon CloudWatch 경보 설정

AWS Glue 지표는 Amazon CloudWatch에서도 사용할 수 있습니다. 예약된 작업의 AWS Glue 측정치에 대해 경보를 설정할 수 있습니다.

경보 설정을 위한 몇 가지 일반적인 시나리오는 다음과 같습니다.

- 메모리 부족(OOM) 발생 작업: 메모리 사용량이 AWS Glue 작업에 대한 드라이버 또는 실행기에 대해 정상 평균을 초과할 때 경보를 설정합니다.
- 스트래글링 실행기: AWS Glue 작업에서 실행기 수가 오랜 시간 동안 특정 임곗값보다 낮을 때 경보를 설정합니다.
- 데이터 백로그 또는 재처리: CloudWatch 수학 표현식을 사용하여 워크플로 내 개별 작업의 지표를 비교합니다. 결과 표현식 값(작업이 쓴 바이트 수와 다음 작업이 읽은 바이트 수의 비율 등)에 대해 경보를 트리거할 수 있습니다.

경보 설정에 대한 자세한 지침은 Amazon CloudWatch Events User Guide의 Create or Edit a CloudWatch Alarm을 참조하세요.

CloudWatch를 사용한 모니터링 및 디버깅 시나리오는 작업 모니터링 및 디버깅 섹션을 참조하세요.

AWS Glue 작업의 로깅

AWS Glue 5.0에서는 모든 작업에 실시간 로깅 기능이 제공됩니다. 또한 사용자 지정 구성 옵션을 지정하여 로깅 동작을 조정할 수 있습니다. 이러한 옵션으로는 Amazon CloudWatch 로그 그룹 이름, Amazon CloudWatch 로그 스트림 접두사(AWS Glue 작업 실행 ID 및 드라이버/실행기 ID 앞에 있음), 로그 메시지의 로그 변환 패턴 설정이 있습니다. 이러한 구성을 사용하면 만료 정책이 서로 다른 사용자 지정 Amazon CloudWatch 로그 그룹의 로그를 집계할 수 있습니다. 여기에 더해 사용자 지정 로그

스트림 접두사와 변환 패턴을 사용하여 로그를 보다 효과적으로 분석할 수 있습니다. 이러한 수준의 사용자 지정을 통해 특정 요구 사항에 따라 로그 관리 및 분석을 최적화할 수 있습니다.

AWS Glue 5.0의 로깅 동작

기본적으로 시스템 로그, Spark 대몬 로그 및 사용자 AWS Glue Logger 로그는 Amazon CloudWatch의 `/aws-glue/jobs/error` 로그 그룹에 기록됩니다. 반면 사용자 stdout(표준 출력) 및 stderr(표준 오류) 로그는 기본적으로 `/aws-glue/jobs/output` 로그 그룹에 기록됩니다.

사용자 지정 로깅

다음 작업 인수를 사용하여 기본 로그 그룹 및 로그 스트림 접두사를 사용자 지정할 수 있습니다.

- `--custom-logGroup-prefix`: `/aws-glue/jobs/error` 및 `/aws-glue/jobs/output` 로그 그룹에 대한 사용자 지정 접두사를 지정할 수 있습니다. 사용자 지정 접두사를 입력하는 경우 로그 그룹 이름의 형식은 다음과 같습니다.
  - `/aws-glue/jobs/error`은 *<customer prefix>*`/error`이 됩니다.
  - `/aws-glue/jobs/output` 은 *<customer prefix>*`/output`이 됩니다.
- `--custom-logStream-prefix`: 로그 그룹 내에 있는 로그 스트림 이름의 사용자 지정 접두사를 지정할 수 있습니다. 사용자 지정 접두사를 입력하는 경우 로그 스트림 이름의 형식은 다음과 같습니다.
  - `jobrunid-driver`은 *<customer log stream>*`-driver`이 됩니다.
  - `jobrunid-executorNum`은 *<customer log stream>*`-executorNum`이 됩니다.

사용자 지정 접두사의 검증 규칙 및 제한 사항:

- 전체 로그 스트림 이름은 1~512자여야 합니다.
- 사용자 지정 접두사 자체는 400자로 제한됩니다.
- 사용자 지정 접두사는 정규식 패턴 `[^:*]*`(허용되는 특수 문자는 '_', '-' 및 '/')과 일치해야 합니다.

사용자 지정 스크립트 로거를 사용하여 애플리케이션별 메시지 로깅

AWS Glue 로거를 사용하면 드라이버 로그 스트림에 실시간으로 전송되는 모든 애플리케이션별 메시지를 스크립트에서 로깅할 수 있습니다.

다음 예제에서는 Python 스크립트를 보여줍니다.

```
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")
```

다음 예제에서는 Scala 스크립트를 보여줍니다.

```
import com.amazonaws.services.glue.log.GlueLogger

object GlueApp {
  def main(sysArgs: Array[String]) {
    val logger = new GlueLogger
    logger.info("info message")
    logger.warn("warn message")
    logger.error("error message")
  }
}
```

작업 진행률을 보여주는 진행률 표시줄 사용 설정

AWS Glue는 AWS Glue 작업 실행 상태를 확인하기 위해 `JOB_RUN_ID-progress-bar` 로그 스트림 아래에 실시간 진행률 표시줄을 제공합니다. 현재 `glueContext`를 초기화하는 작업만 지원합니다. `glueContext`를 초기화하지 않고 기본 Spark 작업을 실행하는 경우에는 AWS Glue 진행률 표시줄이 나타나지 않습니다.

이 진행률 표시줄에서는 5초마다 다음 진행률 업데이트를 표시합니다.

```
Stage Number (Stage Name): > (numCompletedTasks + numActiveTasks) /
 totalNumOfTasksInThisStage]
```

Amazon CloudWatch 로깅을 사용한 보안 구성

Amazon CloudWatch 로그에 대한 보안 구성이 활성화되면 AWS Glue는 보안 구성 이름을 포함하는 특정 명명 패턴이 있는 로그 그룹을 생성합니다.

보안 구성을 사용한 로그 그룹 이름 지정

기본 및 사용자 정의 로그 그룹은 다음과 같습니다.

- 기본 오류 로그 그룹: /aws-glue/jobs/*Security-Configuration-Name*-role/*glue-job-role*/error
- 기본 출력 로그 그룹: /aws-glue/jobs/*Security-Configuration-Name*-role/*glue-job-role*/output
- 사용자 지정 오류 로그 그룹(AWS Glue 5.0): *custom-log-group-prefix*/*Security-Configuration-Name*-role/*glue-job-role*/error
- 사용자 지정 출력 로그 그룹(AWS Glue 5.0): *custom-log-group-prefix*/*Security-Configuration-Name*-role/*glue-job-role*/output

필수 IAM 권한

Amazon CloudWatch Logs로 보안 구성을 사용하는 경우 IAM 역할 권한에 logs:AssociateKmsKey 권한을 추가해야 합니다. 해당 권한이 포함되지 않으면 연속 로깅이 사용 중지됩니다.

또한 Amazon CloudWatch Logs에 대한 암호화를 구성하려면 Amazon Amazon CloudWatch Logs 사용 설명서의 AWS Key Management Service를 사용하여 Amazon CloudWatch 로그에서 로그 데이터 암호화를 참조하세요.

추가 정보

보안 구성 생성에 대한 자세한 내용은 AWS Glue 콘솔에서 보안 구성 관리를 참조하세요.

주제

- AWS Glue 4.0 이하 작업에 대한 지속 로깅 활성화
- AWS Glue 작업 로그 보기

AWS Glue 4.0 이하 작업에 대한 지속 로깅 활성화

> **Note**
>
> AWS Glue 4.0 이하 버전에서는 지속 로깅이 사용 가능한 기능이었습니다. 하지만 AWS Glue 5.0이 도입됨에 따라 모든 작업에는 실시간 로깅 기능이 제공됩니다. AWS Glue 5.0의 로깅 기능 및 구성 옵션에 대한 자세한 내용은 AWS Glue 작업의 로깅을 참조하세요.

AWS Glue 콘솔을 사용하거나 AWS Command Line Interface(AWS CLI)를 통해 지속 로깅을 활성화할 수 있습니다.

새 작업을 만들거나 기존 작업을 편집할 때 또는 AWS CLI를 통해 연속 로깅을 사용 설정할 수 있습니다.

Amazon CloudWatch 로그 그룹 이름, AWS Glue 작업 실행 ID 드라이버/실행기 ID 앞의 CloudWatch 로그 스트림 접두사 및 로그 메시지에 대한 로그 변환 패턴 등의 사용자 정의 구성 옵션을 지정할 수도 있습니다. 이러한 구성을 사용하면 만료 정책이 서로 다른 사용자 지정 CloudWatch 로그 그룹에 집계 로그를 설정하고 사용자 지정 로그 스트림 접두사 및 변환 패턴을 사용하여 더 자세히 분석할 수 있습니다.

주제

- AWS Management Console 사용
- 사용자 지정 스크립트 로거를 사용하여 애플리케이션별 메시지 로깅
- 작업 진행률을 보여주는 진행률 표시줄 사용 설정
- 지속 로깅을 사용한 보안 구성

AWS Management Console 사용

AWS Glue 작업을 생성하거나 편집할 때 콘솔에서 지속 로깅을 활성화하려면 다음 절차를 수행합니다.

지속 로깅으로 새 AWS Glue 작업을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 ETL 작업을 선택합니다.
3. 시각적 ETL을 선택합니다.
4. 작업 세부 정보 탭에서 고급 속성 섹션을 확장합니다.
5. 연속 로깅에서 CloudWatch에서 로그 활성화를 선택합니다.

기존 AWS Glue 작업에 대한 지속 로깅을 활성화하려면

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서, 작업을 선택합니다.
3. 작업 목록에서 기존 작업을 선택합니다.
4. 작업, 작업 편집을 선택합니다.
5. 작업 세부 정보 탭에서 고급 속성 섹션을 확장합니다.

6. 연속 로깅에서 CloudWatch에서 로그 활성화를 선택합니다.

AWS CLI 사용

지속 로깅을 활성화하려면 작업 파라미터를 AWS Glue 작업에 전달합니다. 다른 AWS Glue 작업 파라미터와 유사한 다음 특수 작업 파라미터를 전달합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용 섹션을 참조하세요.

```
'--enable-continuous-cloudwatch-log': 'true'
```

사용자 정의 Amazon CloudWatch 로그 그룹 이름을 지정할 수 있습니다. 지정하지 않으면 기본 로그 그룹 이름은 `/aws-glue/jobs/logs-v2`입니다.

```
'--continuous-log-logGroup': 'custom_log_group_name'
```

사용자 정의 Amazon CloudWatch 로그 스트림 접두사를 지정할 수 있습니다. 지정하지 않으면 기본 로그 스트림 접두사는 작업 실행 ID입니다.

```
'--continuous-log-logStreamPrefix': 'custom_log_stream_prefix'
```

사용자 지정 지속적 로깅 변환 규칙을 지정할 수 있습니다. 지정하지 않으면 기본 변환 패턴은 `%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n`입니다. 변환 패턴은 드라이버 로그 및 실행기 로그에만 적용되며 AWS Glue 진행률 표시줄에는 영향을 주지 않습니다.

```
'--continuous-log-conversionPattern': 'custom_log_conversion_pattern'
```

사용자 지정 스크립트 로거를 사용하여 애플리케이션별 메시지 로깅

AWS Glue 로거를 사용하면 드라이버 로그 스트림에 실시간으로 전송되는 모든 애플리케이션별 메시지를 스크립트에서 로깅할 수 있습니다.

다음 예제에서는 Python 스크립트를 보여줍니다.

```
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")
```

다음 예제에서는 Scala 스크립트를 보여줍니다.

```
import com.amazonaws.services.glue.log.GlueLogger

object GlueApp {
  def main(sysArgs: Array[String]) {
    val logger = new GlueLogger
    logger.info("info message")
    logger.warn("warn message")
    logger.error("error message")
  }
}
```

작업 진행률을 보여주는 진행률 표시줄 사용 설정

AWS Glue는 AWS Glue 작업 실행 상태를 확인하기 위해 JOB_RUN_ID-progress-bar 로그 스트림 아래에 실시간 진행률 표시줄을 제공합니다. 현재 glueContext를 초기화하는 작업만 지원합니다. glueContext를 초기화하지 않고 기본 Spark 작업을 실행하는 경우에는 AWS Glue 진행률 표시줄이 나타나지 않습니다.

이 진행률 표시줄에서는 5초마다 다음 진행률 업데이트를 표시합니다.

```
Stage Number (Stage Name): > (numCompletedTasks + numActiveTasks) /
 totalNumOfTasksInThisStage]
```

지속 로깅을 사용한 보안 구성

CloudWatch 로그에 보안 구성이 사용되는 경우 AWS Glue는 연속 로그에 대해 다음과 같은 이름의 로그 그룹을 생성합니다.

```
<Log-Group-Name>-<Security-Configuration-Name>
```

기본 및 사용자 정의 로그 그룹은 다음과 같습니다.

- 기본 연속 로그 그룹은 /aws-glue/jobs/error-<*Security-Configuration-Name*>입니다.

- 사용자 정의 연속 로그 그룹은 <*custom-log-group-name*>-<*Security-Configuration-Name*>입니다.

CloudWatch Logs로 보안 구성을 사용하는 경우 IAM 역할 권한에 `logs:AssociateKmsKey`를 추가해야 합니다. 해당 권한이 포함되지 않으면 연속 로깅이 사용 중지됩니다. 또한 CloudWatch Logs에 대한 암호화를 구성하려면 Amazon CloudWatch Logs User Guide의 Encrypt Log Data in CloudWatch Logs Using AWS Key Management Service를 참조하세요.

보안 구성 생성에 대한 자세한 내용은 AWS Glue 콘솔에서 보안 구성 관리 섹션을 참조하세요.

> **Note**
>
> 로깅을 활성화하고 추가 CloudWatch 로그 이벤트가 생성되면 추가 요금이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

AWS Glue 작업 로그 보기

AWS Glue 콘솔 또는 Amazon CloudWatch 콘솔을 사용하여 실시간 로그를 볼 수 있습니다.

AWS Glue 콘솔 대시보드를 사용하여 실시간 로그를 보려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서, 작업을 선택합니다.
3. 기존 작업을 추가하거나 시작합니다. 작업, 작업 실행을 선택합니다.

   작업 실행을 시작할 때 실행 중인 작업에 대한 정보가 포함된 페이지로 이동합니다.

   - 로그 탭에 이전에 집계된 애플리케이션 로그가 표시됩니다.
   - 로그 탭에는 `glueContext`가 초기화된 작업이 실행 중일 때 실시간 진행률 표시줄이 표시됩니다.
   - 로그 탭에는 실시간 Apache Spark 드라이버 로그를 캡처하는 드라이버 로그 및 작업이 실행 중일 때 AWS Glue 애플리케이션 로거를 사용하여 로깅한 스크립트의 애플리케이션 로그도 포함됩니다.
4. 이전 작업의 경우 [로그(Logs)]를 선택하여 [작업 기록(Job History)] 보기에서 실시간 로그를 볼 수도 있습니다. 이러한 작업을 수행하면 해당 작업 실행에 대한 모든 Spark 드라이버, 실행기 및 진행률 표시줄 로그 스트림을 표시하는 CloudWatch 콘솔로 이동합니다.

CloudWatch 콘솔 대시보드를 사용하여 실시간 로그를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택합니다.
3. /aws-glue/jobs/error/ 로그 그룹을 선택합니다.
4. 필터 상자에 작업 실행 ID를 붙여넣습니다.

   드라이버 로그, 실행기 로그 및 진행률 표시줄을 볼 수 있습니다(Standard filter(표준 필터)를 사용하는 경우).

AWS Glue 관찰성 메트릭을 사용한 모니터링

> **Note**
>
> AWS Glue 관찰성 지표는 AWS Glue 4.0 이상 버전에서 사용할 수 있습니다.

AWS Glue 관찰성 메트릭을 사용하면 Apache Spark의 AWS Glue 내부에서 일어나는 일에 대한 통찰력을 얻어 문제의 분류 및 분석을 개선할 수 있습니다. 관찰성 지표는 Amazon CloudWatch 대시보드를 통해 시각화되며 오류의 근본 원인 분석을 수행하고 성능 병목 현상을 진단하는 데 사용할 수 있습니다. 대규모 문제 디버깅에 소요되는 시간을 줄여 문제를 더 빠르고 효과적으로 해결하는 데 집중할 수 있습니다.

AWS Glue 관찰성은 다음 네 그룹으로 분류된 Amazon CloudWatch 메트릭을 제공합니다.

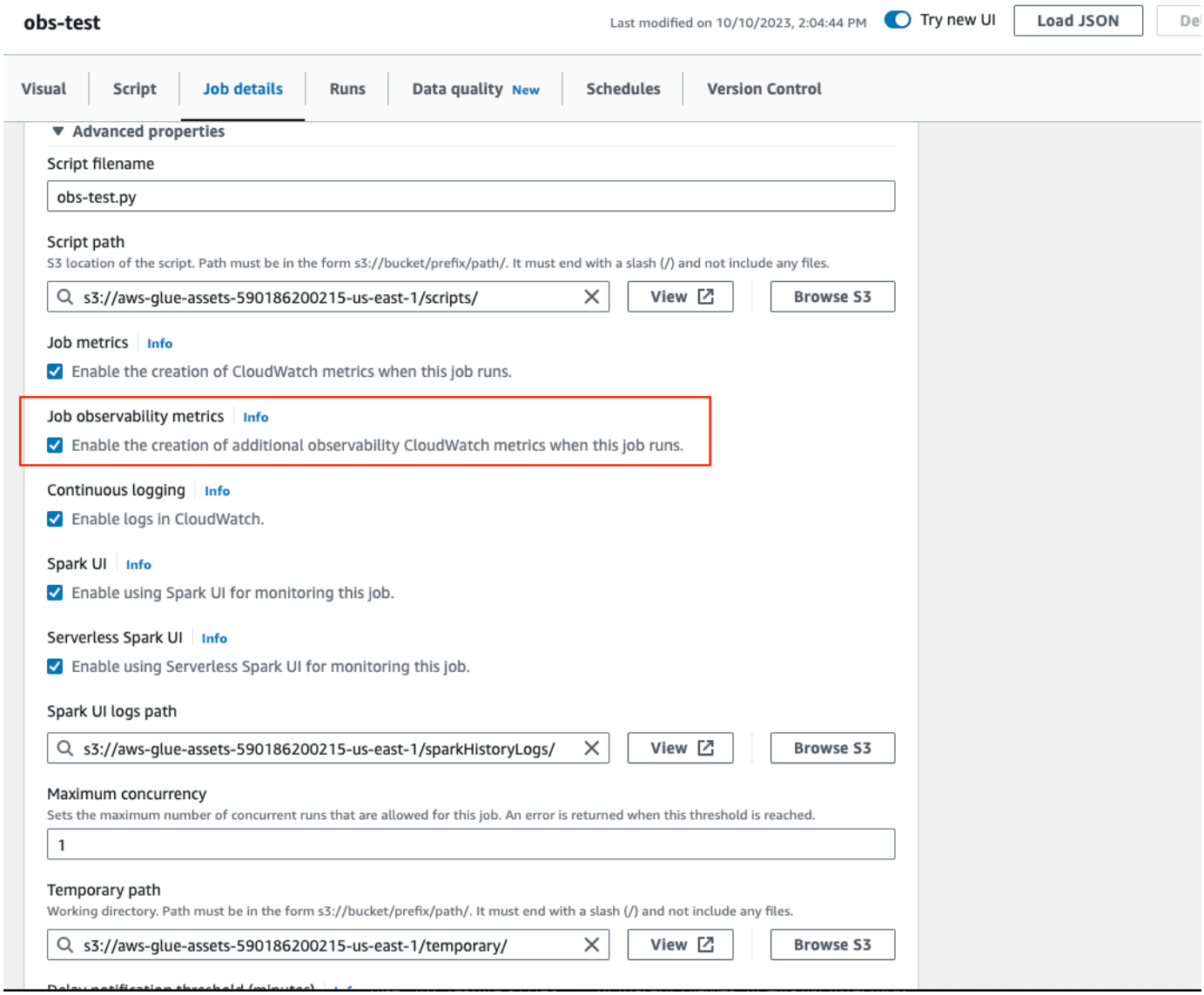
- 신뢰성(예: 오류 클래스) - 주어진 시간 범위에서 해결해야 할 가장 일반적인 장애 원인을 쉽게 식별할 수 있습니다.
- 성능(예: Skewness) - 성능 병목 지점을 식별하고 조정 기법을 적용합니다. 예를 들어 작업 왜곡으로 인해 성능이 저하되는 경우 Spark Adaptive Query Execution을 활성화하고 스큐 조인 임곗값을 미세 조정하는 것이 좋습니다.
- 처리량(예: 소스/싱크당 처리량) - 데이터 읽기 및 쓰기 추세를 모니터링합니다. 또한 이상 현상에 대한 Amazon CloudWatch 경보를 구성할 수 있습니다.
- 리소스 사용률(예: 작업자, 메모리 및 디스크 사용률) - 용량 사용률이 낮은 작업을 효율적으로 찾을 수 있습니다. 이러한 작업에 대해 AWS Glue Auto Scaling을 활성화할 수 있습니다.

AWS Glue 관찰성 메트릭 시작하기

> **Note**
>
> 새로운 지표는 AWS Glue Studio 콘솔에서 기본적으로 활성화됩니다.

AWS Glue Studio에서 관찰성 지표를 구성하려면 다음을 수행합니다.

1. AWS Glue 콘솔에 로그인하고 콘솔 메뉴에서 ETL 작업을 선택합니다.
2. 내 작업 섹션에서 작업 이름을 클릭하여 작업을 선택합니다.
3. [작업 세부 정보(Job details)] 탭을 선택합니다.
4. 하단으로 스크롤하여 고급 속성을 선택한 다음 작업 관찰성 지표를 선택합니다.



AWS CLI를 사용하여 AWS Glue 관찰성 지표를 활성화하려면 다음을 수행합니다.

- 입력 `--default-arguments` JSON 파일의 다음 키-값을 맵에 추가합니다.

```
--enable-observability-metrics, true
```

AWS Glue 관찰성 사용

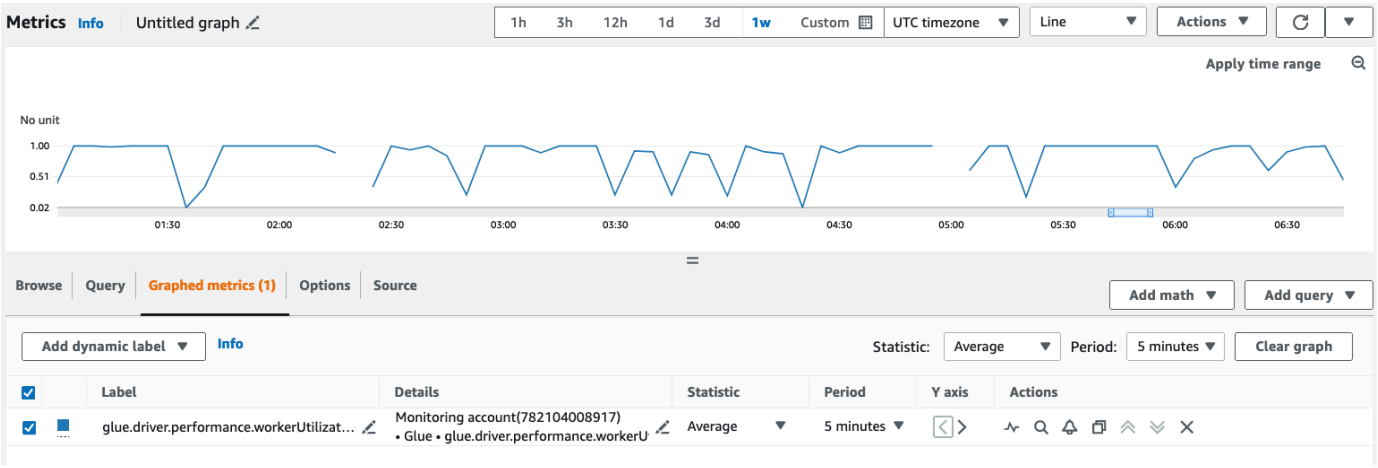
AWS Glue 관찰성 지표는 Amazon CloudWatch를 통해 제공되므로 Amazon CloudWatch 콘솔, AWS CLI, SDK 또는 API를 사용하여 관찰성 지표 데이터 포인트를 쿼리할 수 있습니다. AWS Glue 관찰

성 지표를 사용하는 사용 사례의 예는 Using Glue Observability for monitoring resource utilization to reduce cost를 참조하세요.

Amazon CloudWatch 콘솔에서 AWS Glue 관찰성 사용

Amazon CloudWatch 콘솔에서 메트릭을 쿼리하고 시각화는 방법:

1. Amazon CloudWatch 콘솔을 열고 모든 지표를 선택합니다.
2. 사용자 지정 네임스페이스에서 AWS Glue를 선택합니다.
3. 작업 관찰성 지표, 소스별 관찰 가능성 지표 또는 싱크당 관찰 가능성 지표를 선택합니다.
4. 특정 지표 이름, 작업 이름, 작업 실행 ID를 검색하고 선택합니다.
5. 그래프로 표시된 지표 탭에서 원하는 통계, 기간 및 기타 옵션을 구성합니다.



AWS CLI를 사용하여 관찰성 지표를 쿼리하려면 다음을 수행합니다.

1. 지표 정의 JSON 파일을 생성하고 `your-Glue-job-name` 및 `your-Glue-job-run-id`를 변경합니다.

```
$ cat multiplequeries.json
[
    {
        "Id": "avgWorkerUtil_0",
        "MetricStat": {
            "Metric": {
                "Namespace": "Glue",
                "MetricName": "glue.driver.workerUtilization",
                "Dimensions": [
                    {
```

```
                        "Name": "JobName",
                        "Value": "<your-Glue-job-name-A>"
                    },
                    {
                        "Name": "JobRunId",
                        "Value": "<your-Glue-job-run-id-A>"
                    },
                    {
                        "Name": "Type",
                        "Value": "gauge"
                    },
                    {
                        "Name": "ObservabilityGroup",
                        "Value": "resource_utilization"
                    }
                ]
            },
            "Period": 1800,
            "Stat": "Minimum",
            "Unit": "None"
        }
    },
    {
        "Id": "avgWorkerUtil_1",
        "MetricStat": {
            "Metric": {
                "Namespace": "Glue",
                "MetricName": "glue.driver.workerUtilization",
                "Dimensions": [
                    {
                        "Name": "JobName",
                        "Value": "<your-Glue-job-name-B>"
                    },
                    {
                        "Name": "JobRunId",
                        "Value": "<your-Glue-job-run-id-B>"
                    },
                    {
                        "Name": "Type",
                        "Value": "gauge"
                    },
                    {
                        "Name": "ObservabilityGroup",
                        "Value": "resource_utilization"
```

```
                    }
                ]
            },
            "Period": 1800,
            "Stat": "Minimum",
            "Unit": "None"
        }
    }
]
```

2. get-metric-data 명령을 실행합니다.

```
$ aws cloudwatch get-metric-data --metric-data-queries file: //multiplequeries.json
 \
     --start-time '2023-10-28T18: 20' \
     --end-time '2023-10-28T19: 10'  \
     --region us-east-1
{
    "MetricDataResults": [
        {
            "Id": "avgWorkerUtil_0",
            "Label": "<your-label-for-A>",
            "Timestamps": [
                "2023-10-28T18:20:00+00:00"
            ],
            "Values": [
                0.06718750000000001
            ],
            "StatusCode": "Complete"
        },
        {
            "Id": "avgWorkerUtil_1",
            "Label": "<your-label-for-B>",
            "Timestamps": [
                "2023-10-28T18:50:00+00:00"
            ],
            "Values": [
                0.5959183673469387
            ],
            "StatusCode": "Complete"
        }
    ],
```

```
    "Messages": []
}
```

관찰성 메트릭

AWS Glue 관찰성은 다음 지표의 프로필을 작성하고 30초마다 Amazon CloudWatch로 전송하며, 이러한 지표 중 일부는 AWS Glue Studio 작업 실행 모니터링 페이지에서 확인할 수 있습니다.

지표	설명	범주
glue.driver.skewness.stage	지표 범주: job_performance Spark 스테이지 실행 왜도: 이 지표는 입력 데이터 왜도 또는 변환(예: 왜곡된 조인)으로 인해 발생할 수 있는 실행 왜도를 캡처합니다. 이 지표의 값은 [0, 무한] 범위에 속합니다. 여기서 0은 스테이지의 모든 작업 중 최대 작업 실행 시간과 중간 작업 실행 시간의 비율이 특정 스테이지 왜도 인자보다 작다는 것을 의미합니다. 기본 스테이지 왜도 인자는 `5`이며 spark conf: spark.metrics.conf.driver.source.glue.jobPerformance.skewnessFactor를 통해 덮어쓸 수 있습니다. 스테이지 왜도 값이 1이면 비율이 단계 왜도 인자의 두 배임을 의미합니다. 스테이지 왜도 값은 현재 왜도를 반영하여 30초마다 업데이트됩니다. 스테이지가 끝날 때	job_performance

지표	설명	범주
	의 값은 최종 스테이지 왜도를 반영합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup (Job_Performance) 유효 통계: 평균, 최대, 최소, 백분위수 단위: 수	

지표	설명	범주
glue.driver.skewness.job	지표 범주: job_performance 작업 왜도는 작업 스테이지 왜도의 가중치 평균입니다. 가중치 평균은 실행하는 데 시간이 오래 걸리는 스테이지에 더 많은 가중치를 부여합니다. 이는 매우 왜곡된 스테이지가 실제로 다른 스테이지에 비해 매우 짧은 시간 동안 실행되는 경우를 피하기 위한 것입니다. 따라서 왜도는 전체 작업 성능에 있어 중요하지 않으며 왜도를 해결하려고 노력할 필요가 없습니다. 이 지표는 각 단계가 완료될 때마다 업데이트되므로 마지막 값은 실제 전체 작업 왜도를 반영합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup (Job_Performance) 유효 통계: 평균, 최대, 최소, 백분위수 단위: 수	job_performance

지표	설명	범주
glue.succeed.ALL	지표 범주: 오류 실패 범주의 그림을 완성하기 위한 총 작업 실행 성공 횟수 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(개수), ObservabilityGroup(오류) 유효 통계: SUM 단위: 수	오류
glue.error.ALL	지표 범주: 오류 실패 범주의 그림을 완성하기 위한 총 작업 실행 오류 횟수 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(개수), ObservabilityGroup(오류) 유효 통계: SUM 단위: 수	오류

지표	설명	범주
glue.error.[error category]	지표 범주: 오류 이것은 실제로 작업 실행이 실패할 때만 업데이트되는 지표 세트입니다. 오류 분류는 분류 및 디버깅에 도움이 됩니다. 작업 실행이 실패하면 실패의 원인이 되는 오류가 분류되고 해당 오류 범주 지표가 1로 설정됩니다. 이를 통해 시간 경과에 따른 오류 분석은 물론 모든 작업에 대한 오류 분석을 수행하여 가장 일반적인 오류 범주를 식별하고 문제 해결을 시작할 수 있습니다. AWS Glue에는 OUT_OF_MEMORY(드라이버 및 실행기), PERMISSION, SYNTAX 및 THROTTLING 오류 범주를 포함한 28개의 오류 범주가 있습니다. 오류 범주는 COMPILATION, LAUNCH, TIMEOUT 오류 범주도 포함합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(개수), ObservabilityGroup(오류) 유효 통계: SUM 단위: 수	오류

지표	설명	범주
glue.driver.workerUtilization	지표 범주: resource_utilization 할당된 작업자 중 실제로 사용된 작업자의 비율입니다. 상황이 좋지 않다면 Auto Scaling이 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) 유효 통계: 평균, 최대, 최소, 백분위수 단위: 퍼센트	resource_utilization

지표	설명	범주
glue.driver.memory.heap.[available | used]	지표 범주: resource_utilization 작업 실행 중 드라이버의 사용 가능/사용된 힙 메모리입니다. 이는 특히 시간 경과에 따른 메모리 사용량 추세를 파악하는 데 도움이 되며 메모리 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 바이트	resource_utilization

지표	설명	범주
glue.driver.memory.heap.used.percentage	지표 범주: resource_utilization 작업 실행 중 드라이버가 사용한 힙 메모리(%)입니다. 이는 특히 시간 경과에 따른 메모리 사용량 추세를 파악하는 데 도움이 되며 메모리 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization

지표	설명	범주
glue.driver.memory.non-heap.[available | used]	지표 범주: resource_utilization 작업 실행 중 드라이버가 사용 가능한 또는 사용한 힙이 아닌 메모리입니다. 이는 특히 시간 경과에 따른 메모리 사용량 추세를 파악하는 데 도움이 되며 메모리 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup (resource_utilization) Valid Statistics: Average 단위: 바이트	resource_utilization

지표	설명	범주
glue.driver.memory.non-heap.used.percentage	지표 범주: resource_utilization 작업 실행 중 드라이버가 사용한 힙이 아닌 메모리(%)입니다. 이는 특히 시간 경과에 따른 메모리 사용량 추세를 파악하는 데 도움이 되며 메모리 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization

지표	설명	범주
glue.driver.memory.total.[available | used]	지표 범주: resource_utilization 작업 실행 중 드라이버가 사용 가능한 또는 사용한 총 메모리입니다. 이는 특히 시간 경과에 따른 메모리 사용량 추세를 파악하는 데 도움이 되며 메모리 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup (resource_utilization) Valid Statistics: Average 단위: 바이트	resource_utilization

지표	설명	범주
glue.driver.memory.total.used.percentage	지표 범주: resource_utilization 작업 실행 중 드라이버가 사용한 총 메모리(%)입니다. 이는 특히 시간 경과에 따른 메모리 사용량 추세를 파악하는 데 도움이 되며 메모리 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization
glue.ALL.memory.heap.[available | used]	지표 범주: resource_utilization 실행기의 사용 가능한/사용된 힙 메모리입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 바이트	resource_utilization

지표	설명	범주
glue.ALL.memory.heap.used.percentage	지표 범주: resource_utilization 실행기가 사용한 힙 메모리(%) 입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup (resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization
glue.ALL.memory.non-heap.[available | used]	지표 범주: resource_utilization 실행기의 사용 가능한/사용된 힙이 아닌 메모리입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup (resource_utilization) Valid Statistics: Average 단위: 바이트	resource_utilization

지표	설명	범주
glue.ALL.memory.non-heap.used.percentage	지표 범주: resource_utilization 실행기가 사용한 힙이 아닌 메모리(%)입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization
glue.ALL.memory.total.[available | used]	지표 범주: resource_utilization 실행기의 사용 가능한/사용된 총 메모리입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 바이트	resource_utilization

지표	설명	범주
glue.ALL.memory.total.used.percentage	지표 범주: resource_utilization 실행기의 총 메모리 사용률(%)입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization
glue.driver.disk.[available_GB | used_GB]	지표 범주: resource_utilization 작업 실행 중 드라이버의 사용 가능한/사용된 디스크 공간입니다. 이는 특히 시간 경과에 따른 디스크 사용량 추세를 파악하는 데 도움이 되며 디스크 공간 부족 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 기가바이트	resource_utilization

지표	설명	범주
glue.driver.disk.used.percentage]	지표 범주: resource_utilization 작업 실행 중 드라이버의 사용 가능한/사용된 디스크 공간입니다. 이는 특히 시간 경과에 따른 디스크 사용량 추세를 파악하는 데 도움이 되며 디스크 공간 부족 관련 오류를 디버깅할 뿐만 아니라 잠재적인 오류를 방지하는 데도 도움이 될 수 있습니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization
glue.ALL.disk.[available_GB | used_GB]	지표 범주: resource_utilization 실행기의 사용 가능한/사용된 디스크 공간입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 기가바이트	resource_utilization

지표	설명	범주
glue.ALL.disk.used.percentage	지표 범주: resource_utilization 실행기의 사용 가능한/사용된/사용된(%) 디스크 공간입니다. ALL은 모든 실행기를 의미합니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization) Valid Statistics: Average 단위: 퍼센트	resource_utilization
glue.driver.bytesRead	지표 범주: 처리량 이 작업 실행에서 입력 소스당 및 모든 소스에 대해 읽은 바이트 수입니다. 이를 통해 데이터 양과 시간 경과에 따른 변화를 파악할 수 있으므로 데이터 왜도와 같은 문제를 해결하는 데 도움이 됩니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization), Source(소스 데이터 위치) Valid Statistics: Average 단위: 바이트	처리량

지표	설명	범주
glue.driver.[recordsRead | filesRead]	지표 범주: 처리량 이 작업 실행에서 입력 소스당 및 모든 소스에 대해 읽은 레코드/파일 수입니다. 이를 통해 데이터양과 시간 경과에 따른 변화를 파악할 수 있으므로 데이터 왜도와 같은 문제를 해결하는 데 도움이 됩니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization), Source(소스 데이터 위치) Valid Statistics: Average 단위: 수	처리량
glue.driver.partitionsRead	지표 범주: 처리량 이 작업 실행에서 Amazon S3 입력 소스당 및 모든 소스에 대해 읽은 파티션 수입니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization), Source(소스 데이터 위치) Valid Statistics: Average 단위: 수	처리량

지표	설명	범주
glue.driver.bytesWrittten	지표 범주: 처리량 이 작업 실행에서 출력 싱크당 및 모든 싱크에 대해 작성된 바이트 수입니다. 이를 통해 데이터양과 시간이 지남에 따라 어떻게 변화하는지 파악할 수 있으므로 처리 왜도와 같은 문제를 해결하는 데 도움이 됩니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup(resource_utilization), Sink(싱크 데이터 위치) Valid Statistics: Average 단위: 바이트	처리량

지표	설명	범주
glue.driver.[recordsWritten | filesWritten]	지표 범주: 처리량 이 작업 실행에서 출력 싱크당 및 모든 싱크에 대해 작성된 레코드/파일 수입니다. 이를 통해 데이터양과 시간이 지남에 따라 어떻게 변화하는지 파악할 수 있으므로 처리 왜도와 같은 문제를 해결하는 데 도움이 됩니다. 유효 차원: JobName(AWS Glue 작업 이름), JobRunId(JobRun ID 또는 ALL), Type(게이지), ObservabilityGroup (resource_utilization), Sink(싱크 데이터 위치) Valid Statistics: Average 단위: 수	처리량

오류 범주

오류 범주	설명
COMPILATION_ERROR	Scala 코드를 컴파일하는 동안 오류가 발생합니다.
CONNECTION_ERROR	서비스/원격 호스트/데이터베이스 서비스 등에 연결하는 동안 오류가 발생합니다.
DISK_NO_SPACE_ERROR	드라이버/실행기의 디스크에 공간이 남아 있지 않은 경우 오류가 발생합니다.

오류 범주	설명
OUT_OF_MEMORY_ERROR	드라이버/실행기의 메모리에 공간이 남아 있지 않은 경우 오류가 발생합니다.
IMPORT_ERROR	종속성을 가져올 때 오류가 발생합니다.
INVALID_ARGUMENT_ERROR	입력 인수가 유효하지 않거나 잘못된 경우 오류가 발생합니다.
PERMISSION_ERROR	서비스, 데이터 등에 대한 권한이 없는 경우 오류가 발생합니다.
RESOURCE_NOT_FOUND_ERROR	데이터, 위치 등이 종료되지 않을 경우 오류가 발생합니다.
QUERY_ERROR	Spark SQL 쿼리 실행으로 인해 오류가 발생합니다.
SYNTAX_ERROR	스크립트에 구문 오류가 있는 경우 오류가 발생합니다.
THROTTLING_ERROR	서비스 동시성 한도에 도달하거나 서비스 할당량 한도를 초과하는 경우 오류가 발생합니다.
DATA_LAKE_FRAMEWORK_ERROR	Hudi, Iceberg 등과 같은 AWS Glue 기본 지원 데이터 레이크 프레임워크에서 오류가 발생합니다.
UNSUPPORTED_OPERATION_ERROR	지원되지 않는 작업을 수행하는 경우 오류가 발생합니다.
RESOURCES_ALREADY_EXISTS_ERROR	생성하거나 추가하려는 리소스가 이미 존재하는 경우 오류가 발생합니다.
GLUE_INTERNAL_SERVICE_ERROR	AWS Glue 내부 서비스 문제가 있는 경우 오류가 발생합니다.
GLUE_OPERATION_TIMEOUT_ERROR	AWS Glue 작업 시간이 초과되는 경우 오류가 발생합니다.

오류 범주	설명
GLUE_VALIDATION_ERROR	AWS Glue 작업에 필요한 값을 검증할 수 없는 경우 오류가 발생합니다.
GLUE_JOB_BOOKMARK_VERSION_MISMATCH_ERROR	동일한 소스 버킷에서 동일한 작업이 실행되고 동일한/다른 대상에 동시에 작성되는 경우 오류가 발생합니다(동시성 > 1).
LAUNCH_ERROR	AWS Glue 작업 시작 단계에서 오류가 발생합니다.
DYNAMODB_ERROR	Amazon DynamoDB 서비스에서 일반 오류가 발생합니다.
GLUE_ERROR	AWS Glue 서비스에서 일반 오류가 발생합니다.
LAKEFORMATION_ERROR	AWS Lake Formation 서비스에서 일반 오류가 발생합니다.
REDSHIFT_ERROR	Amazon Redshift 서비스에서 일반 오류가 발생합니다.
S3_ERROR	Amazon S3 서비스에서 일반 오류가 발생합니다.
SYSTEM_EXIT_ERROR	일반 시스템 종료 오류입니다.
TIMEOUT_ERROR	작업 시간 초과로 인해 작업이 실패하면 일반 오류가 발생합니다.
UNCLASSIFIED_SPARK_ERROR	Spark에서 일반 오류가 발생합니다.
UNCLASSIFIED_ERROR	기본 오류 범주입니다.

제한 사항

> **Note**
>
> `glueContext`를 초기화해야만 지표를 게시할 수 있습니다.

소스 차원에서 값은 소스 유형에 따라 Amazon S3 경로 또는 테이블 이름입니다. 또한 소스가 JDBC이고 쿼리 옵션이 사용되는 경우 쿼리 문자열은 소스 차원에 설정됩니다. 값이 500자보다 길면 500자 이내로 잘립니다. 다음은 값의 제한 사항입니다.

- ASCII가 아닌 문자는 제거됩니다.
- 소스 이름에 ASCII 문자가 포함되어 있지 않으면 <비 ASCII 입력>으로 변환됩니다.

처리량 지표의 제한 사항 및 고려 사항

- DataFrame 및 DataFrame 기반 DynamicFrame(예: JDBC, Amazon S3의 Parquet에서 읽기)은 지원되지만 RDD 기반 DynamicFrame(예: Amazon S3에서 csv, json 읽기 등)은 지원되지 않습니다. 기술적으로는 Spark UI에 표시되는 모든 읽기 및 쓰기가 지원됩니다.
- 데이터 소스가 카탈로그 테이블이고 형식이 JSON, CSV, 텍스트 또는 Iceberg인 경우 `recordsRead` 지표가 내보내집니다.
- `glue.driver.throughput.recordsWritten`, `glue.driver.throughput.bytesWritten`, `glue.driver.throughput.filesWritten` 지표는 JDBC 및 Iceberg 테이블에서 사용할 수 없습니다.
- 지표가 지연될 수 있습니다. 작업이 약 1분 안에 완료되면 Amazon CloudWatch 지표에 처리량 지표가 표시되지 않을 수 있습니다.

작업 모니터링 및 디버깅

AWS Glue 작업에 대한 지표를 수집하고 AWS Glue 및 Amazon CloudWatch 콘솔에서 이러한 지표를 시각화하여 문제를 식별하고 수정할 수 있습니다. AWS Glue 작업을 프로파일링하려면 다음 단계를 수행해야 합니다.

1. 지표를 활성화하려면 다음을 수행합니다.

a. 작업 정의에서 작업 측정치 옵션을 활성화합니다. AWS Glue 콘솔에서 또는 작업의 파라미터로서 프로파일링을 활성화할 수 있습니다. 자세한 정보는 Spark 작업에 대한 작업 속성 정의 또는 AWS Glue 작업에서 작업 파라미터 사용 섹션을 참조하세요.

b. 작업 정의에서 AWS Glue 관찰성 지표 옵션을 활성화합니다. AWS Glue 콘솔에서 또는 작업의 파라미터로서 관찰성을 활성화할 수 있습니다. 자세한 정보는 AWS Glue 관찰성 메트릭을 사용한 모니터링 섹션을 참조하세요.

2. 작업 스크립트에서 `GlueContext`를 초기화하는지 확인합니다. 예를 들면 다음 스크립트 조각은 `GlueContext`를 초가화하고 프로파일링된 코드가 스크립트에 있는 위치를 표시합니다. 이 일반적인 형식은 이어지는 디버깅 시나리오에서 사용됩니다.

```
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
import time

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

...
...
code-to-profile
...
...


job.commit()
```

3. 작업을 실행합니다.

4. 지표를 시각화하려면 다음을 수행합니다.

a. AWS Glue 콘솔에서 작업 지표를 시각화하고 드라이버 또는 실행기의 이상 지표를 식별합니다.

b. 작업 실행 모니터링 페이지, 작업 실행 세부 정보 페이지 또는 Amazon CloudWatch에서 관찰성 지표를 확인합니다. 자세한 내용은 AWS Glue 관찰성 메트릭을 사용한 모니터링 단원을 참조하십시오.

5. 식별된 측정치를 사용하여 근본 원인의 범위를 좁힙니다.
6. 원한다면 식별된 드라이버나 작업 실행기의 로그 스트림을 사용하여 근본 원인을 확인할 수 있습니다.

AWS Glue 관찰성 지표의 사용 사례

- OOM 예외 사항 및 작업 이상 현상 디버깅
- 까다로운 단계와 스트래글러 작업 디버깅
- 여러 작업의 진행 상황 모니터링
- DPU 용량 계획 모니터링
- 리소스 사용률 모니터링을 위해 AWS Glue 관찰성을 사용하여 비용 절감

OOM 예외 사항 및 작업 이상 현상 디버깅

AWS Glue에서 메모리 부족(OOM) 예외 사항 및 작업 이상 현상을 디버깅할 수 있습니다. 다음 단원에서는 Apache Spark 드라이버 또는 Spark 실행기의 메모리 부족 예외 사항을 디버깅하는 시나리오를 설명합니다.

- 드라이버 OOM 예외 사항 디버깅
- 실행기 OOM 예외 사항 디버깅

드라이버 OOM 예외 사항 디버깅

이 시나리오에서는 Spark 작업이 Amazon Simple Storage Service(Amazon S3)에서 여러 작은 파일을 읽습니다. 파일을 Apache Parquet 포맷으로 변환한 다음 Amazon S3에 씁니다. Spark 드라이버에서 메모리 부족이 발생합니다. 입력 Amazon S3 데이터의 파일 수는 서로 다른 Amazon S3 파티션에 걸쳐 100만개가 넘습니다.

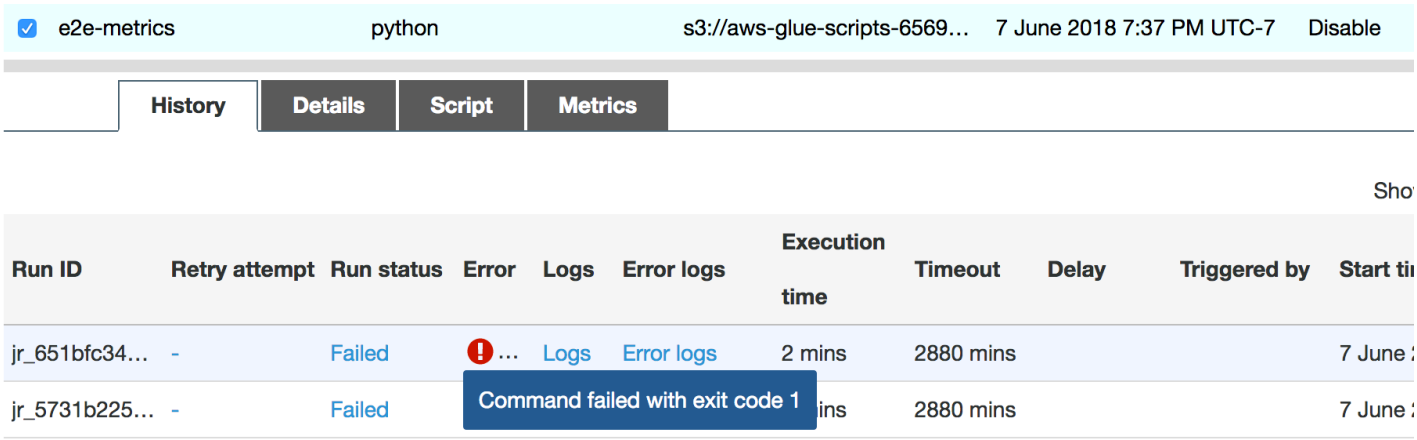
프로파일링된 코드는 다음과 같습니다.

```
data = spark.read.format("json").option("inferSchema", False).load("s3://input_path")
data.write.format("parquet").save(output_path)
```

AWS Glue 콘솔에서 프로파일링된 지표 시각화

다음 그래프는 드라이버와 실행기에 대한 메모리 사용률(%)을 표시합니다. 이 사용률은 지난 1분 안에 보고된 값에 대한 평균으로 계산된 데이터 하나로 표시됩니다. 드라이버 메모리가 50% 사용률의 안전 임계값을 넘는 작업의 메모리 프로필을 볼 수 있습니다. 반면에, 모든 실행기의 평균 메모리 사용률은 여전히 4%보다 낮습니다. 이로써 이 Spark 작업에서 드라이버 실행에 이상이 있음을 명확히 알 수 있습니다.

작업 실행이 곧 실패하고 AWS Glue 콘솔의 [기록(History)] 탭에 [종료 코드 1로 명령 실패(Command Failed with Exit Code 1)] 오류가 나타납니다. 이 오류 문자열은 시스템 전체 오류(이 경우 메모리 부족)로 인해 작업이 실패했음을 의미합니다.



콘솔의 [기록(History)] 탭에서 [오류 로그(Error logs)] 링크를 선택하여 CloudWatch Logs에서 드라이버 OOM에 대한 결과를 확인합니다. 작업의 오류 로그에서 "**`Error`**"를 검색하여 작업 실패의 원인이 된 OOM 예외 사항이 정말 발생했는지 확인합니다.

```
# java.lang.OutOfMemoryError: Java heap space
# -XX:OnOutOfMemoryError="kill -9 %p"
# Executing /bin/sh -c "kill -9 12039"...
```

작업에 대해 기록 탭에서 로그를 선택합니다. 작업의 시작 부분에서 다음과 같은 CloudWatch Logs의 드라이버 실행 추적 정보를 찾아볼 수 있습니다. Spark 드라이버에서 모든 디렉터리의 모든 파일을 나열하려고 했으며, `InMemoryFileIndex`를 생성하고 파일마다 작업 하나를 실행합니다. 이로 인해 Spark 드라이버는 모든 작업을 추적하기 위해 메모리에서 매우 많은 상태를 유지해야 합니다. 따라서 메모리 내 인덱스에 대해 다량의 파일을 포함하는 전체 목록을 캐시하므로 드라이버 OOM이 발생하게 됩니다.

그룹화를 사용하여 여러 파일의 처리 수정

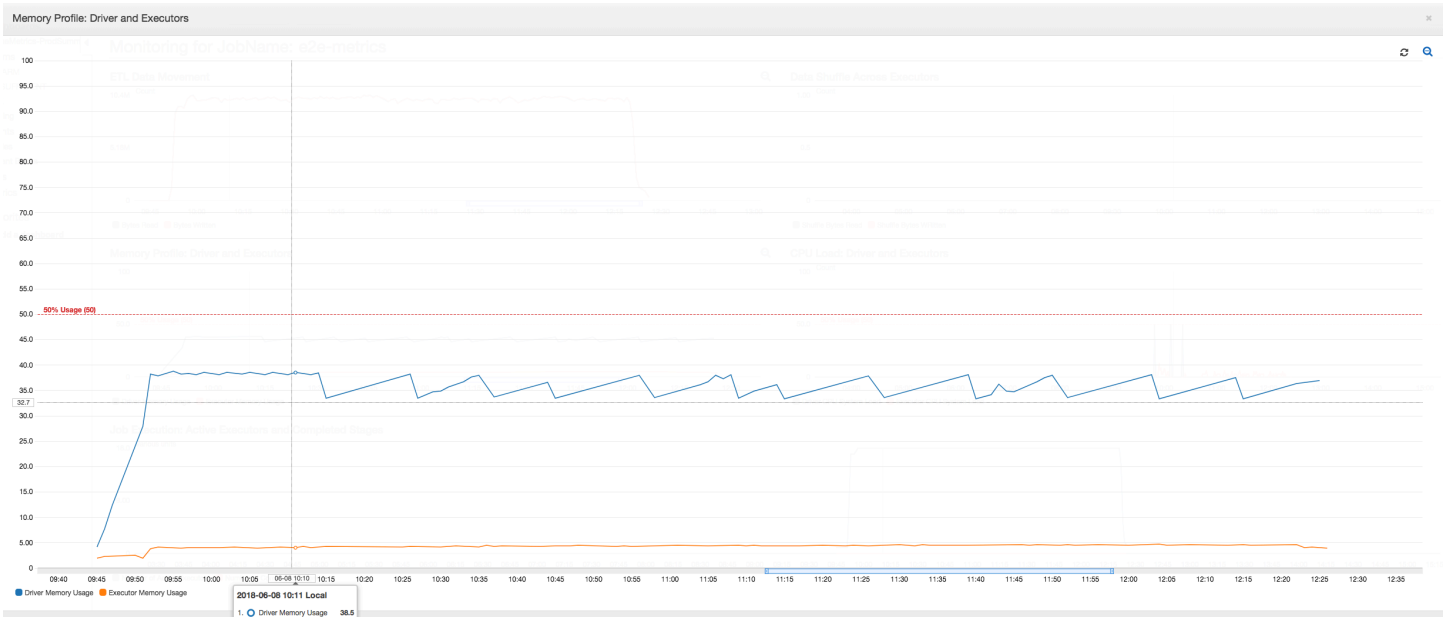
AWS Glue의 그룹화 기능을 사용하여 여러 파일의 처리를 수정할 수 있습니다. 그룹화는 동적 프레임을 사용할 때와 입력 데이터 세트에 파일 수가 많을 때(50,000개 초과) 자동으로 활성화됩니다. 그룹화를 통해 여러 파일을 그룹 하나로 합칠 수 있으므로 작업 하나로 단일 파일 대신에 전체 그룹을 처리할 수 있습니다. 결과적으로 Spark 드라이버는 메모리 안에 훨씬 더 적은 상태를 저장하여 몇 개 작업만 추적합니다. 데이터 세트에 대한 그룹화 수동 활성화에 대한 자세한 내용은 입력 파일을 더 큰 그룹에서 읽기 단원을 참조하십시오.

AWS Glue 작업의 메모리 프로파일을 확인하려면 그룹화가 활성화된 상태에서 다음과 같은 코드를 프로파일링합니다.

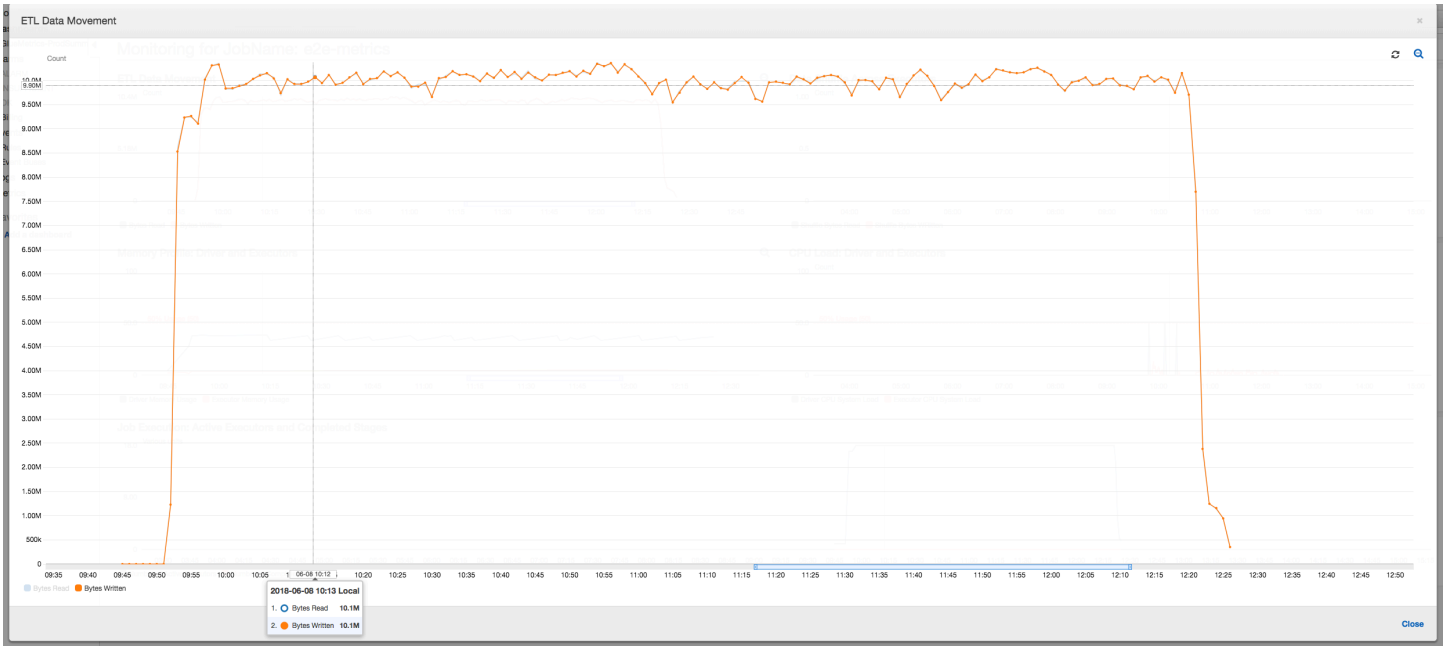
```
df = glueContext.create_dynamic_frame_from_options("s3", {'paths': ["s3://input_path"],
 "recurse":True, 'groupFiles': 'inPartition'}, format="json")
datasink = glueContext.write_dynamic_frame.from_options(frame = df, connection_type
 = "s3", connection_options = {"path": output_path}, format = "parquet",
 transformation_ctx = "datasink")
```

AWS Glue 작업 프로필에서 메모리 프로필 및 ETL 데이터 이동을 모니터링할 수 있습니다.

이 드라이버는 AWS Glue 작업의 전체 기간 동안 메모리 사용량의 50%인 임계값 미만으로 실행됩니다. 실행기가 Amazon S3에서 데이터를 스트리밍하고 처리한 후 Amazon S3에 씁니다. 결과적으로 모든 시점에서 5%보다 적은 양의 메모리가 사용됩니다.



아래 데이터 이동 프로파일은 작업이 진행됨에 따라 모든 실행기가 마지막 1분 동안 읽고 쓴 총 Amazon S3 바이트 수를 보여줍니다. 둘 다 모든 실행기에서 데이터가 스트리밍될 때 유사한 패턴을 따릅니다. 이 작업은 세 시간 내에 100만 개에 이르는 모든 파일에 대한 처리를 완료합니다.



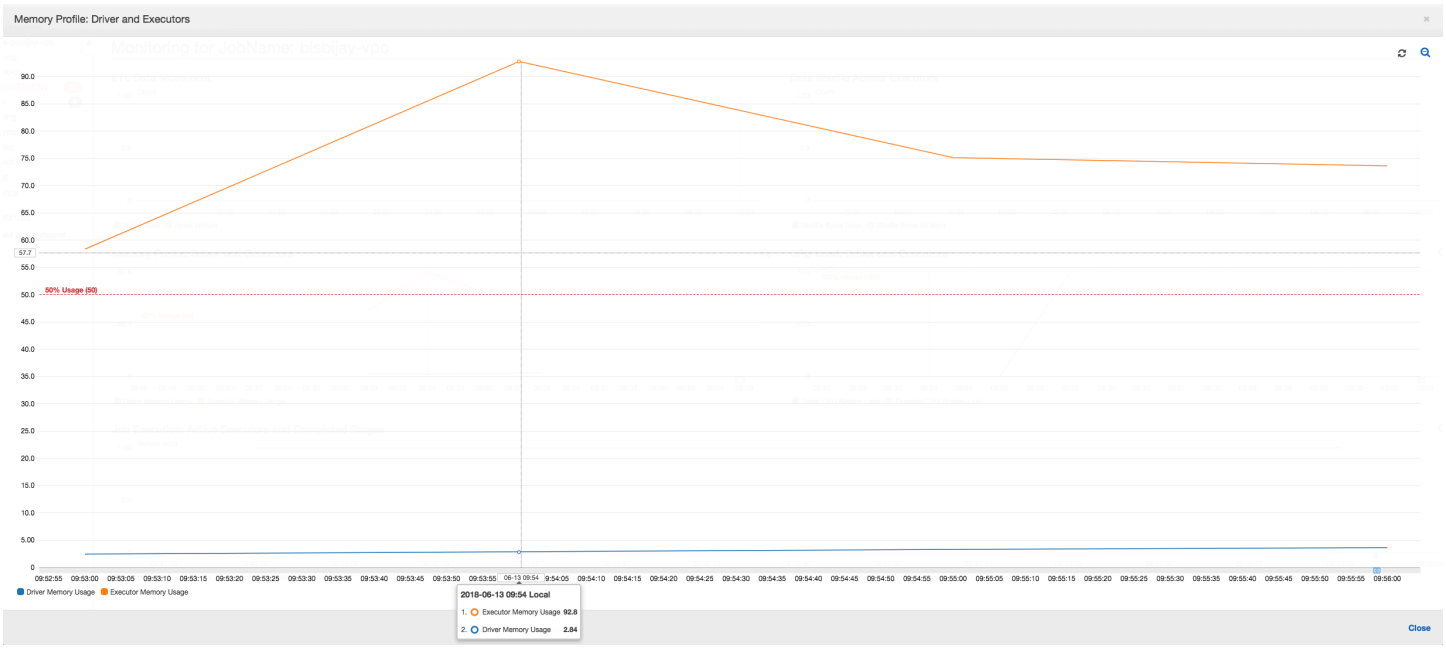
실행기 OOM 예외 사항 디버깅

이 시나리오에서는 Apache Spark 실행기에서 발생할 수 있는 OOM 예외 사항을 디버깅하는 방법을 알아볼 수 있습니다. 다음 코드는 Spark MySQL 리더를 사용하여 약 3400만개 행으로 이루어진 대형 테이블을 Spark 데이터 프레임으로 읽어 들입니다. 그런 다음 Parquet 포맷으로 Amazon S3에 씁니다. 연결 속성을 제공하고 기본 Spark 구성을 사용하여 테이블을 읽을 수 있습니다.

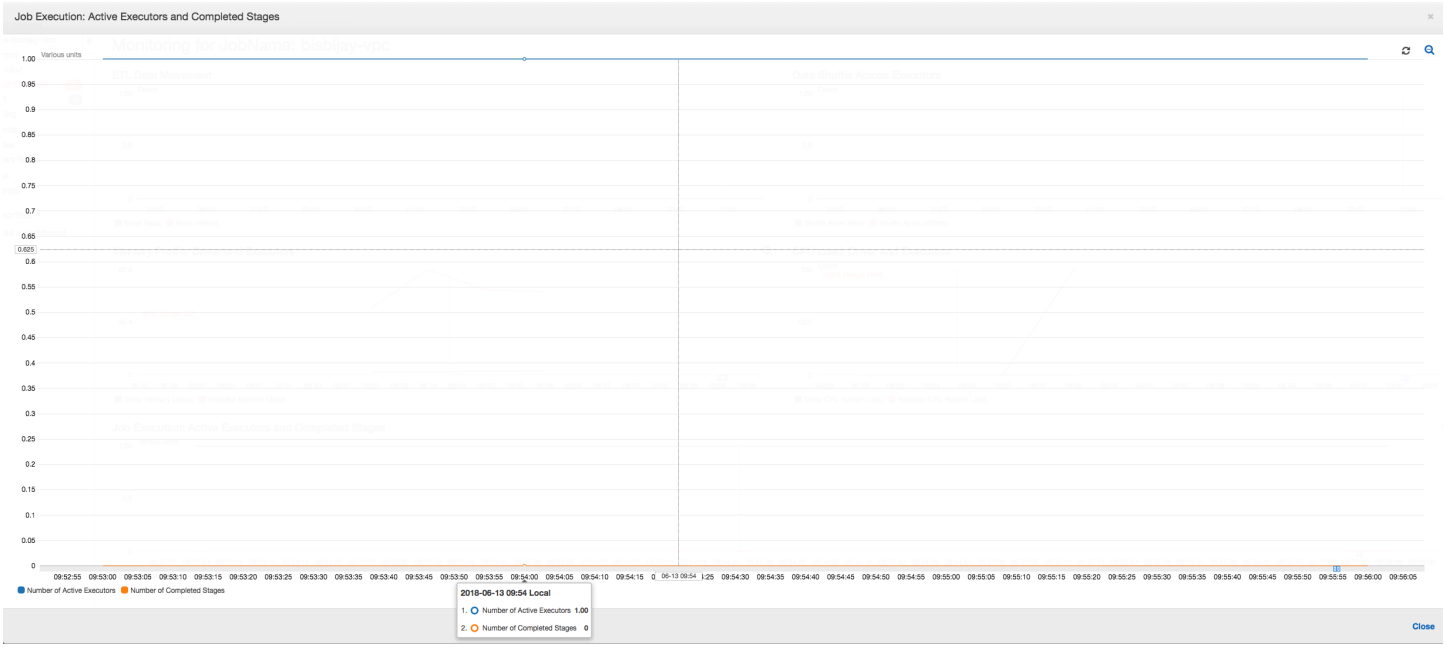
```
val connectionProperties = new Properties()
connectionProperties.put("user", user)
connectionProperties.put("password", password)
connectionProperties.put("Driver", "com.mysql.jdbc.Driver")
val sparkSession = glueContext.sparkSession
val dfSpark = sparkSession.read.jdbc(url, tableName, connectionProperties)
dfSpark.write.format("parquet").save(output_path)
```

AWS Glue 콘솔에서 프로파일링된 지표 시각화

메모리 사용량 그래프의 경사가 양수이고 50%를 넘는 경우 다음 지표를 내보내기 전에 작업이 실패하면 메모리 부족이 실패 원인이 될 수 있습니다. 다음 그래프는 1분 실행 기간 내에 모든 실행기의 평균 메모리 사용률이 50%를 급격하게 초과함을 보여줍니다. 이 사용률은 최대 92%에 도달하며 해당 실행기를 실행 중인 컨테이너가 Apache Hadoop YARN에 의해 중지됩니다.



다음 그래프에서처럼, 작업이 실패할 때까지 항상 단일 실행기가 실행 중입니다. 이는 새 실행기가 시작되어 중지된 실행기를 대체하기 때문입니다. JDBC 데이터 원본 읽기는 기본적으로 병렬화되지 않습니다. 그 이유는 한 열에서 테이블을 분할하고 여러 연결을 열어야 하기 때문입니다. 따라서 실행기 하나만 전체 테이블에서 순차적으로 읽혀집니다.



다음 그래프에서처럼, Spark는 작업 실패 전에 새 작업 시작을 4회 시도합니다. 세 개 실행기의 메모리 프로필을 볼 수 있습니다. 각 실행기가 모든 메모리를 신속히 소비합니다. 네 번째 실행기에서 메모리 부족이 발생하여 작업이 실패합니다. 따라서 이 작업의 측정치가 즉시 보고되지 않습니다.

다음 이미지와 같이, OOM 예외로 인해 작업이 실패했던 AWS Glue 콘솔에서 오류 문자열을 확인할 수 있습니다.

작업 출력 로그: 실행기 OOM 예외 사항의 결과를 추가로 확인하려면 CloudWatch Logs를 살펴봅니다. **`Error`**를 검색하면 지표 대시보드처럼 거의 동일한 기간 내에 네 개 실행기가 중지됨을 확인할 수 있습니다. 이들 실행기는 메모리 제한을 초과할 때 YARN에 의해 모두 종료됩니다.

실행기 1

```
18/06/13 16:54:29 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 ERROR YarnClusterScheduler: Lost executor 1 on
 ip-10-1-2-175.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 WARN TaskSetManager: Lost task 0.0 in stage 0.0 (TID 0,
 ip-10-1-2-175.ec2.internal, executor 1): ExecutorLostFailure (executor 1
```

```
 exited caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

실행기 2

```
18/06/13 16:55:35 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 ERROR YarnClusterScheduler: Lost executor 2 on
 ip-10-1-2-16.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 WARN TaskSetManager: Lost task 0.1 in stage 0.0 (TID 1,
 ip-10-1-2-16.ec2.internal, executor 2): ExecutorLostFailure (executor 2 exited
 caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

실행기 3

```
18/06/13 16:56:37 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 ERROR YarnClusterScheduler: Lost executor 3 on
 ip-10-1-2-189.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 WARN TaskSetManager: Lost task 0.2 in stage 0.0 (TID 2,
 ip-10-1-2-189.ec2.internal, executor 3): ExecutorLostFailure (executor 3
 exited caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

실행기 4

```
18/06/13 16:57:18 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 ERROR YarnClusterScheduler: Lost executor 4 on
 ip-10-1-2-96.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 WARN TaskSetManager: Lost task 0.3 in stage 0.0 (TID 3,
 ip-10-1-2-96.ec2.internal, executor 4): ExecutorLostFailure (executor 4 exited
 caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

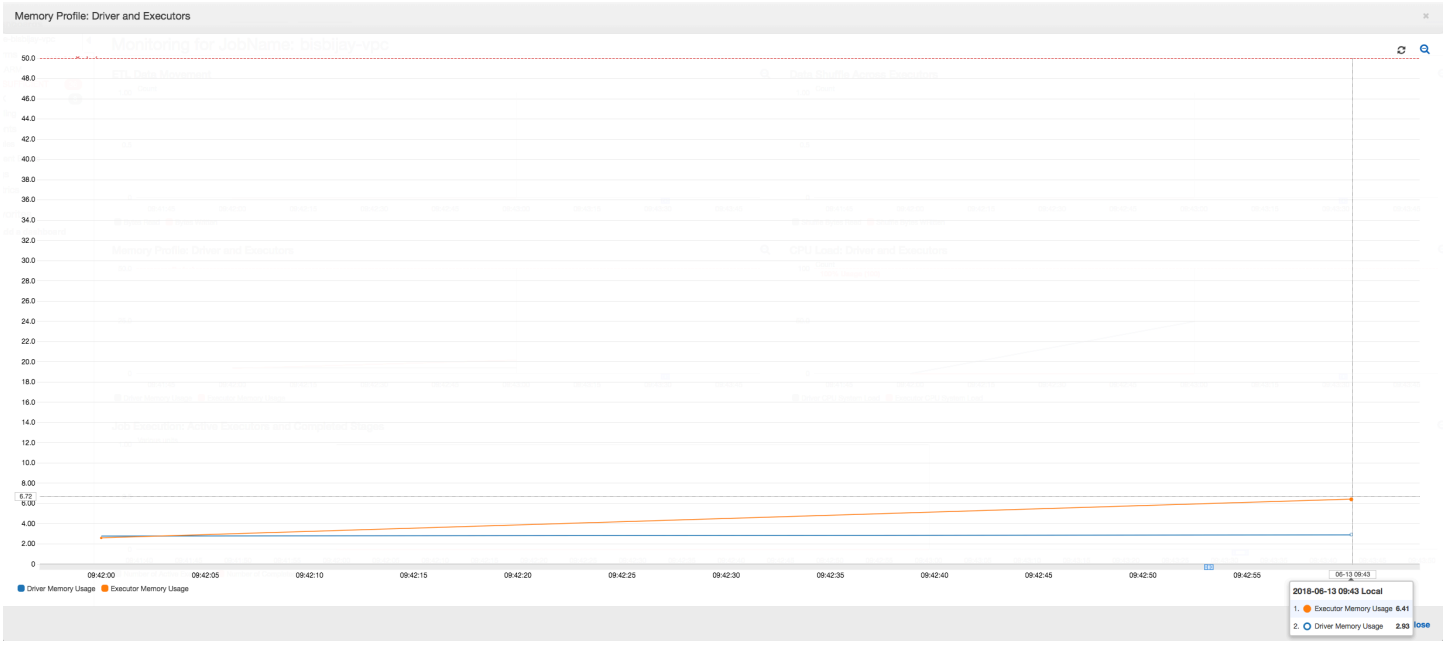
AWS Glue 동적 프레임을 사용하여 가져오기 크기 설정 수정

Spark JDBC 가져오기 크기에 대한 기본 구성이 0이므로 JDBC 테이블을 읽는 동안 실행기에서 메모리 부족이 발생했습니다. 이는 Spark가 한 번에 한 개씩 행을 스트리밍하더라도 Spark 실행기의 JDBC 드라이버가 데이터베이스의 3400만개 행을 함께 가져와서 캐시하려고 함을 의미합니다. Spark에서 가져오기 크기 파라미터를 0 이외의 기본값으로 설정하여 이 시나리오를 방지할 수 있습니다.

또한 AWS Glue 동적 프레임을 대신 사용하여 이 문제를 해결할 수도 있습니다. 기본적으로 동적 프레임에서는 가져오기 크기로 1,000개 행이 사용되는데, 일반적으로 충분한 값입니다. 결과적으로 실행기에 사용되는 메모리의 양이 총 메모리의 7%를 초과하지 않습니다. AWS Glue 작업은 단일 실행기만으로 2분 안에 완료됩니다. AWS Glue 동적 프레임을 사용하는 것이 권장되는 접근 방식이지만, Apache Spark `fetchsize` 속성을 사용하여 가져오기 크기를 설정할 수도 있습니다. Spark SQL, DataFrames 및 데이터 세트 가이드를 참조하십시오.

```
val (url, database, tableName) = {
 ("jdbc_url", "db_name", "table_name")
 }
val source = glueContext.getSource(format, sourceJson)
val df = source.getDynamicFrame
glueContext.write_dynamic_frame.from_options(frame = df, connection_type = "s3",
 connection_options = {"path": output_path}, format = "parquet", transformation_ctx =
 "datasink")
```

정상 프로파일링된 측정치: 실행기 메모리(AWS Glue 동적 프레임 사용)가 다음 이미지와 같이 안전 임곗값을 절대로 초과하지 않습니다. 데이터베이스에서 행을 읽고 어느 시점에든 JDBC 드라이버에 1,000개 행만 캐시합니다. 메모리 부족 예외는 발생하지 않습니다.



까다로운 단계와 스트래글러 작업 디버깅

AWS Glue 작업 프로파일링을 사용하여 추출, 변환 및 로드(ETL) 작업에서 까다로운 단계와 스트래글러 작업을 식별할 수 있습니다. 스트래글러 작업은 AWS Glue 작업의 단계에서 나머지 작업보다 더 오래 걸립니다. 따라서 해당 단계를 완료하는 데 더 오래 걸리므로 작업의 총 실행 시간도 지연됩니다.

작은 입력 파일을 큰 출력 파일로 병합

스트래글러 작업은 여러 작업들 간에 작업량이 균일하지 않게 배포되거나, 데이터 스큐로 인해 한 작업에서 더 많은 데이터를 처리하는 경우에 발생할 수 있습니다.

다음 코드(Apache Spark의 일반 패턴)를 프로파일링하여 다량의 작은 파일을 큰 출력 파일로 병합할 수 있습니다. 예를 들면, 입력 데이터 세트는 JSON Gzip 압축 파일 32GB입니다. 출력 데이터 세트는 거의 190GB에 이르는 압축되지 않은 JSON 파일입니다.

프로파일링된 코드는 다음과 같습니다.

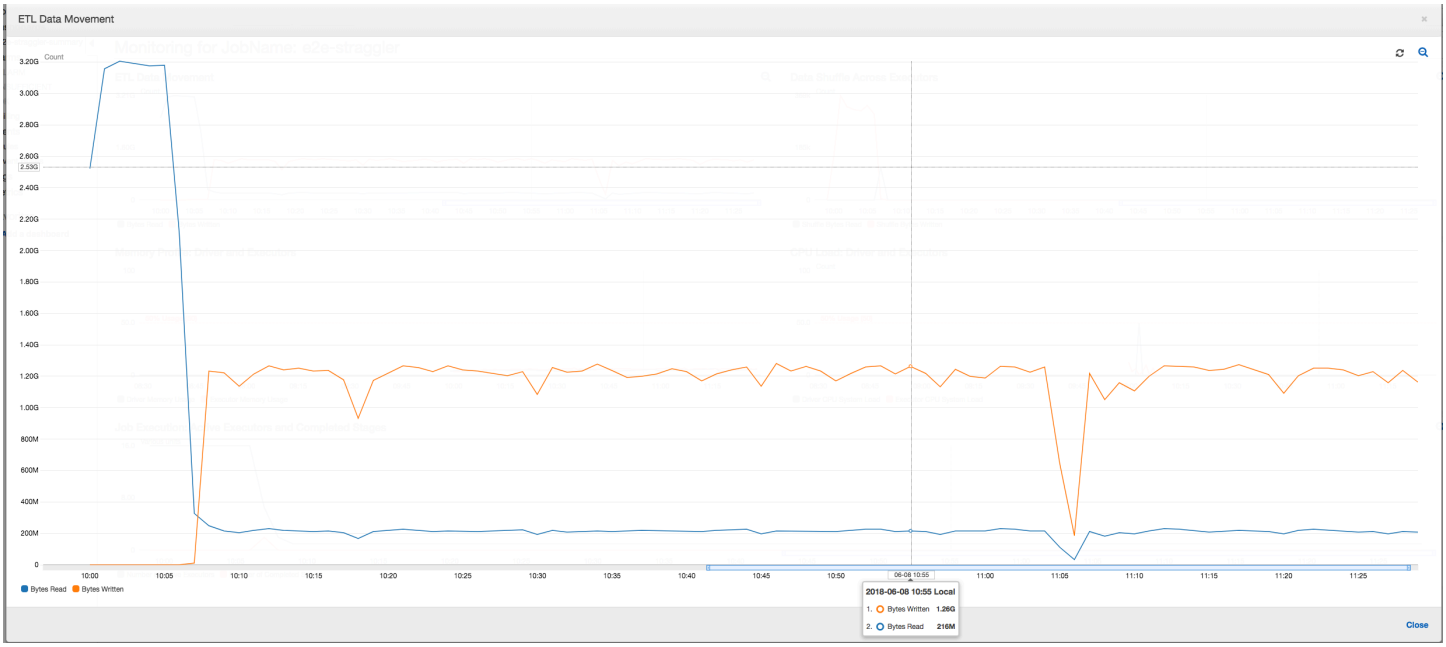
```
datasource0 = spark.read.format("json").load("s3://input_path")
df = datasource0.coalesce(1)
df.write.format("json").save(output_path)
```

AWS Glue 콘솔에서 프로파일링된 지표 시각화

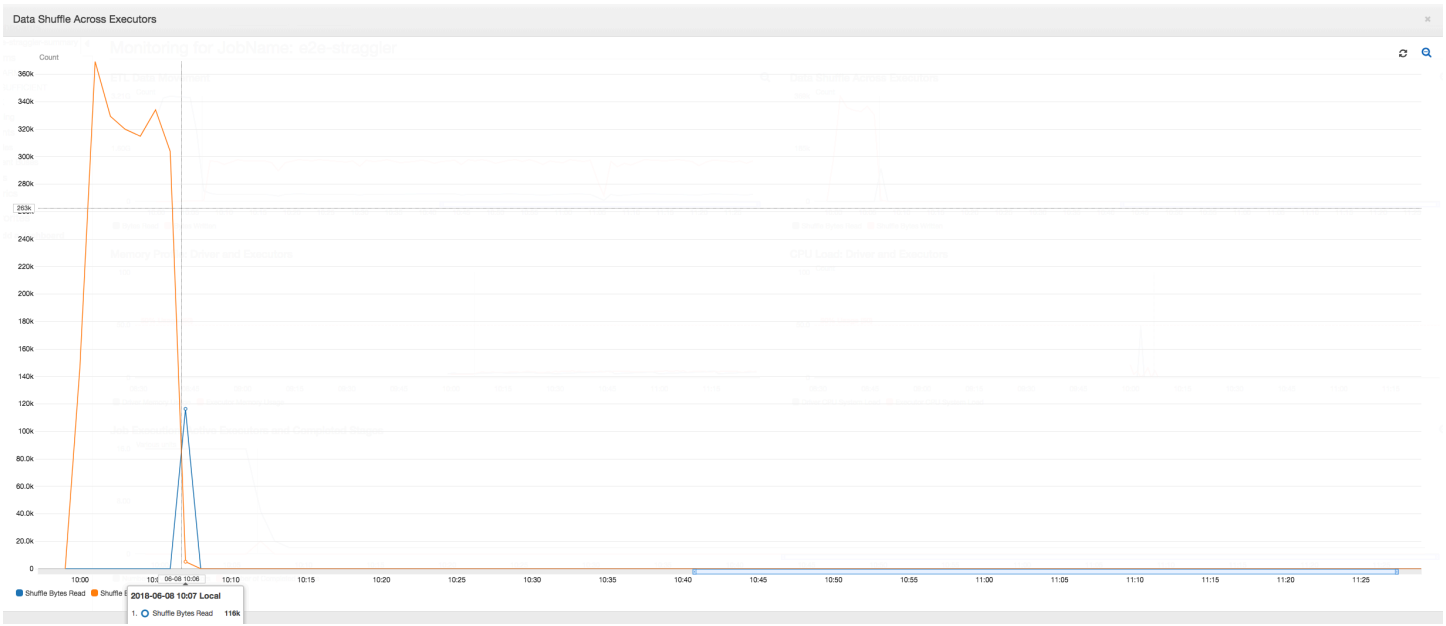
작업을 프로파일링하여 네 가지 지표 세트를 살펴볼 수 있습니다.

- ETL 데이터 이동
- 실행기 간의 데이터 셔플
- 작업 실행
- 메모리 프로필

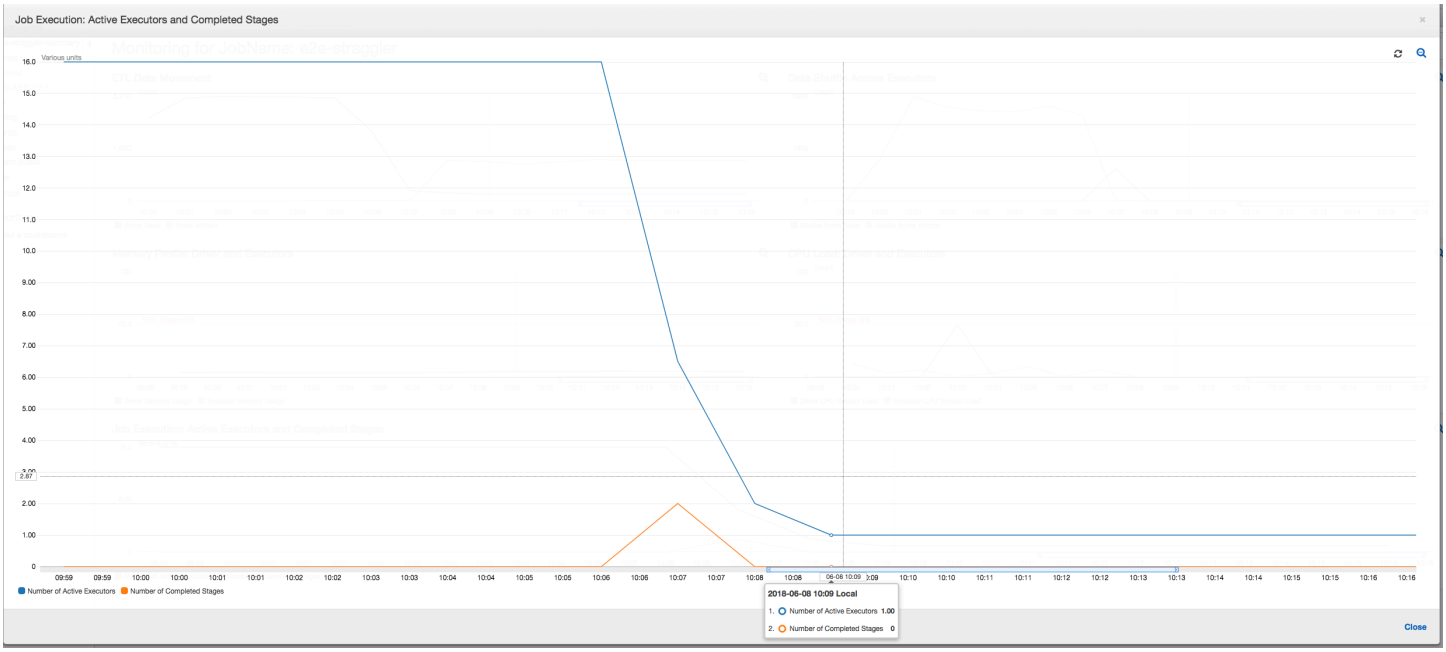
ETL 데이터 이동: ETL 데이터 이동 프로필에서는 처음 6분 안에 완료되는 첫 번째 단계에서 모든 실행기가 바이트를 매우 빠르게 읽습니다. 하지만 총 작업 실행 시간은 거의 1시간으로, 이 시간 중 대부분을 데이터 쓰기가 차지합니다.



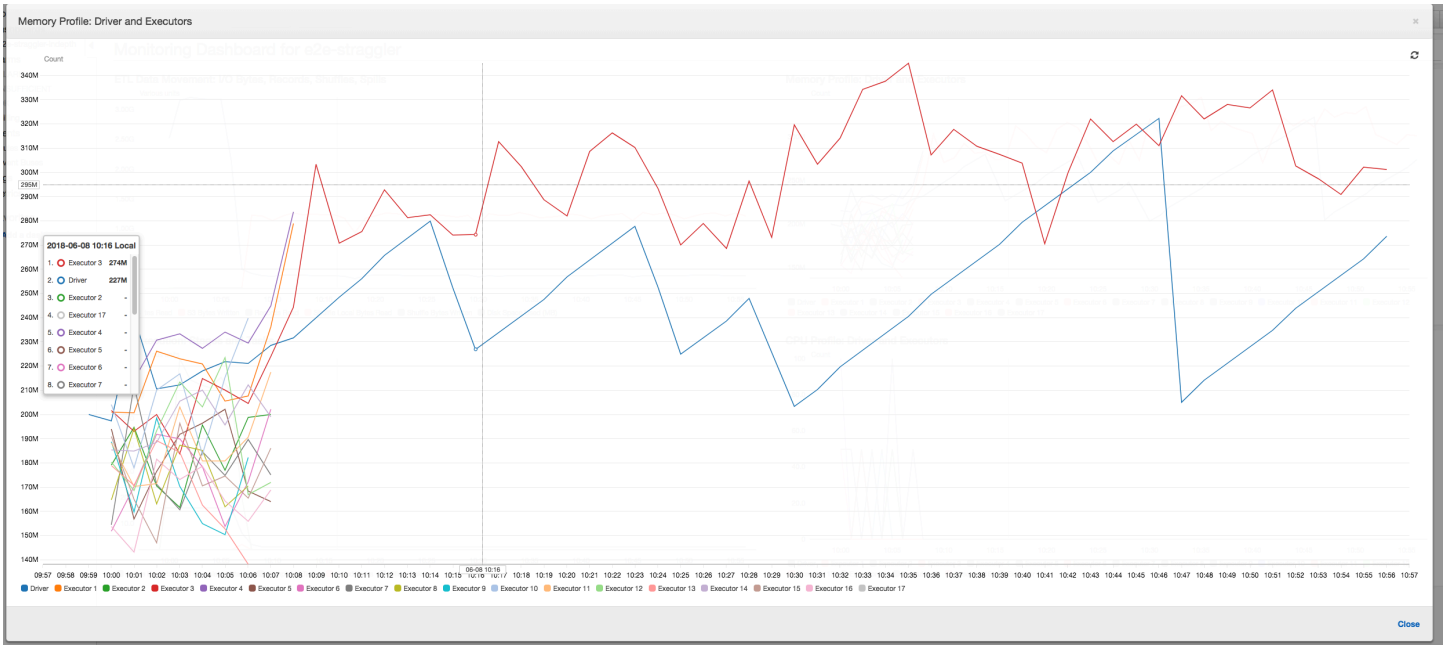
실행기 간 데이터 셔플: 셔플링 중에 읽고 쓴 바이트 수도 [작업 실행(Job Execution)] 및 [데이터 셔플(Data Shuffle)] 지표에서 알 수 있듯이 2단계가 끝나기 전 스파이크를 나타냅니다. 모든 실행기의 데이터 셔플 후에는 실행기 번호 3에서만 읽기 및 쓰기가 진행됩니다.



작업 실행: 아래 그래프와 같이, 다른 모든 실행기는 유휴 상태이며 최종적으로 10:09에 중지됩니다. 이 시점에는 총 실행기 수가 1로 감소됩니다. 이는 실행기 번호 3이 실행 시간이 가장 오래 걸리고 작업 실행 시간의 대부분을 차지하는 스트래글러 작업으로 이루어짐을 보여줍니다.



메모리 프로필: 처음 두 단계 이후에는 실행기 번호 3에서 메모리를 적극적으로 소비하며 데이터를 처리합니다. 나머지 실행기는 단순 유휴 상태이거나 처음 두 단계가 완료되고 나서 잠시 후에 중지되었습니다.



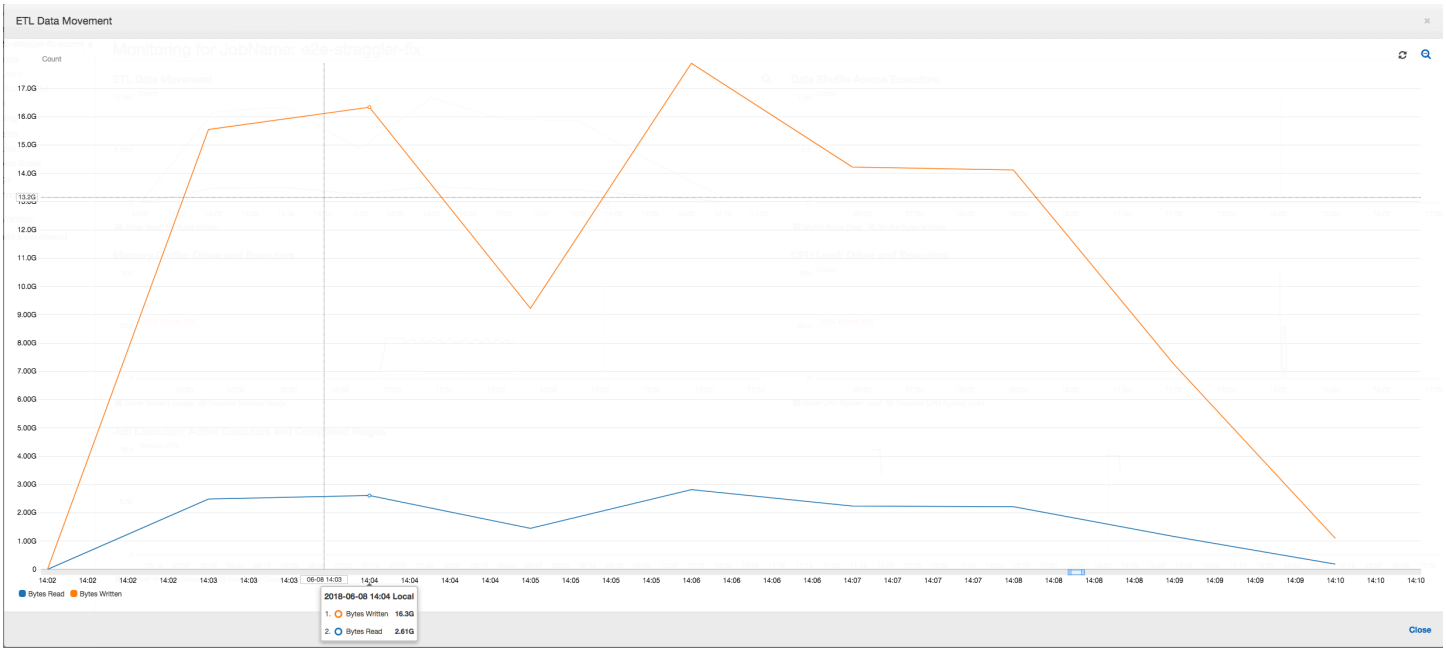
그룹화를 사용하여 스트래글링 실행기 수정

AWS Glue의 그룹화 기능을 사용하면 실행기가 뒤처지는 것을 방지할 수 있습니다. 그룹화를 사용하면 데이터를 모든 실행기에 균일하게 분포하고 클러스터의 모든 사용 가능한 실행기를 사용하여 파일을 더 큰 파일로 병합할 수 있습니다. 자세한 내용은 입력 파일을 더 큰 그룹에서 읽기 단원을 참조하십시오.

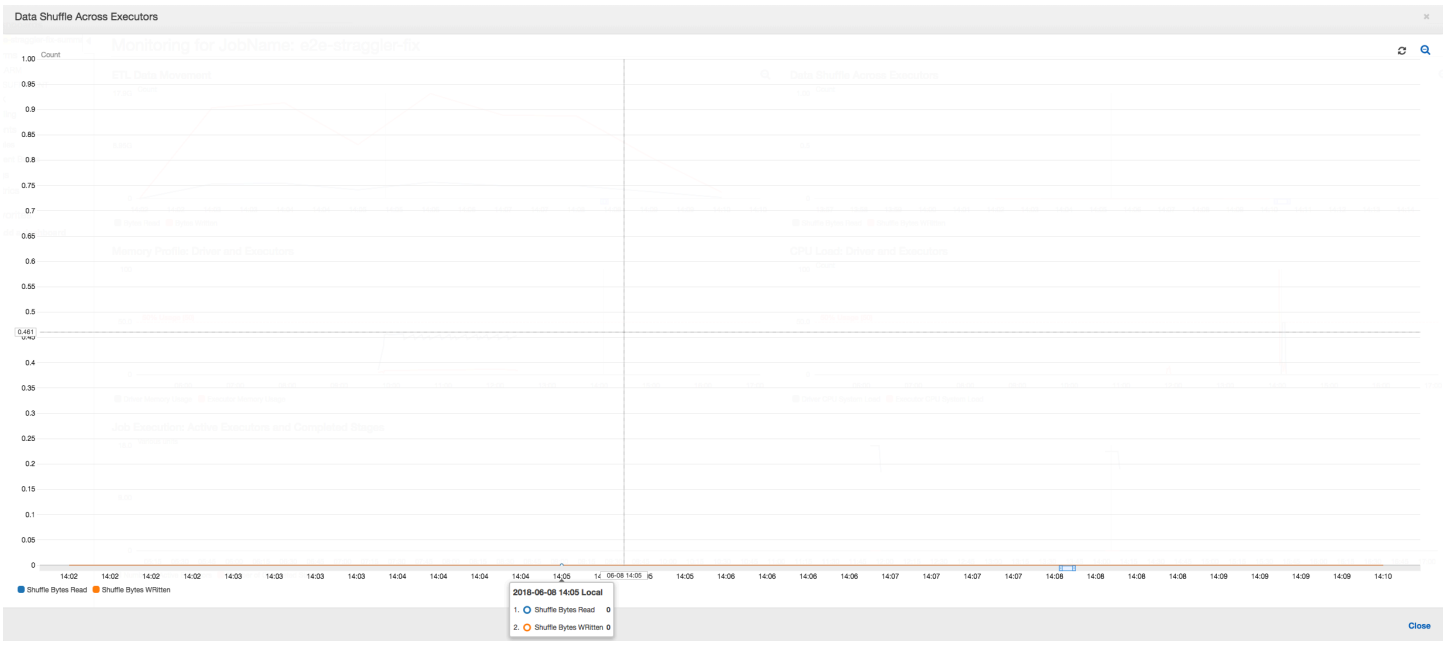
AWS Glue 작업의 ETL 데이터 이동을 확인하려면 그룹화가 활성화된 상태에서 다음과 같은 코드를 프로파일링합니다.

```
df = glueContext.create_dynamic_frame_from_options("s3", {'paths': ["s3://input_path"],
 "recurse":True, 'groupFiles': 'inPartition'}, format="json")
datasink = glueContext.write_dynamic_frame.from_options(frame = df, connection_type =
 "s3", connection_options = {"path": output_path}, format = "json", transformation_ctx
 = "datasink4")
```

ETL 데이터 이동: 이제 작업 실행 시간 전체에 걸쳐 데이터 쓰기와 데이터 읽기를 병렬로 스트리밍합니다. 따라서 작업이 8분(이전보다 훨씬 빠름) 내에 완료됩니다.

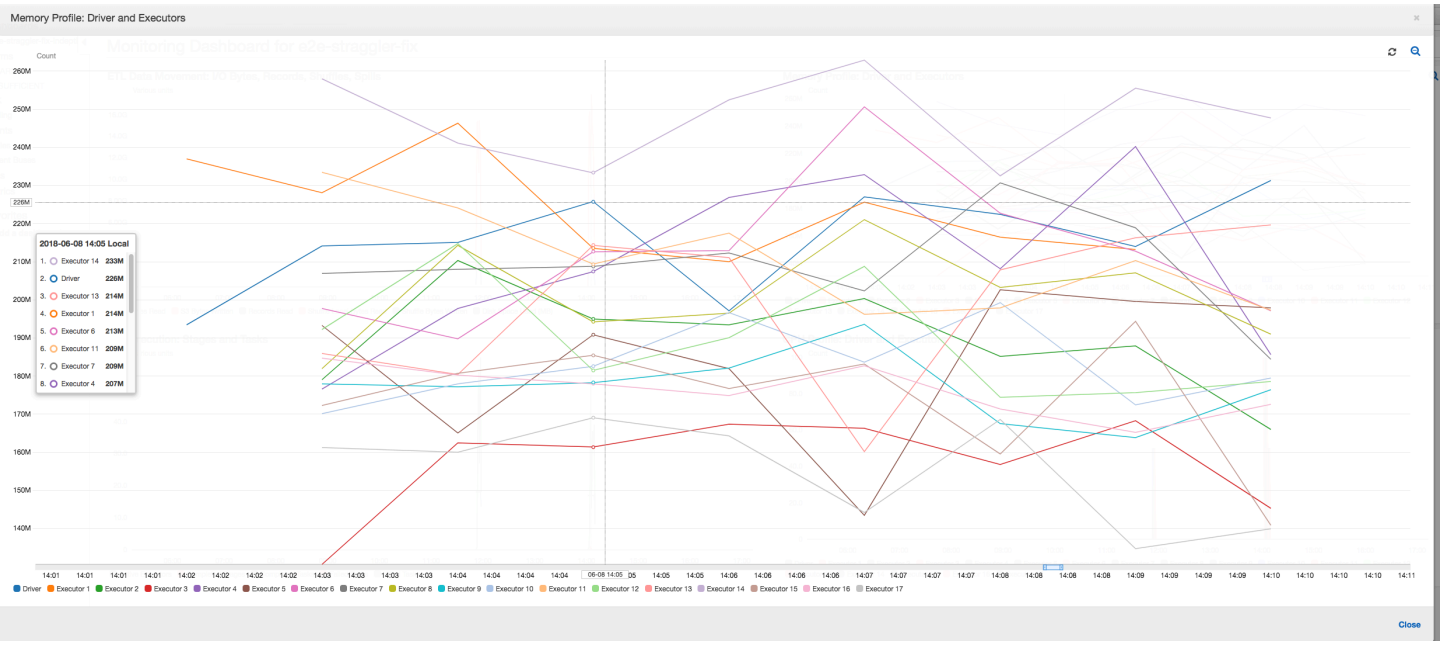


실행기 간의 데이터 셔플: 입력 파일이 그룹화 기능을 사용하여 읽는 동안 병합되므로 데이터를 읽은 후에 비용이 많이 들던 데이터 셔플이 발생하지 않습니다.



작업 실행: 작업 실행 측정치는 데이터 실행 및 처리 중인 총 활성 실행기 수가 상당히 일관되게 유지됨을 보여줍니다. 작업에 단일 스트래글러가 없습니다. 모든 실행기가 활성 상태이며 작업이 완료될 때까지 활성 상태로 유지됩니다. 실행기 간에 중간 데이터 셔플이 없으므로 작업에 단일 단계만 존재합니다.

메모리 프로파일: 이 지표는 모든 실행기의 활성 메모리 소비를 표시하여 모든 실행기에서 활동이 존재함을 재확인합니다. 데이터가 동시에 스트리밍되고 작성되므로, 모든 실행기의 총 메모리 공간이 모든 실행기의 안전 임계값보다 훨씬 낮고 거의 균일합니다.



여러 작업의 진행 상황 모니터링

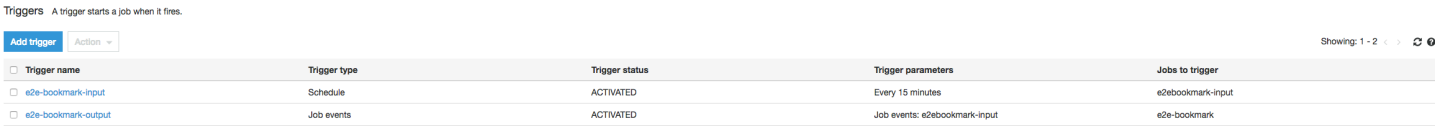
여러 AWS Glue 작업을 함께 프로파일링하고 작업 간에 데이터 흐름을 모니터링할 수 있습니다. 이 모니터링은 일반적인 워크플로우 패턴으로, 개별 작업 진행 상황, 데이터 처리 백로그, 데이터 재처리 및 작업 북마크를 모니터링해야 합니다,

주제

- 프로파일링된 코드
- AWS Glue 콘솔에서 프로파일링된 지표 시각화
- 파일 처리 수정

프로파일링된 코드

이 워크플로우에는 두 가지 작업, 즉 입력 작업과 출력 작업이 있습니다. 입력 작업은 정기 트리거를 사용하여 30분마다 실행하도록 예약되어 있습니다. 출력 작업은 입력 작업의 각각의 성공적인 실행 이후에 실행하도록 예약되어 있습니다. 이러한 예약된 작업은 작업 트리거를 사용하여 제어됩니다.



입력 작업: 이 작업은 Amazon Simple Storage Service(Amazon S3) 위치에서 데이터를 읽고 ApplyMapping을 사용하여 변환한 다음, 스테이징 Amazon S3 위치에 씁니다. 다음 코드는 입력 작업용으로 프로파일링된 코드입니다.

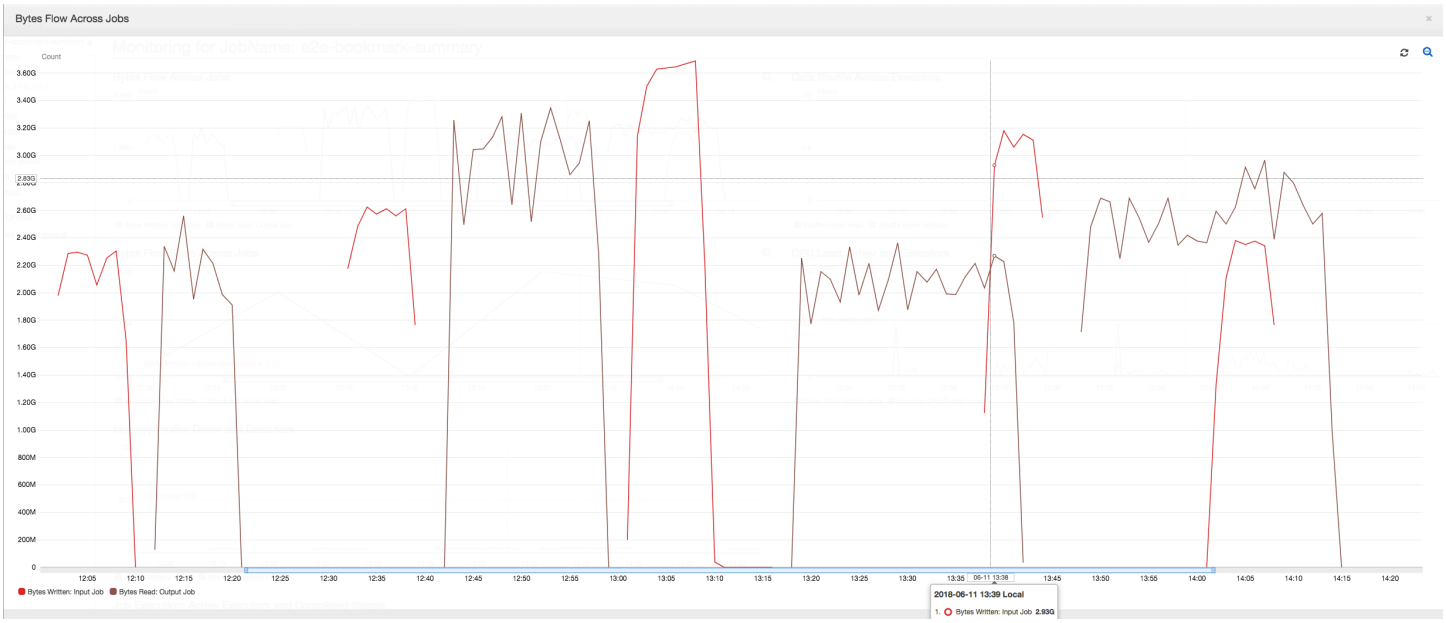
```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": ["s3://input_path"],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [map_spec])
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": staging_path, "compression":
 "gzip"}, format = "json")
```

출력 작업: 이 작업은 Amazon S3의 스테이징 위치에서 입력 작업의 출력을 읽고. 다시 변환한 다음, 대상에 씁니다.

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [staging_path],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [map_spec])
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": output_path}, format = "json")
```

AWS Glue 콘솔에서 프로파일링된 지표 시각화

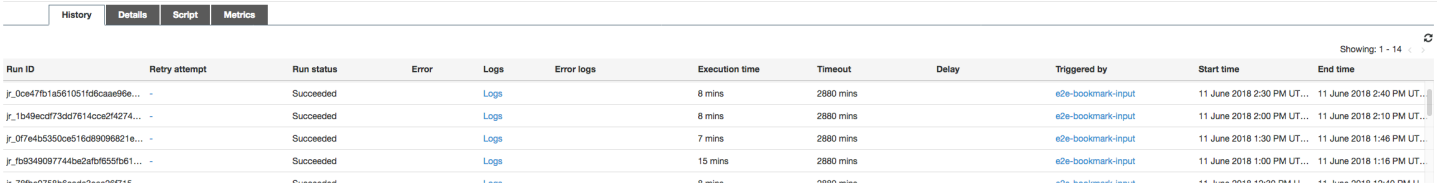
다음 대시보드는 출력 작업에 대한 동일한 타임라인에서 입력 작업의 Amazon S3 바이트 쓰기 지표를 Amazon S3 바이트 읽기 지표에 중첩합니다. 이 타임라인은 입력 작업과 출력 작업의 서로 다른 작업 실행을 보여줍니다. 입력 작업(빨간색으로 표시됨)이 30분마다 시작됩니다. 출력 작업(갈색으로 표시됨)은 입력 작업 완료 시 시작되며, 최대 동시성은 1입니다.



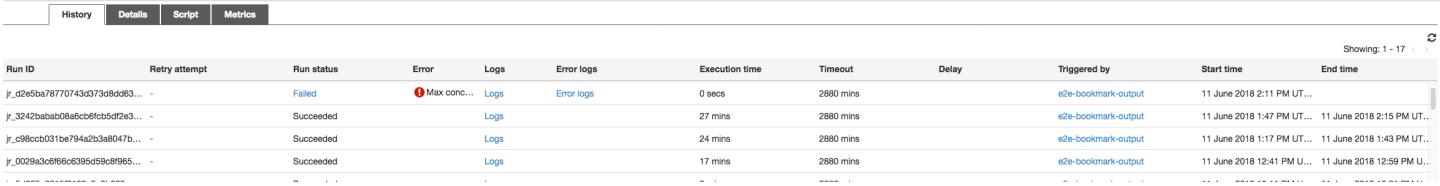
이 예에서는, 작업 북마크가 활성화되지 않습니다. 변환 컨텍스트를 사용하여 스크립트 코드에서 작업 북마크를 활성화하지 않습니다.

작업 기록: 입력 및 출력 작업은 오후 12:00부터 시작되는 기록 탭에서처럼 여러 번 실행되었습니다.

AWS Glue 콘솔의 입력 작업은 다음과 같습니다.

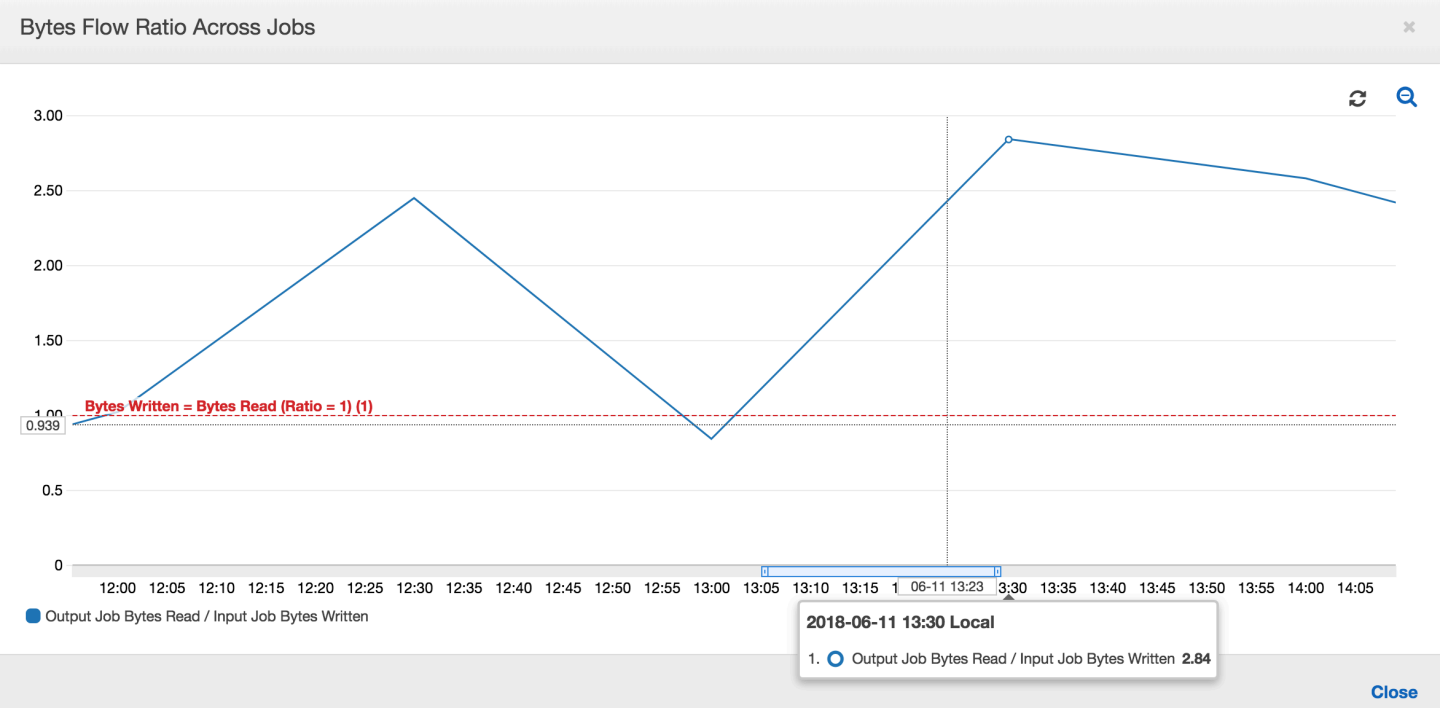


다음 이미지는 출력 작업을 보여줍니다.



첫 번째 작업 실행: 아래 읽고 쓴 데이터 바이트 그래프에서처럼, 12:00 ~ 12:30의 입력 및 출력 작업의 첫 번째 작업 실행은 거의 동일한 곡선하 면적을 보입니다. 이 면적은 입력 작업에서 쓴 Amazon S3 바이트와 출력 작업에서 읽은 Amazon S3 바이트를 나타냅니다. 또한 이 데이터는 작성된 Amazon S3 바이트의 비율(30분 동안의 합계 - 입력 작업의 경우 작업 트리거 빈도)로 확인됩니다. 오후 12:00에 시작된 입력 작업 비율의 데이터 지점도 1입니다.

다음 그래프는 모든 작업 실행에 대한 데이터 흐름 비율을 보여줍니다.



두 번째 작업 실행: 두 번째 작업 실행에서 입력 작업에서 쓴 바이트 수와 출력 작업에서 읽은 바이트 수 사이에는 명확한 차이가 있습니다. (출력 작업에 대한 두 작업 실행의 곡선하 면적을 비교하거나, 입력 작업과 출력 작업의 두 번째 실행 시 면적을 비교합니다.) 읽은 바이트 수와 쓴 바이트 수의 비율은 12:30 ~ 13:00의 두 번째 기간인 30분 동안 입력 작업에 의해 작성된 데이터의 약 2.5배를 출력 작업에서 읽었음을 보여줍니다. 이는 작업 북마크가 활성화되지 않았기 때문에 출력 작업이 입력 작업의 첫 번째 작업 실행에서 나오는 출력을 재처리하기 때문입니다. 비율이 1보다 크면 출력 작업에 의해 데이터의 추가 백로그가 처리되었음을 의미합니다.

세 번째 작업 실행: 입력 작업은 작성된 바이트 수 면에서 일관성이 거의 유지됩니다(빨간색 곡선하 면적 참조). 하지만, 입력 작업의 세 번째 작업 실행 시간이 예상보다 더 길어졌습니다(빨간색 곡선의 긴 후미부 참조). 결과적으로 출력 작업의 세 번째 작업 실행이 늦게 시작되었습니다. 세 번째 작업은 13:00 ~ 13:30의 나머지 30분 동안 스테이징 위치에 누적된 데이터 부분만 처리했습니다. 바이트 흐름의 비율은 입력 작업의 세 번째 작업 실행에 의해 작성된 데이터의 0.83만 처리되었음을 보여줍니다(13:00의 비율 참조).

입력 작업과 출력 작업의 중첩: 입력 작업의 네 번째 작업 실행이 출력 작업의 세 번째 작업 실행이 완료되기 전에 일정에 따라 13:30에 시작되었습니다. 두 작업 실행 사이에 약간의 중첩 부분이 있습니다. 하지만, 출력 작업의 세 번째 작업 실행은 약 13:17에 시작되었을 때 Amazon S3의 스테이징 위치에 나열되어 있는 파일만 캡처합니다. 캡처 데이터는 입력 작업의 첫 번째 작업 실행에서 얻은 모든 데이터 출력으로 구성됩니다. 13:30에서 실제 비율은 약 2.75입니다. 출력 작업의 세 번째 작업 실행에서는 13:30~14:00의 입력 작업 중 네 번째 작업 실행에서 작성된 데이터의 약 2.75x를 처리했습니다.

이러한 이미지에 표시된 바와 같이, 출력 작업은 입력 작업에 대한 모든 이전 작업 실행의 스테이징 위치부터 데이터를 처리하고 있습니다. 결과적으로, 출력 작업에 대한 네 번째 작업 실행이 가장 길며 입력 작업의 전체 다섯 번째 작업 실행과 겹칩니다.

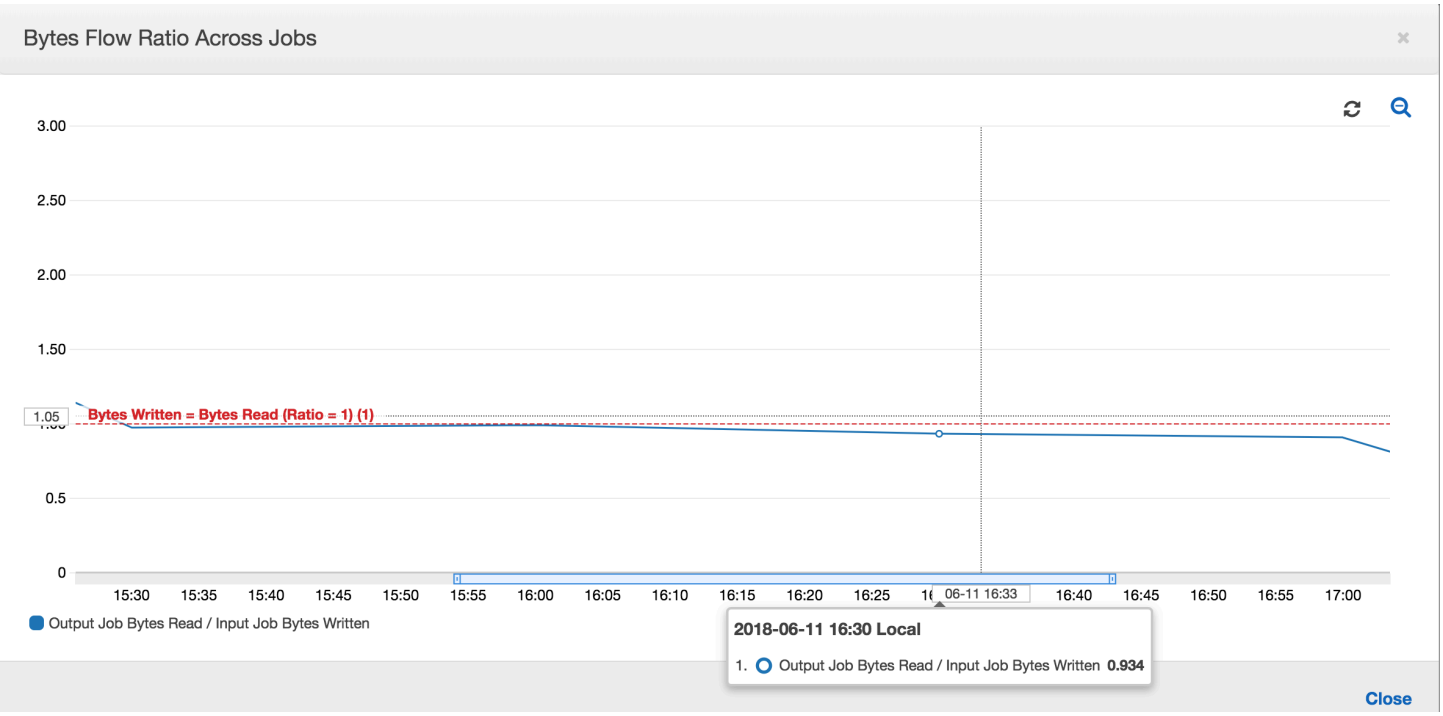
파일 처리 수정

출력 작업에서 출력 작업의 이전 작업 실행에 의해 처리되지 않은 파일만 처리하도록 해야 합니다. 이렇게 하려면 다음과 같이 출력 작업에서 작업 북마크를 활성화하고 변환 컨텍스트를 설정합니다.

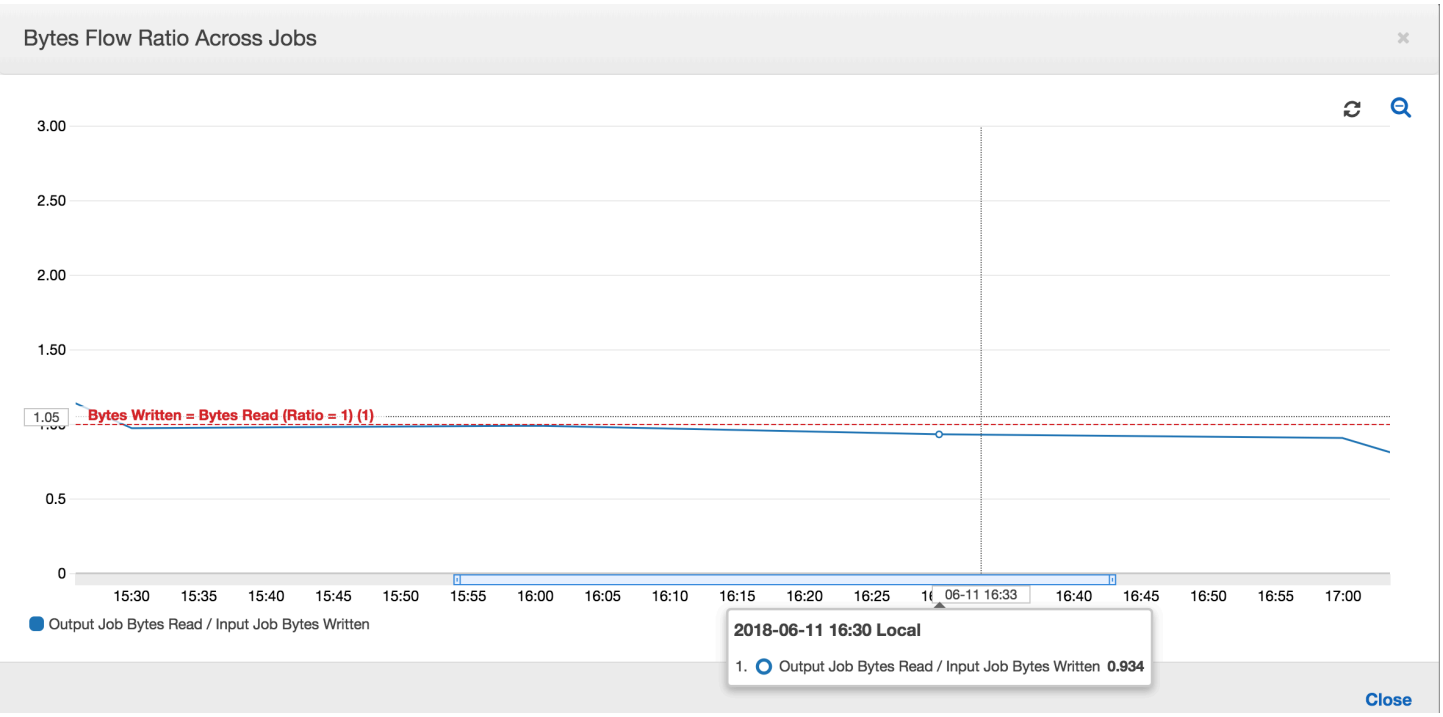
```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [staging_path],
 "useS3ListImplementation":True,"recurse":True}, format="json", transformation_ctx =
 "bookmark_ctx")
```

작업 북마크를 활성화한 상태에서는 출력 작업이 입력 작업에 대한 모든 이전 작업 실행의 스테이징 위치에 있는 데이터를 재처리하지 않습니다. 읽고 쓴 데이터를 보여주는 다음 이미지에서는 갈색 곡선하 면적이 상당히 일관되며 빨간 곡선과 비슷합니다.

추가 데이터 처리가 없으므로 바이트 흐름의 비율도 거의 1에 가깝게 유지됩니다.



다음 입력 작업 실행이 시작되기 전에 출력 작업에 대한 작업 실행이 시작되고 스테이징 위치에 있는 파일을 캡처하여 더 많은 데이터를 스테이징 위치에 넣습니다. 이 작업이 계속되는 동안은 이전 입력 작업 실행에서 캡처한 파일만 처리되며 비율이 계속 1에 가깝게 유지됩니다.



입력 작업이 예상보다 길어지며, 그 결과로서 출력 작업이 두 입력 작업 실행의 스테이징 위치에 있는 파일을 캡처한다고 가정해 보겠습니다. 그러면 해당 출력 작업 실행에 대해 비율이 1보다 높습니다. 하지만, 출력 작업의 다음 작업 실행에서는 출력 작업의 이전 작업 실행에 의해 이미 처리된 파일이 하나도 처리되지 않습니다.

DPU 용량 계획 모니터링

AWS Glue의 작업 측정치를 사용하여 AWS Glue 작업을 확장하는 데 사용할 수 있는 데이터 처리 단위(DPU) 수를 추정할 수 있습니다.

> **Note**
>
> 이 페이지는 AWS Glue 버전 0.9 및 1.0에만 적용됩니다. 최신 버전의 AWS Glue에는 용량 계획 시 추가 고려 사항을 적용하는 비용 절감 기능이 포함되어 있습니다.

주제

- 프로파일링된 코드
- AWS Glue 콘솔에서 프로파일링된 지표 시각화
- 최적 DPU 용량 결정

프로파일링된 코드

다음 스크립트는 428개의 gzip 압축된 JSON 파일을 포함하는 Amazon Simple Storage Service(Amazon S3) 파티션을 읽습니다. 이 스크립트는 필드 이름에 변경할 매핑을 적용하고 변환한 후 다시 Amazon S3에 Apache Parquet 포맷으로 씁니다. 기본값에 따라 10개 DPU를 프로비저닝하고 이 작업을 실행할 수 있습니다.

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [input_path],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [(map_spec])
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": output_path}, format =
 "parquet")
```

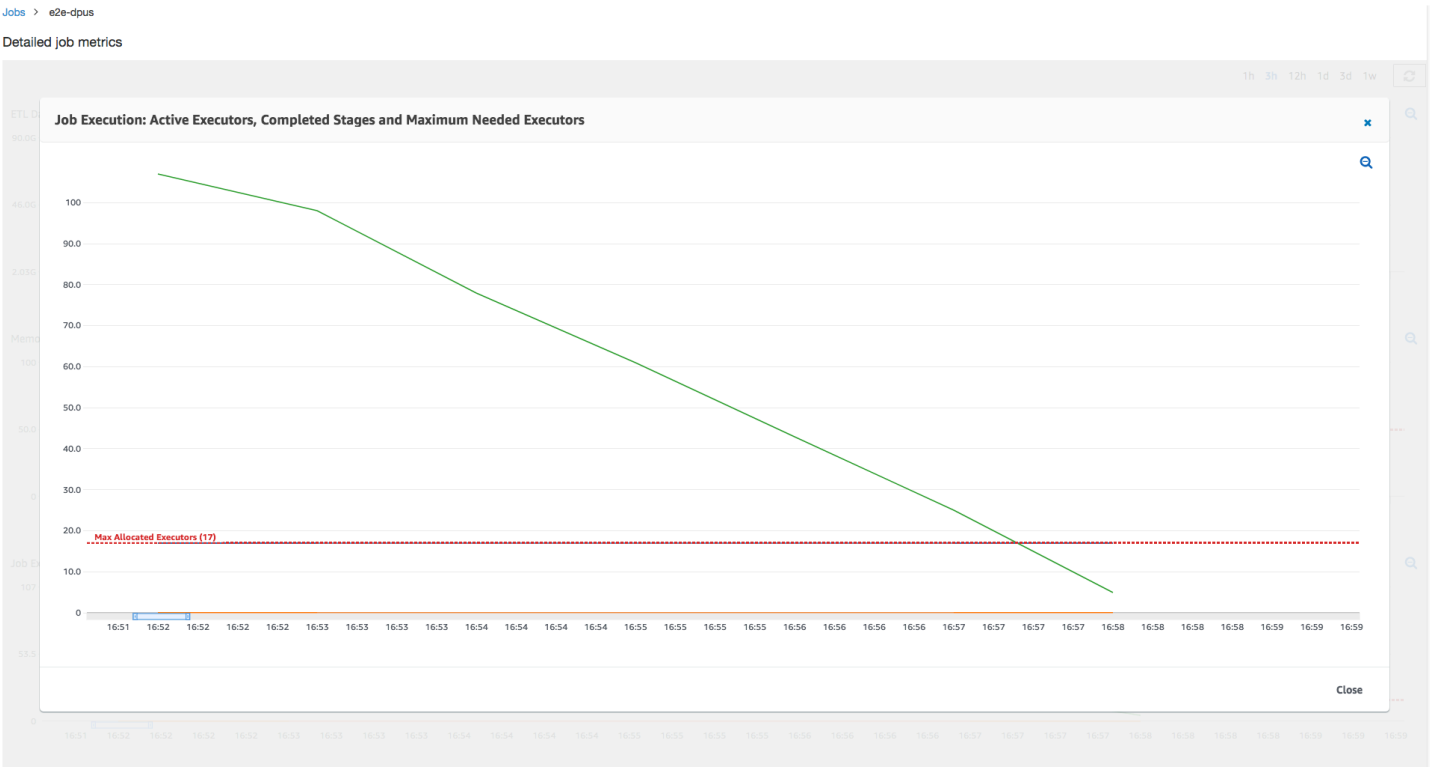
AWS Glue 콘솔에서 프로파일링된 지표 시각화

작업 실행 1: 이 작업 실행에서는 클러스터에서 부족 프로비저닝된 DPU가 있는지 알아보는 방법을 보여줍니다. AWS Glue의 작업 실행 기능은 능동적으로 실행 중인 총 실행기 수, 완료된 단계 수, 그리고 최대 필요 실행기 수를 표시합니다.

최대 필요 실행기 수는 총 실행 중인 작업 수와 보류 작업 수를 합한 다음 실행기당 작업 수로 나누어서 계산합니다. 이 결과는 현재 부하를 충족하는 데 필요한 총 실행기 수를 나타내는 측정치입니다.

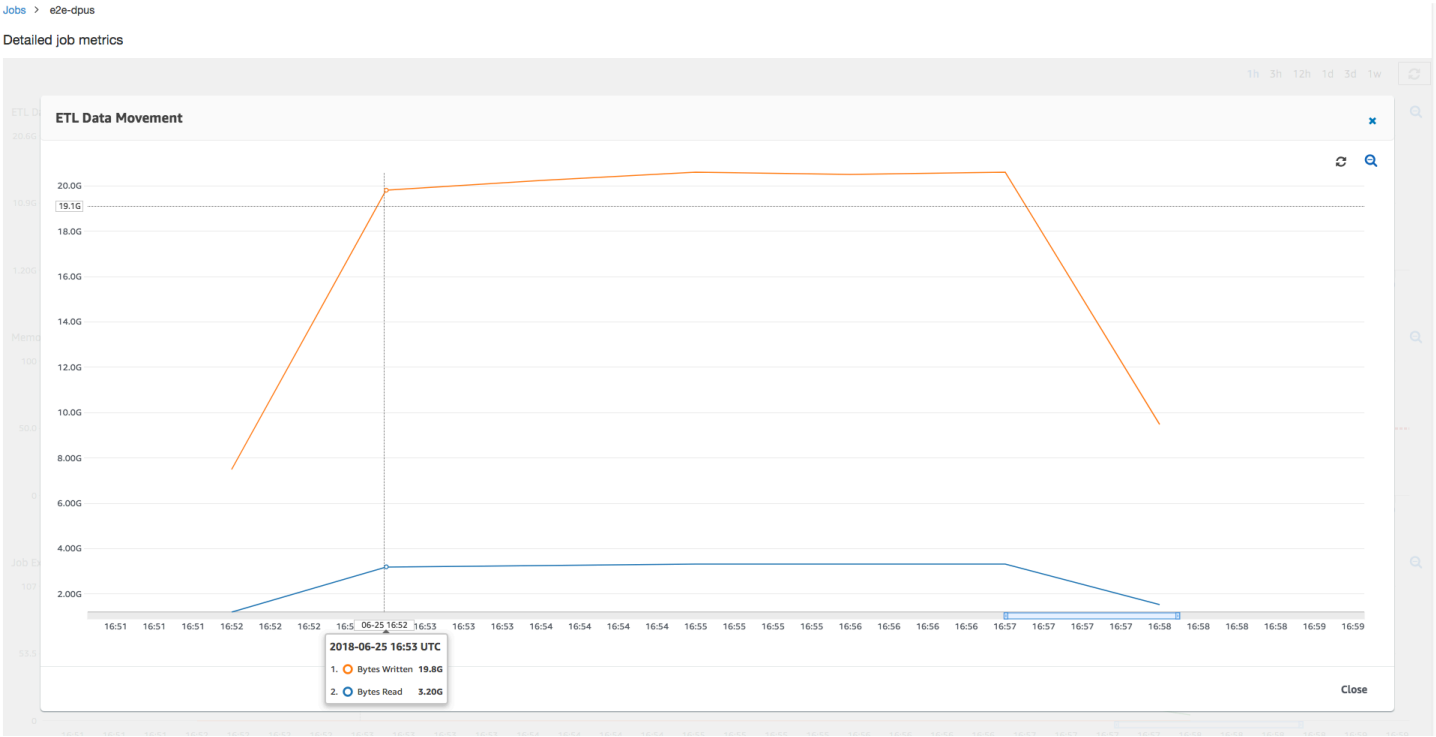
이와는 대조적으로, 능동적으로 실행 중인 실행기 수는 활성 Apache Spark 작업을 실행 중인 실행기 수를 측정합니다. 작업이 진행됨에 따라 최대 필요 실행기가 변경될 수 있으며 일반적으로 보류 작업 대기열이 축소되면서 작업 완료에 가까워질수록 줄어듭니다.

다음 그래프의 가로 빨간색 선은 최대 할당 실행기 수를 표시하며, 이는 작업에 할당하는 DPU 수에 따라 달라집니다. 이 경우, 작업 실행에 대해 10개 DPU를 할당합니다. 하나의 DPU는 관리용으로 예약되어 있습니다. 9개 DPU가 각각 실행기 두 개를 실행하며 실행기 하나가 Spark 드라이버용으로 예약되어 있습니다. Spark 드라이버는 기본 애플리케이션에서 실행됩니다. 그러므로 최대 할당 실행기 수는 2*9 - 1 = 17개입니다.



그래프가 표시될 때, 최대 필요 실행기 수는 작업 시작 시 107로 시작하지만, 활성 실행기 수는 17개로 유지됩니다. 이 개수는 DPU 수가 10개인 최대 할당 실행기 수와 동일합니다. 최대 필요 실행기 수와 최대 할당 실행기 수 간의 비율(Spark 드라이버에 대해 두 수치에 모두 1을 더함)은 부족 프로비저닝 계수를 제공합니다. 즉, 108/18 = 6x. 최대 병렬화로 실행하고 더 빨리 완료하도록 작업을 확장하기 위해 6(언더프로비저닝 비율)*9(현재 DPU 용량 - 1) + DPU 1개 = DPU 55개를 프로비저닝할 수 있습니다.

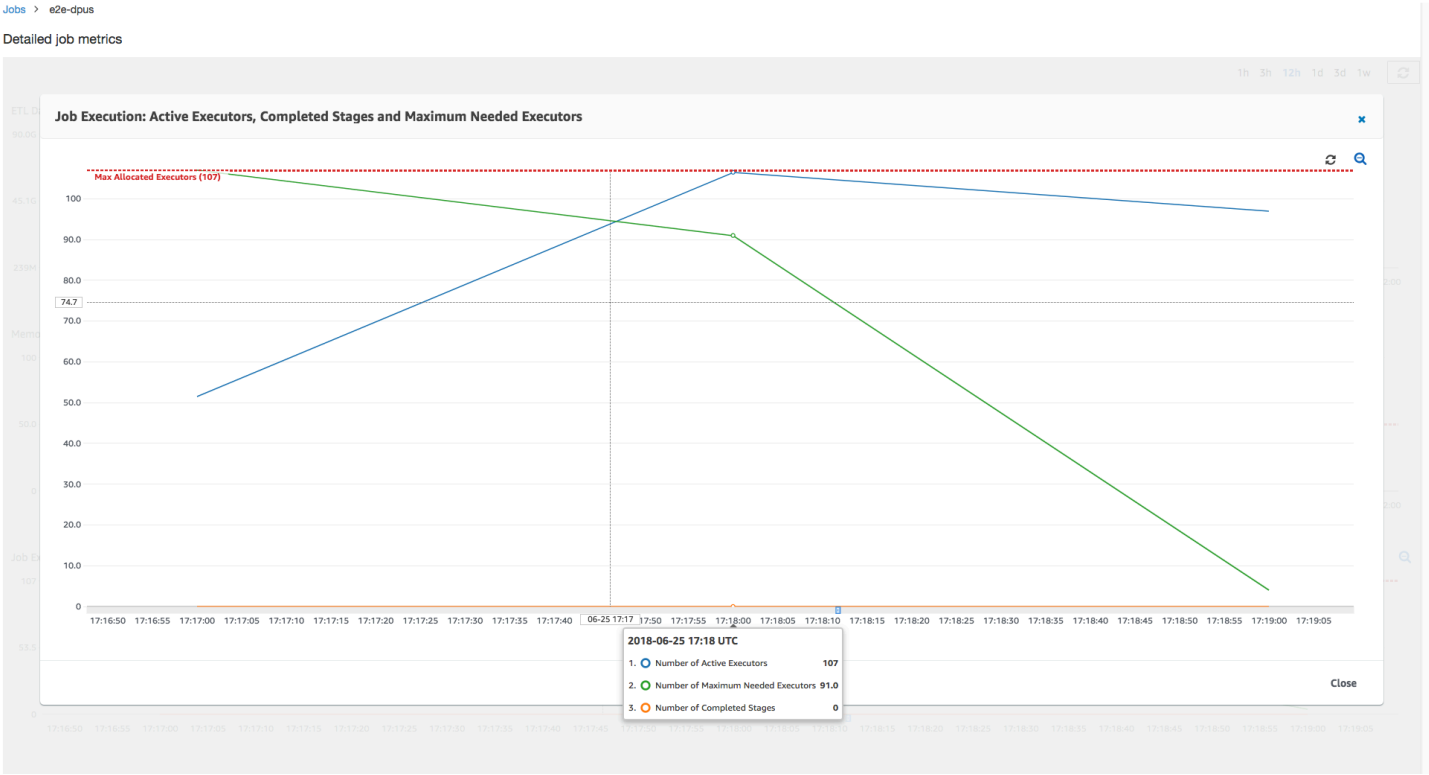
AWS Glue 콘솔은 자세한 작업 지표를 원래 할당된 최대 실행기 수를 나타내는 정적 행으로 표시합니다. 이 콘솔은 지표에 대한 작업 정의에서 할당된 최대 실행기를 계산합니다. 반면 자세한 작업 실행 지표의 경우 이 콘솔은 작업 실행 구성에서 할당된 최대 실행기, 특히 작업 실행에 대해 할당된 DPU를 계산합니다. 개별 작업 실행에 대한 지표를 보려면 작업 실행을 선택하고 View run metrics(실행 지표 보기)를 선택합니다.



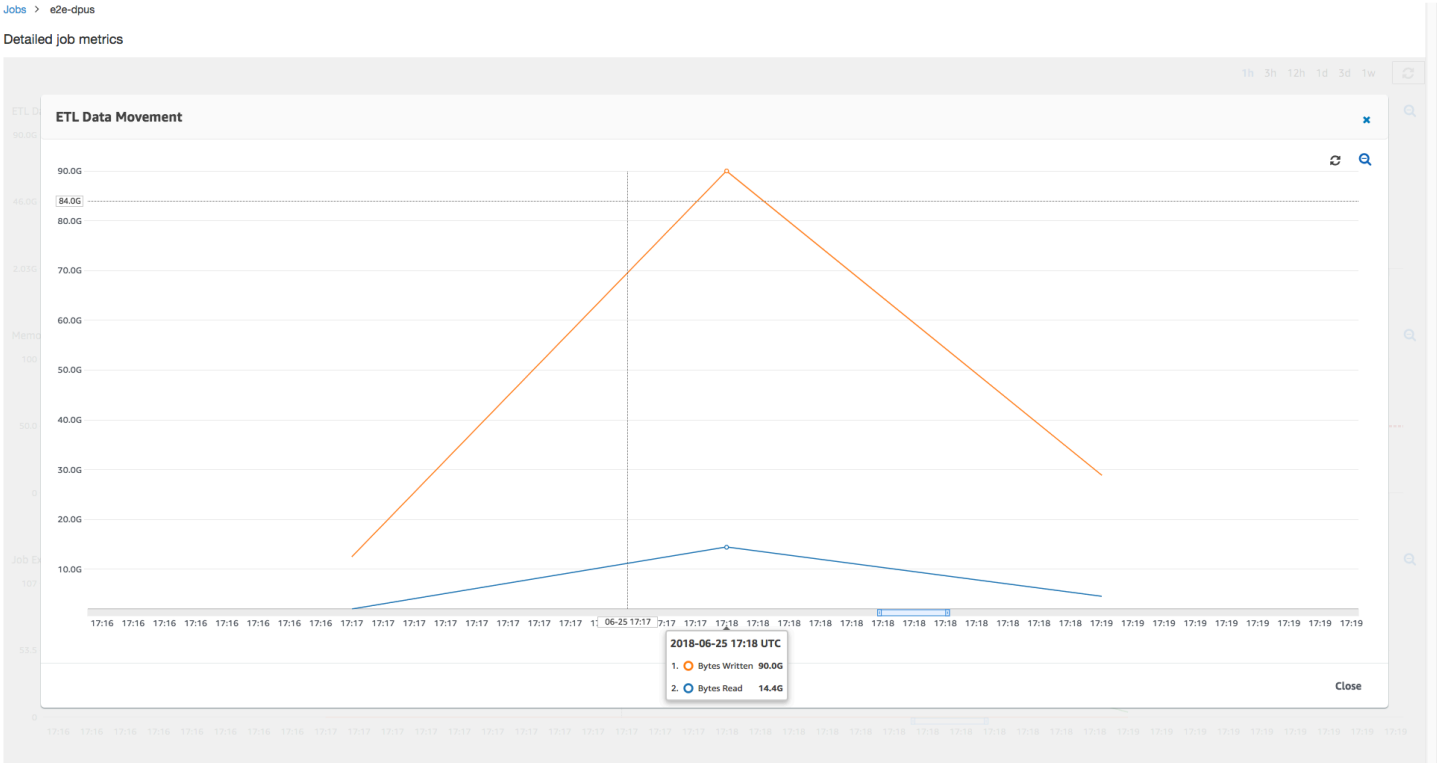
읽고 쓴 Amazon S3 바이트를 살펴보면 작업이 Amazon S3에서 데이터를 스트리밍하고 병렬로 쓰는 데 6분을 모두 소비함을 알 수 있습니다. 할당된 DPU의 모든 코어가 Amazon S3에(서) 읽고 쓰고 있습니다. 최대 필요 실행기 수(107)는 입력 Amazon S3 경로의 파일 수(428)와도 일치합니다. 각 실행기는 Spark 작업 네 개를 시작하여 네 개 입력 파일(JSON gzip 압축됨)을 처리할 수 있습니다.

최적 DPU 용량 결정

이전 작업 실행 결과에 따라, 총 할당된 DPU 수를 55로 늘리고 작업이 어떻게 수행되는지 알아볼 수 있습니다. 작업이 이전 소요 시간의 절반에 해당하는 3분 이내에 완료됩니다. 단기 실행 작업이므로 이 경우에는 작업 확장이 선형적이지 않습니다. 장기 지속 작업이나 다량의 작업을 포함하는 작업(최대 필요 실행기 수가 많음)은 선형에 가까운 DPU 확장 성능 가속화가 도움이 됩니다.



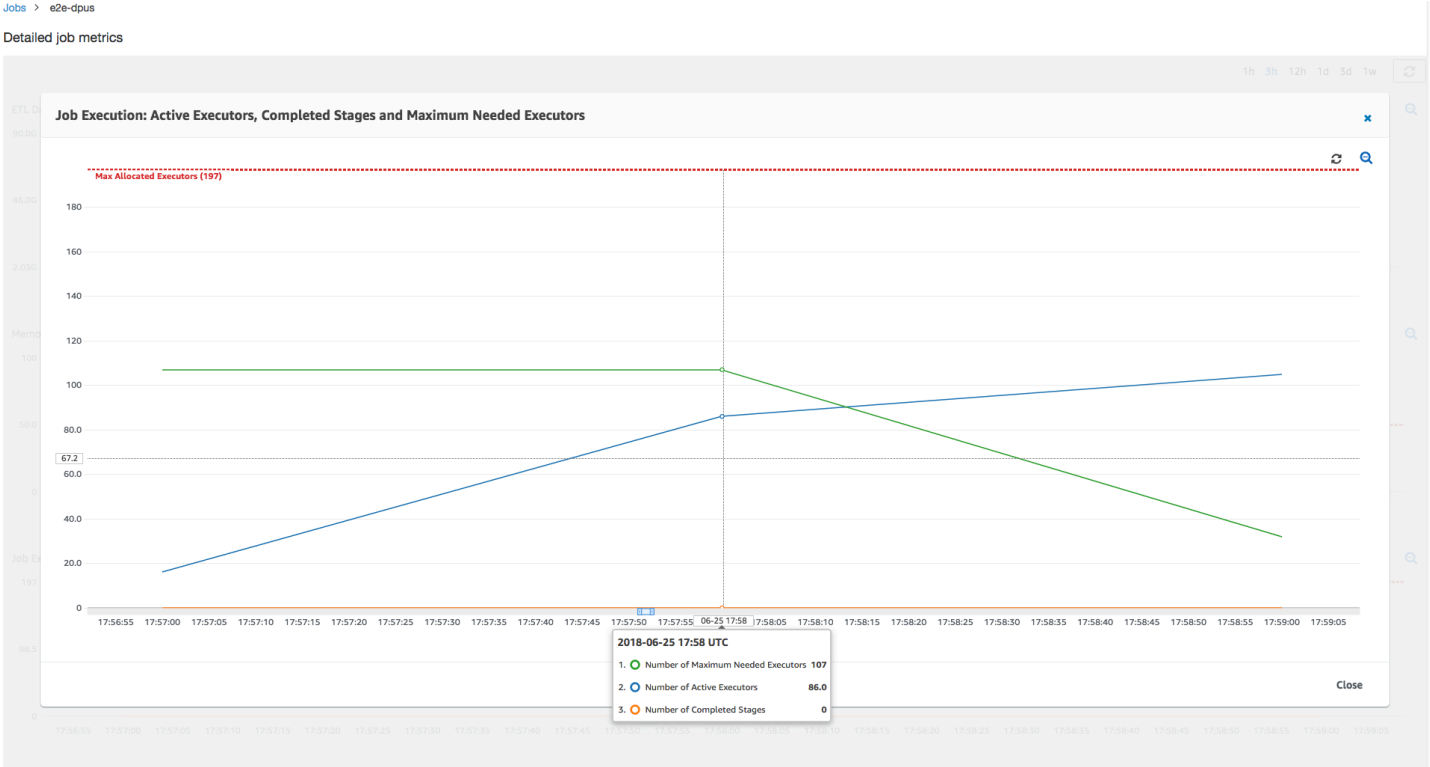
위 이미지에서 볼 수 있듯이 활성 실행기의 총 수는 할당된 최대값인 107개에 도달합니다. 마찬가지로, 최대 필요 실행기는 최대 할당 실행기 수보다 절대 높지 않습니다. 최대 필요 실행기 수는 능동적으로 실행 중인 작업 수와 보류 중인 작업 수에서 계산되므로 활성 실행기 수보다 작을 수도 있습니다. 이는 짧은 시간 동안 부분적으로 또는 전체적으로 유휴 상태이거나 아직 폐기되지 않은 실행기가 있을 수 있기 때문입니다.



이 작업 실행은 6x 이상의 실행기를 사용하여 Amazon S3에서 데이터를 동시에 읽고 씁니다. 결과적으로 이 작업은 읽기 및 쓰기에 모두 더 많은 Amazon S3 대역폭을 사용하여 더 빨리 완료합니다.

과다 프로비저닝된 DPU 식별

다음에는, DPU가 100개(99 * 2 = 198개 실행기)인 작업을 확장하는 것이 성능을 향상하는 데 도움이 되는지 여부를 결정할 수 있습니다. 다음 그래프와 같이, 작업을 완료하는 데 여전히 3분이 걸립니다. 마찬가지로, 이 작업은 실행기를 107 개 이상(55개 DPU 구성)으로 확장하지 않으며, 남은 91개 실행기가 과다 프로비저닝되고 전혀 사용되지 않습니다. 이는 최대 필요 실행기 수에서 분명히 알 수 있듯이 DPU 수를 늘리는 것이 항상 성능을 향상시키지는 않을 수도 있음을 보여줍니다.



시간차 비교

다음 표에 나와 있는 세 가지 작업 실행은 10개 DPU, 55개 DPU 및 100개 DPU에 대한 작업 실행 시간을 요약한 것입니다. 첫 번째 작업 실행을 모니터링함으로써 설정한 추정치를 사용하여 작업 실행 시간을 개선할 DPU 용량을 찾아낼 수 있습니다.

작업 ID	DPU 수	실행 시간
jr_c894524c8ef5048a4d9...	10	6분
jr_1a466cf2575e7ffe6856...	55	3분
jr_34fa1ed4c6aa9ff0a814...	100	3분

AWS Glue에서 Apache Spark에 대한 생성형 AI 문제 해결

Apache Spark를 위한 생성형 AI 문제 해결 기능의 미리 보기는 AWS Glue 4.0 및 AWS Glue 5.0에서 실행 중인 작업에서 사용할 수 있으며 다음 AWS 리전에서 지원됩니다.

- 미국 동부(버지니아 북부)

- 미국 동부(오하이오)
- 미국 서부(오리건)
- 미국 서부(캘리포니아 북부)
- 남아메리카(상파울루)
- 캐나다(중부)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(스톡홀름)
- 유럽(밀라노)
- 유럽(프랑크푸르트)
- 중동(바레인)
- 중동(UAE)
- 아프리카(케이프타운)
- 아시아 태평양(도쿄)
- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(싱가포르)
- 아시아 태평양(자카르타)
- 아시아 태평양(서울)
- 아시아 태평양(오사카)
- 아시아 태평양(시드니)

평가판 기능은 변경될 수 있습니다.

AWS Glue에서 Apache Spark 작업에 대한 생성형 AI 문제 해결은 데이터 엔지니어와 과학자는 Spark 애플리케이션에서 문제를 쉽게 진단하고 수정할 수 있는 새로운 기능입니다. 이 기능은 기계 학습 및 생성형 AI 기술을 활용하여 Spark 작업의 문제를 분석하고 이러한 문제를 해결하기 위한 실행 가능한 권장 사항과 함께 자세한 근본 원인 분석을 제공합니다.

Apache Spark에 대한 생성형 AI 문제 해결은 어떻게 작동하나요?

실패한 Spark 작업에 대해 생성형 AI 문제 해결은 작업 메타데이터와 작업의 오류 서명과 연결된 정확한 지표 및 로그를 분석하여 근본 원인 분석을 생성하고 작업 실패를 해결하는 데 도움이 되는 특정 솔루션과 모범 사례를 권장합니다.

작업과 관련해 Apache Spark에 대한 생성형 AI 문제 해결 설정

IAM 권한 구성

AWS Glue의 작업에 대해 Spark 문제 해결이 사용하는 API에 권한을 부여하려면 적절한 IAM 권한이 필요합니다. 다음과 같은 사용자 지정 AWS 정책을 IAM ID(예: 사용자, 역할 또는 그룹)에 연결하여 권한을 확보할 수 있습니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": [
                "glue:StartCompletion",
                "glue:GetCompletion"
            ],
            "Resource": [
                "arn:aws:glue:*:*:completion/*",
                "arn:aws:glue:*:*:job/*"
            ]
        }
    ]
}
```

> **Note**
>
> 평가판 중에 Spark 문제 해결에는 프로그래밍 방식으로 사용할 수 있는 AWS SDK를 통해 사용 가능한 API가 없습니다. AWS Glue Studio 콘솔을 통해 이 환경을 활성화하기 위해 IAM 정책에서 `StartCompletion` 및 `GetCompletion`과 같은 두 가지 API가 사용됩니다.

권한 할당

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

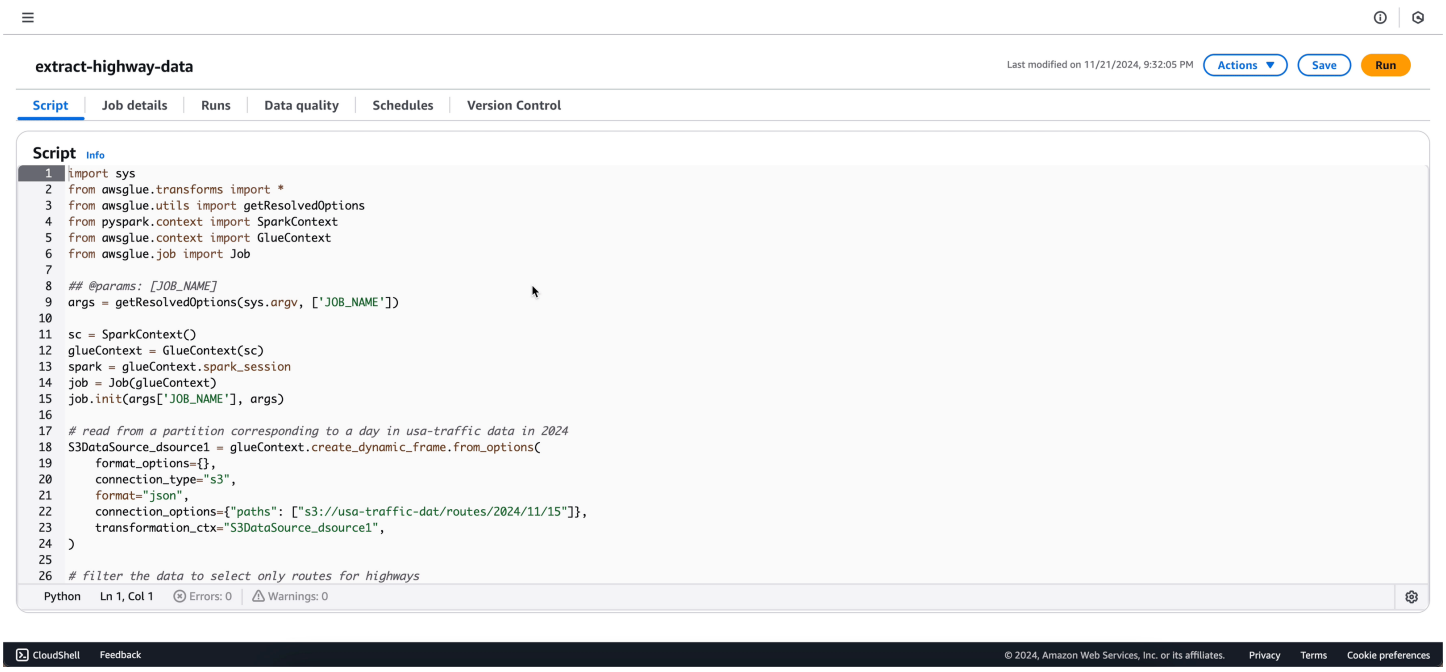
- IAM Identity Center에서 사용자 및 그룹의 경우: 권한 세트를 생성합니다. IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자의 경우: ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 타사 ID 제공업체의 역할 생성(페더레이션)의 지침을 따릅니다.
- IAM 사용자의 경우: 사용자가 수임할 수 있는 역할을 생성합니다. IAM 사용 설명서의 IAM 사용자 역할 생성의 지침을 따릅니다.

실패한 작업 실행에서 문제 해결 분석 실행

AWS Glue 콘솔의 여러 경로를 통해 문제 해결 기능에 액세스할 수 있습니다. 시작하는 방법은 다음과 같습니다.

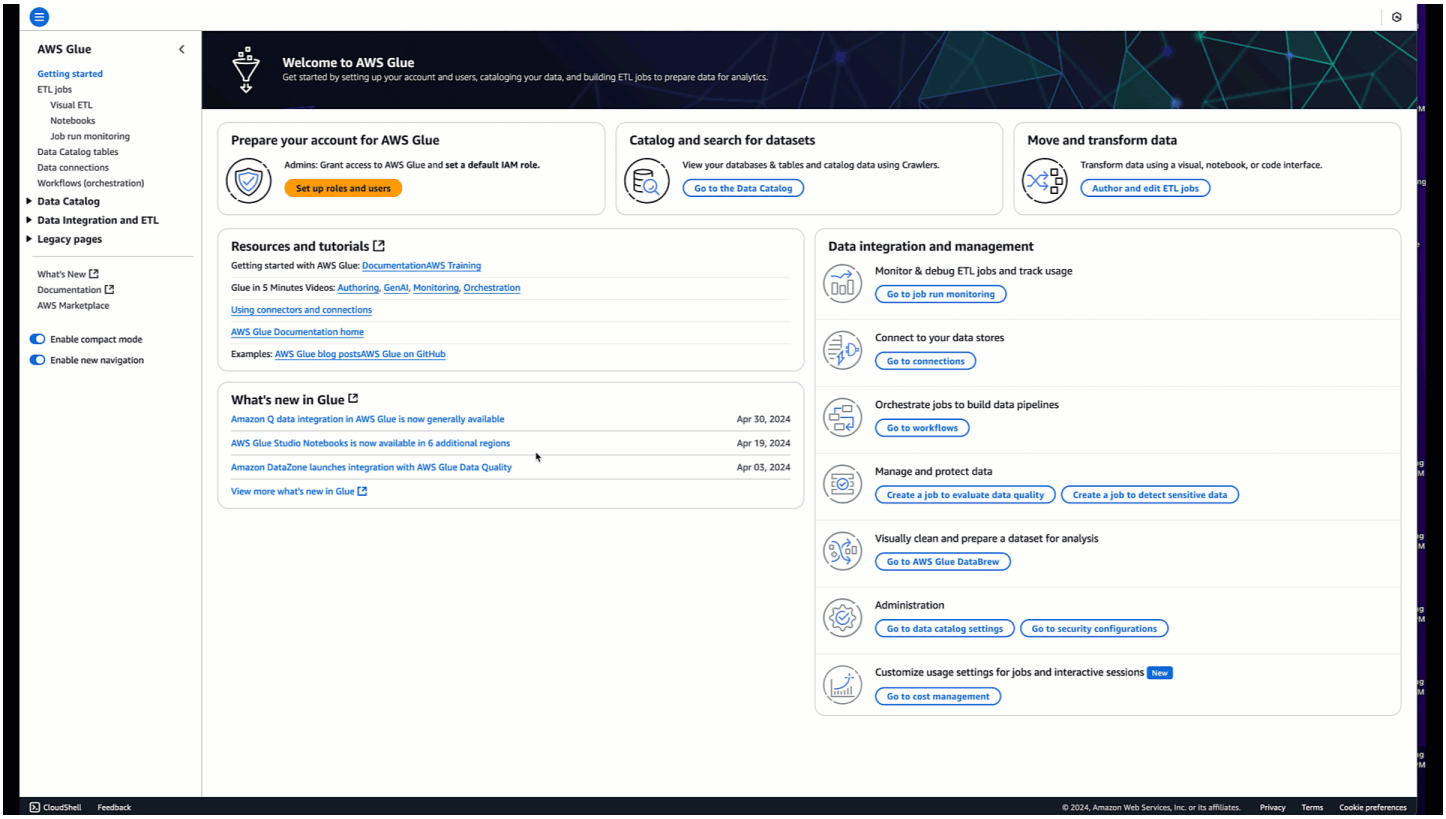
옵션 1: 작업 목록 페이지에서

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 ETL 작업을 선택하세요.
3. 작업 목록에서 실패한 작업을 찾으세요.
4. 작업 세부 정보 섹션에서 실행 탭을 선택하세요.
5. 분석하려는 실패한 작업 실행을 클릭하세요.
6. AI를 사용하여 문제 해결을 선택하여 분석을 시작하세요.
7. 문제 해결 분석이 완료되면 화면 하단의 문제 해결 분석 탭에서 근본 원인 분석 및 권장 사항을 볼 수 있습니다.



옵션 2: 작업 실행 모니터링 페이지 사용

1. 작업 실행 모니터링 페이지로 이동하세요.
2. 실패한 작업 실행을 찾으세요.
3. 드롭다운 메뉴에서 작업을 선택하세요.
4. AI를 사용하여 문제 해결을 선택하세요.



옵션 3: 작업 실행 세부 정보 페이지에서

1. 실행 탭에서 실패한 실행의 세부 정보 보기를 클릭하거나 작업 실행 모니터링 페이지에서 작업 실행을 선택하여 실패한 작업 실행의 세부 정보 페이지로 이동하세요.
2. 작업 실행 세부 정보 페이지에서 문제 해결 분석 탭을 찾으세요.

지원되는 문제 해결 카테고리(평가판)

이 서비스는 데이터 엔지니어와 개발자가 Spark 애플리케이션에서 자주 접하는 세 가지 주요 카테고리의 문제에 중점을 둡니다.

- 리소스 설정 및 액세스 오류: AWS Glue에서 Spark 애플리케이션을 실행하는 경우 리소스 설정 및 액세스 오류는 가장 흔하지만 진단하기 어려운 문제 중 하나입니다. 이러한 오류는 Spark 애플리케이션에서 AWS 리소스와 상호 작용하려고 시도하지만 권한 문제, 리소스 누락 또는 구성 문제가 나타날 때 종종 발생합니다.
- Spark 드라이버 및 실행기 메모리 문제: Apache Spark 작업에서 메모리 관련 오류는 진단 및 해결이 복잡할 수 있습니다. 이러한 오류는 데이터 처리 요구 사항이 드라이버 노드 또는 실행기 노드에서 사용 가능한 메모리 리소스를 초과할 때 종종 발생합니다.

- Spark 디스크 용량 문제: AWS Glue Spark 작업의 스토리지 관련 오류는 셔플 작업, 데이터 유출 또는 대규모 데이터 변환 처리 중에 종종 발생합니다. 이러한 오류는 한동안 작업이 실행될 때까지 명확히 드러나지 않으므로 귀중한 컴퓨팅 시간과 리소스가 낭비될 수 있어서 특히 처리하기 까다로울 수 있습니다.

> **Note**
>
> 프로덕션 환경에서 제안된 변경 사항을 구현하기 전에 제안된 변경 사항을 철저히 검토합니다. 이 서비스에서는 패턴과 모범 사례를 기반으로 권장 사항을 제공하지만 특정 사용 사례에는 추가 고려 사항이 필요할 수 있습니다.

AWS Glue에서 스트리밍 ETL 작업

지속적으로 실행되고 Amazon Kinesis Data Streams, Apache Kafka 및 Amazon Managed Streaming for Apache Kafka(Amazon MSK)와 같은 스트리밍 소스의 데이터를 사용하는 스트리밍 추출, 변환, 로드 작업을 생성할 수 있습니다. 작업은 데이터를 정리 및 변환한 다음 결과를 Amazon S3 데이터 레이크 또는 JDBC 데이터 스토어에 로드합니다.

또한 Amazon Kinesis Data Streams 스트림용 데이터를 생성할 수 있습니다. 이 기능은 AWS Glue 스크립트를 작성할 때만 사용할 수 있습니다. 자세한 내용은 the section called “Kinesis 연결” 단원을 참조하십시오.

기본적으로 AWS Glue는 100초 기간에 데이터를 처리하고 작성합니다. 이를 통해 데이터를 효율적으로 처리할 수 있으며 예상보다 늦게 도착하는 데이터에 대해 집계를 수행할 수 있습니다. 이 기간 크기를 수정하여 적시성 또는 집계 정확도를 높일 수 있습니다. AWS Glue 스트리밍 작업은 작업 북마크가 아닌 체크포인트를 사용하여 읽은 데이터를 추적합니다.

> **Note**
>
> AWS Glue는 ETL 스트리밍 작업이 실행되는 동안 시간당 요금을 청구합니다.

이 비디오에서는 스트리밍 ETL 비용 문제와 AWS Glue에서의 비용 절감 기능에 대해 설명합니다.

스트리밍 ETL 작업을 생성하려면 다음 단계를 수행합니다.

1. Apache Kafka 스트리밍 소스의 경우 Kafka 소스 또는 Amazon MSK 클러스터에 대한 AWS Glue 연결을 생성합니다.

2. 스트리밍 소스에 대한 Data Catalog 테이블을 수동으로 생성합니다.
3. 스트리밍 데이터 원본에 대한 ETL 작업을 생성합니다. 스트리밍 관련 작업 속성을 정의하고 자체 스크립트를 제공하거나 필요에 따라 생성된 스크립트를 수정합니다.

자세한 내용은 AWS Glue의 스트리밍 ETL 단원을 참조하십시오.

Amazon Kinesis Data Streams에 대한 스트리밍 ETL 작업을 생성할 때 AWS Glue 연결을 생성할 필요가 없습니다. 그러나 Kinesis Data Streams를 소스로 포함하는 AWS Glue 스트리밍 ETL 작업에 첨부된 연결이 있는 경우 Kinesis에 대한 Virtual Private Cloud(VPC) 엔드포인트가 필요합니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요. 다른 계정에서 Amazon Kinesis Data Streams 스트림을 지정할 때 교차 계정 액세스를 허용하도록 역할과 정책을 설정해야 합니다. 자세한 내용은 예: 다른 계정의 Kinesis 스트림에서 읽기를 참조하세요.

AWS Glue 스트리밍 ETL 작업은 압축된 데이터를 자동으로 감지하고, 스트리밍 데이터를 투명하게 압축 해제할 수 있으며, 입력 소스에 대해 일반적인 변환을 수행하고, 출력 저장소에 로드할 수 있습니다.

AWS Glue에서는 입력 형식에 따라 다음과 같은 압축 유형에 대해 자동 압축 해제를 지원합니다.

압축 유형	Avro 파일	Avro 데이터	JSON	CSV	Grok
bzip2	예	예	예	예	예
GZIP	No	예	예	예	예
Snappy	예(원시 Snappy)	예(프레임 처리된 Snappy)	예(프레임 처리된 Snappy)	예(프레임 처리된 Snappy)	예(프레임 처리된 Snappy)
XZ	예	예	예	예	예
ZSTD	예	아니요	아니요	아니요	No
DEFLATE	예	예	예	예	예

주제

- Apache Kafka 데이터 스트림에 대한 AWS Glue 연결 생성
- 스트리밍 소스에 대한 Data Catalog 테이블 생성
- Avro 스트리밍 소스에 대한 참고 사항 및 제한 사항
- 스트리밍 소스에 Grok 패턴 적용

- 스트리밍 ETL 작업에 대한 작업 속성 정의
- 스트리밍 ETL 참고 사항 및 제한 사항

Apache Kafka 데이터 스트림에 대한 AWS Glue 연결 생성

Apache Kafka 스트림에서 읽으려면 AWS Glue 연결을 생성해야 합니다.

Kafka 소스에 대한 AWS Glue 연결을 생성하려면(콘솔)

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 데이터 카탈로그에서 연결을 선택합니다.
3. 연결 추가를 선택하고 연결 속성 설정 페이지에서 연결 이름을 입력합니다.

   > **Note**
   >
   > 연결 속성 지정에 대한 자세한 내용은 AWS Glue 연결 속성을 참조하세요.

4. 연결 유형에서 Kafka를 선택합니다.
5. [Kafka 부트스트랩 서버 URL(Kafka bootstrap servers URLs)]에 Amazon MSK 클러스터 또는 Apache Kafka 클러스터용 부트스트랩 브로커의 호스트 및 포트 번호를 입력합니다. Kafka 클러스터에 대한 초기 연결을 설정하려면 전송 계층 보안(TLS) 엔드포인트만 사용합니다. Plaintext 엔드포인트는 지원되지 않습니다.

   다음은 Amazon MSK 클러스터의 호스트 이름 및 포트 번호 페어 목록의 예입니다.

   ```
   myserver1.kafka.us-east-1.amazonaws.com:9094,myserver2.kafka.us-
   east-1.amazonaws.com:9094,
   myserver3.kafka.us-east-1.amazonaws.com:9094
   ```

   부트스트랩 브로커 정보 가져오기에 대한 자세한 내용은 Amazon Managed Streaming for Apache Kafka Developer Guide의 Getting the Bootstrap Brokers for an Amazon MSK Cluster를 참조하세요.
6. Kafka 데이터 원본에 대한 보안 연결을 원하는 경우 [SSL 연결 필요(Require SSL connection)]를 선택하고 [Kafka 프라이빗 CA 인증서 위치(Kafka private CA certificate location)]에 사용자 정의 SSL 인증서에 대한 유효한 Amazon S3 경로를 입력합니다.

   자체 관리형 Kafka에 대한 SSL 연결의 경우 사용자 정의 인증서가 필수입니다. Amazon MSK의 경우 선택 사항입니다.

Kafka에 대한 사용자 정의 인증서 지정에 대한 자세한 내용은 the section called “SSL 연결 속성” 섹션을 참조하세요.

7. AWS Glue Studio 또는 AWS CLI를 사용하여 Kafka 클라이언트 인증 방법을 지정합니다. AWS Glue Studio에 액세스하려면 왼쪽 탐색 창의 ETL 메뉴에서 AWS Glue를 선택합니다.

   Kafka 클라이언트 인증 방법에 대한 자세한 내용은 클라이언트 인증을 위한 AWS Glue Kafka 연결 속성 단원을 참조하세요.
8. 필요에 따라 설명을 입력하고 [다음(Next)]을 선택합니다.
9. Amazon MSK 클러스터의 경우 Virtual Private Cloud (VPC), 서브넷 및 보안 그룹을 지정합니다. 자체 관리형 Kafka의 경우 VPC 정보는 선택 사항입니다.
10. [다음(Next)]을 선택하여 모든 연결 속성을 검토하고 [마침(Finish)]을 선택합니다.

AWS Glue 연결에 대한 자세한 정보는 데이터에 연결 섹션을 참조하세요.

클라이언트 인증을 위한 AWS Glue Kafka 연결 속성

SASL/GSSAPI(Kerberos) 인증

이 인증 방법을 선택하면 Kerberos 속성을 지정할 수 있습니다.

Kerberos Keytab

keytab 파일의 위치를 선택합니다. keytab은 하나 이상의 보안 주체에 대한 장기 키를 저장합니다. 자세한 내용은 MIT Kerberos Documentation: Keytab(MIT Kerberos 설명서: Keytab)을 참조하세요.

Kerberos krb5.conf 파일(Kerberos krb5.conf file)

krb5.conf 파일을 선택합니다. 여기에는 KDC 서버의 기본 영역(동일한 KDC 아래에 있는 시스템 그룹을 정의하는 도메인과 유사한 논리 네트워크)과 위치가 포함됩니다. 자세한 내용은 MIT Kerberos Documentation: krb5.conf(MIT Kerberos 설명서: krb5.conf)를 참조하세요.

Kerberos 보안 주체 및 Kerberos 서비스 이름(Kerberos principal and Kerberos service name)

Kerberos 보안 주체 및 서비스 이름을 입력합니다. 자세한 내용은 MIT Kerberos 설명서: Kerberos 보안 주체(MIT Kerberos Documentation: Kerberos principal)을 참조하세요.

SASL/SCRAM-SHA-512 인증

이 인증 방법을 선택하면 인증 자격 증명을 지정할 수 있습니다.

AWS Secrets Manager

이름 또는 ARN을 입력하여 검색 상자에서 토큰을 검색합니다.

사용자 이름 및 암호 직접 제공(Provide username and password directly)

이름 또는 ARN을 입력하여 검색 상자에서 토큰을 검색합니다.

SSL 클라이언트 인증

이 인증 방법을 선택하면 Amazon S3를 검색하여 Kafka 클라이언트 키 스토어의 위치를 선택할 수 있습니다. 선택 사항으로 Kafka 클라이언트 키 스토어 암호와 Kafka 클라이언트 키 암호를 입력할 수 있습니다.

IAM 인증.

이 인증 방법은 추가 사양이 필요하지 않으며 스트리밍 소스가 MSK Kafka인 경우에만 적용됩니다.

SASL/PLAIN 인증

이 인증 방법을 선택하면 인증 자격 증명을 지정할 수 있습니다.

스트리밍 소스에 대한 Data Catalog 테이블 생성

데이터 스키마를 포함한 소스 데이터 스트림 속성을 지정하는 데이터 카탈로그 테이블을 스트리밍 소스에 대해 수동으로 생성할 수 있습니다. 이 테이블은 스트리밍 ETL 작업에 대한 데이터 원본으로 사용됩니다.

원본 데이터 스트림에 있는 데이터의 스키마를 모르는 경우 스키마 없이 테이블을 생성할 수 있습니다. 그런 다음 스트리밍 ETL 작업을 생성할 때 AWS Glue 스키마 감지 기능을 설정할 수 있습니다. AWS Glue는 스트리밍 데이터에서 스키마를 결정합니다.

AWS Glue 콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS Glue API를 사용하여 테이블을 생성합니다. AWS Glue 콘솔을 사용하여 수동으로 테이블을 생성하는 방법에 대한 자세한 내용은 the section called “테이블 생성” 단원을 참조하십시오.

> **Note**
>
> AWS Lake Formation 콘솔을 사용하여 테이블을 생성할 수 없습니다. AWS Glue 콘솔을 사용해야 합니다.

또한 Avro 포맷의 스트리밍 소스 또는 Grok 패턴을 적용할 수 있는 로그 데이터에 대한 다음 정보를 고려합니다.

- the section called “Avro 스트리밍 소스에 대한 참고 사항 및 제한 사항”

- the section called “스트리밍 소스에 Grok 패턴 적용”

주제

- Kinesis 데이터 원본
- Kafka 데이터 원본
- AWS Glue Schema Registry 테이블 소스

Kinesis 데이터 원본

테이블 생성 시 다음 스트리밍 ETL 속성(콘솔)을 설정합니다.

소스 유형

Kinesis

동일한 계정에 있는 Kinesis 소스의 경우

리전

Amazon Kinesis Data Streams 서비스가 상주하는 AWS 리전입니다. 리전 및 Kinesis 스트림 이름은 함께 스트림 ARN으로 변환됩니다.

예: https://kinesis.us-east-1.amazonaws.com

Kinesis 스트림 이름

Amazon Kinesis Data Streams Developer Guide의 Creating a Stream에 설명된 스트림 이름입니다.

다른 계정의 Kinesis 소스의 경우 이 예를 참조하여 교차 계정 액세스를 허용하도록 역할과 정책을 설정합니다. 다음 설정을 구성합니다.

스트림 ARN

소비자가 등록된 Kinesis 데이터 스트림의 ARN입니다. 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스를 참조하세요.

수임된 역할 ARN

수임할 역할의 Amazon 리소스 이름(ARN)입니다.

세션 이름(선택 사항)

수임한 역할 세션의 식별자입니다.

역할 세션 이름을 사용하여 다른 보안 주체가 동일한 역할을 수임할 때 또는 다른 이유로 세션을 고유하게 식별합니다. 교차 계정 시나리오에서는 역할을 소유하는 계정이 역할 세션 이름을 보고 로그할 수 있습니다. 역할 세션 이름은 수임한 역할 보안 주체의 ARN에도 사용됩니다. 즉, 임시 보안 자격 증명을 사용하는 후속 교차 계정 API 요청은 역할 세션 이름을 AWS CloudTrail 로그의 외부 계정에 노출합니다.

Amazon Kinesis Data Streams(AWS Glue API 또는 AWS CLI)에 대한 스트리밍 ETL 속성을 설정하려면

- 동일한 계정에서 Kinesis 소스에 대한 스트리밍 ETL 속성을 설정하려면 `CreateTable` API 작업 또는 `create_table` CLI 명령의 `StorageDescriptor` 구조에서 `streamName` 및 `endpointUrl` 파라미터를 지정합니다.

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamName": "sample-stream",
  "endpointUrl": "https://kinesis.us-east-1.amazonaws.com"
 }
 ...
}
```

또는 `streamARN`을 지정합니다.

Example

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamARN": "arn:aws:kinesis:us-east-1:123456789:stream/sample-stream"
 }
 ...
}
```

- 다른 계정에서 Kinesis 소스에 대한 스트리밍 ETL 속성을 설정하려면 `CreateTable` API 작업 또는 `create_table` CLI 명령의 `StorageDescriptor` 구조에서 `streamARN`, `awsSTSRoleARN` 및 `awsSTSSessionName`(선택 사항) 파라미터를 지정합니다.

```
"StorageDescriptor": {
```

```
 "Parameters": {
  "typeOfData": "kinesis",
  "streamARN": "arn:aws:kinesis:us-east-1:123456789:stream/sample-stream",
  "awsSTSRoleARN": "arn:aws:iam::123456789:role/sample-assume-role-arn",
  "awsSTSSessionName": "optional-session"
 }
 ...
}
```

Kafka 데이터 원본

테이블 생성 시 다음 스트리밍 ETL 속성(콘솔)을 설정합니다.

소스 유형

Kafka

Kafka 소스의 경우:

주제 이름

Kafka에 지정된 주제 이름입니다.

연결

the section called “Kafka 데이터 스트림에 대한 연결 생성”에 설명된 대로 Kafka 소스를 참조하는 AWS Glue 연결입니다.

AWS Glue Schema Registry 테이블 소스

AWS Glue Schema Registry를 스트리밍 작업에 사용하려면 사용 사례: AWS Glue Data Catalog에 있는 지침에 따라 Schema Registry 테이블을 생성하거나 업데이트합니다.

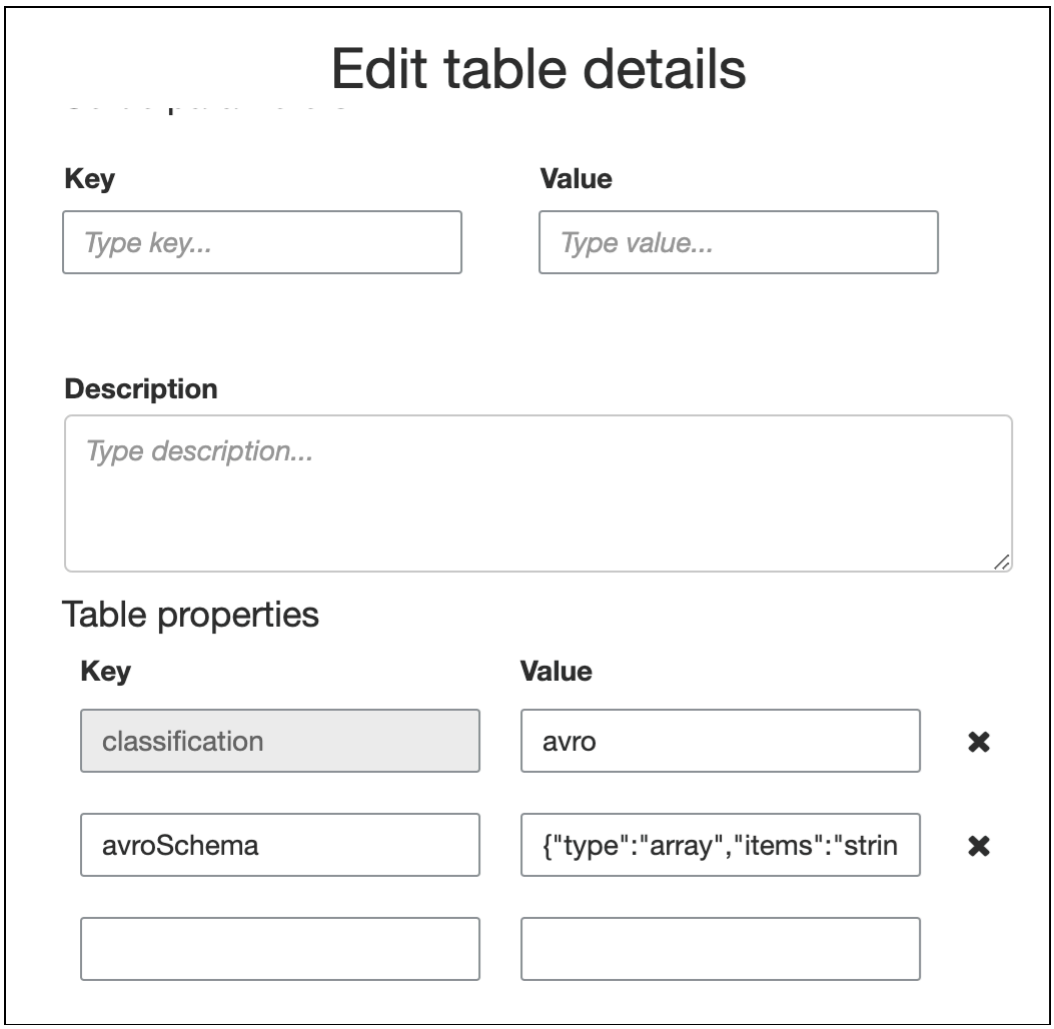
현재 AWS Glue 스트리밍은 스키마 추론이 `false`로 설정된 Glue Schema Registry Avro 포맷만 지원합니다.

Avro 스트리밍 소스에 대한 참고 사항 및 제한 사항

Avro 포맷의 스트리밍 소스에는 다음 참고 및 제한 사항이 적용됩니다.

- 스키마 감지가 설정되어 있으면 Avro 스키마가 페이로드에 포함되어야 합니다. 해제되어 있으면 데이터만 페이로드에 포함되어야 합니다.

- 일부 Avro 데이터 유형은 동적 프레임에서 지원되지 않습니다. AWS Glue 콘솔의 테이블 생성 마법사에서 [스키마 정의(Define a schema)] 페이지를 사용하여 스키마를 정의할 때 이러한 데이터 유형을 지정할 수 없습니다. 스키마 감지 중에 Avro 스키마에서 지원되지 않는 유형은 다음과 같이 지원되는 유형으로 변환됩니다.
  - `EnumType => StringType`
  - `FixedType => BinaryType`
  - `UnionType => StructType`
- 콘솔의 [스키마 정의(Define a schema)] 페이지를 사용하여 테이블 스키마를 정의하는 경우 스키마에 대한 암시적 루트 요소 유형은 `record`입니다. `record` 이외의 루트 요소 유형(예: `array` 또는 `map`)을 원하는 경우 [스키마 정의(Define a schema)] 페이지를 사용하여 스키마를 지정할 수 없습니다. 대신 해당 페이지를 건너뛰고 테이블 속성으로 또는 ETL 스크립트 내에서 스키마를 지정해야 합니다.
  - 테이블 속성에서 스키마를 지정하려면 테이블 생성 마법사를 완료하고 테이블 세부 정보를 편집한 다음 [테이블 속성(Table properties)]에서 새 키-값 페어를 추가합니다. 다음 스크린샷과 같이 키 `avroSchema`를 사용하고 값에 대한 스키마 JSON 객체를 입력합니다.



- ETL 스크립트에서 스키마를 지정하려면 다음 Python 및 Scala 예제와 같이 `datasource0` 할당 문을 수정하고 `avroSchema` 키를 `additional_options` 인수에 추가합니다.

  Python

  ```
  SCHEMA_STRING = '{"type":"array","items":"string"}'
  datasource0 = glueContext.create_data_frame.from_catalog(database =
   "database", table_name = "table_name", transformation_ctx = "datasource0",
   additional_options = {"startingPosition": "TRIM_HORIZON", "inferSchema":
   "false", "avroSchema": SCHEMA_STRING})
  ```

  Scala

  ```
  val SCHEMA_STRING = """{"type":"array","items":"string"}"""
  val datasource0 = glueContext.getCatalogSource(database = "database", tableName
   = "table_name", redshiftTmpDir = "", transformationContext = "datasource0",
  ```

```
additionalOptions = JsonOptions(s"""{"startingPosition": "TRIM_HORIZON",
"inferSchema": "false", "avroSchema":"$SCHEMA_STRING"}""")).getDataFrame()
```

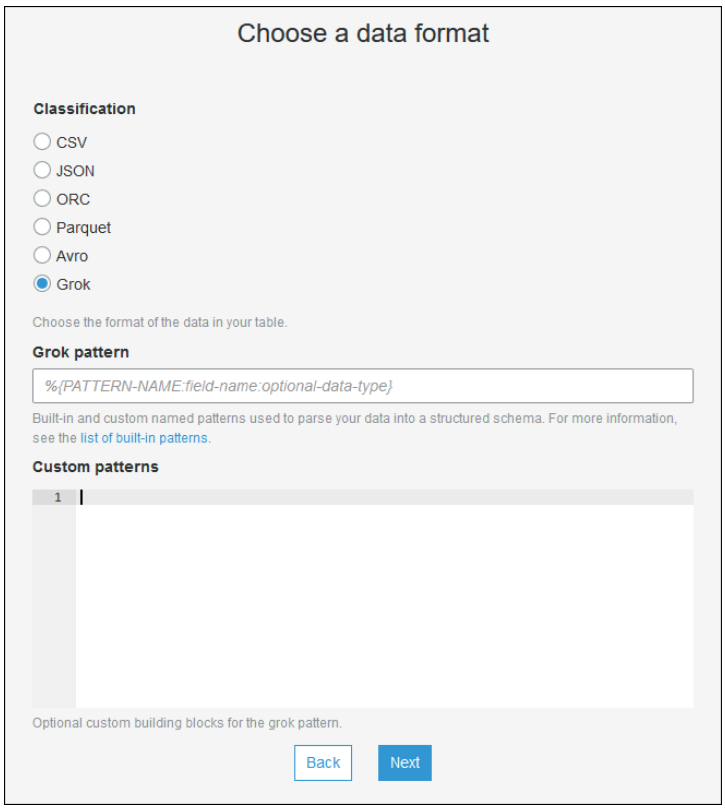
스트리밍 소스에 Grok 패턴 적용

로그 데이터 원본에 대한 스트리밍 ETL 작업을 생성하고 Grok 패턴을 사용하여 로그를 정형 데이터로 변환할 수 있습니다. 그러면 ETL 작업은 데이터를 정형 데이터 원본으로 처리합니다. 스트리밍 원본에 대한 Data Catalog 테이블을 생성할 때 적용할 Grok 패턴을 지정합니다.

Grok 패턴 및 사용자 정의 패턴 문자열 값에 대한 자세한 내용은 Grok 사용자 지정 분류자 작성 섹션을 참조하세요.

Data Catalog 테이블에 Grok 패턴 추가(콘솔)

- 테이블 생성 마법사를 사용하여 the section called “스트리밍 소스에 대한 Data Catalog 테이블 생성”에 지정된 파라미터로 테이블을 생성합니다. 데이터 포맷을 Grok로 지정하고 [Grok 패턴(Grok pattern)] 필드를 채우고 필요에 따라 [사용자 정의 패턴(선택 사항)(Custom patterns (optional))] 아래에 사용자 정의 패턴을 추가합니다.



  각 사용자 정의 패턴 다음에 Enter 키를 누릅니다.

Data Catalog 테이블에 Grok 패턴 추가(AWS Glue API 또는 AWS CLI)

- `CreateTable` API 작업 또는 `create_table` CLI 명령에 `GrokPattern` 파라미터와 `CustomPatterns` 파라미터(선택 사항)를 추가합니다.

```
 "Parameters": {
...
    "grokPattern": "string",
    "grokCustomPatterns": "string",
...
},
```

`grokCustomPatterns`를 문자열로 표현하고 패턴 사이의 구분 기호로 "\n"을 사용합니다.

다음은 이러한 파라미터를 지정하는 예입니다.

Example

```
"parameters": {
...
    "grokPattern": "%{USERNAME:username} %{DIGIT:digit:int}",
    "grokCustomPatterns": "digit \d",
...
}
```

스트리밍 ETL 작업에 대한 작업 속성 정의

AWS Glue 콘솔에서 스트리밍 ETL 작업을 정의할 때 다음 스트림 관련 속성을 제공합니다. 추가 작업 속성에 대한 설명은 Spark 작업에 대한 작업 속성 정의 단원을 참조하십시오.

IAM 역할

작업을 실행하고, 스트리밍 소스에 액세스하고, 대상 데이터 스토어에 액세스하는 데 사용되는 리소스에 대한 권한 부여에 사용되는 AWS Identity and Access Management(IAM) 역할을 지정합니다.

Amazon Kinesis Data Streams에 액세스하려면 `AmazonKinesisFullAccess` AWS 관리형 정책을 역할에 연결하거나 보다 세분화된 액세스를 허용하는 유사한 IAM 정책을 연결합니다. 샘플 정책은 IAM을 사용하여 Amazon Kinesis Data Streams 리소스에 대한 액세스 제어를 참조하십시오.

AWS Glue 작업을 실행하는 권한에 대한 자세한 내용은 AWS Glue의 Identity and Access Management 단원을 참조하십시오.

유형

Spark Streaming을 선택합니다.

AWS Glue 버전

AWS Glue 버전에 따라 작업에 사용할 수 있는 Apache Spark와 Python 또는 Scala의 버전이 정해집니다. 작업에 사용 가능한 Python 또는 Scala 버전을 지정하는 선택 항목을 선택합니다. AWS Glue Python 3을 지원하는 버전 2.0은 스트리밍 ETL 작업의 기본값입니다.

유지보수 윈도우

스트리밍 작업을 다시 시작할 수 있는 기간을 지정합니다. the section called “유지 관리 기간” 섹션을 참조하세요.

작업 제한 시간

필요에 따라 기간(분)을 입력합니다. 기본값은 빈 상태입니다.

- 스트리밍 작업의 제한 시간 값은 7일 또는 10,080분 미만이어야 합니다.
- 값을 비워 두면 유지 관리 기간을 설정하지 않은 경우 7일 후에 작업이 다시 시작됩니다. 유지 관리 기간을 설정한 경우 7일 후 유지 관리 기간에 작업이 다시 시작됩니다.

데이터 소스

the section called “스트리밍 소스에 대한 Data Catalog 테이블 생성”에 생성된 테이블을 지정합니다.

데이터 대상

다음 중 하나를 수행합니다.

- 데이터 대상의 테이블 생성을 선택하고 다음 데이터 대상 속성을 지정합니다.

  데이터 스토어

  Amazon S3 또는 JDBC를 선택합니다.

  형식

  원하는 포맷을 선택합니다. 스트리밍에 모두 지원됩니다.

- [데이터 카탈로그의 테이블 사용 및 데이터 대상 업데이트(Use tables in the data catalog and update your data target)]를 선택하고 JDBC 데이터 스토어에 대한 테이블을 선택합니다.

출력 스키마 정의

다음 중 하나를 수행합니다.

- [각 레코드의 스키마 자동 감지(Automatically detect schema of each record)]를 선택하여 스키마 감지를 설정합니다. AWS Glue는 스트리밍 데이터에서 스키마를 결정합니다.
- [모든 레코드에 대한 출력 스키마 지정(Specify output schema for all records)]을 선택하여 매핑 적용 변환을 사용하여 출력 스키마를 정의합니다.

Script

필요에 따라 자체 스크립트를 제공하거나 생성된 스크립트를 수정하여 Apache Spark Structured Streaming 엔진이 지원하는 작업을 수행합니다. 사용 가능한 작업에 대한 자세한 내용은 스트리밍 데이터 프레임/데이터 세트에 대한 작업을 참조하십시오.

스트리밍 ETL 참고 사항 및 제한 사항

다음 참고 사항 및 제한 사항에 유의하십시오.

- AWS Glue 스트리밍 ETL 작업에 대한 자동 압축 해제는 지원되는 압축 유형에 대해서만 사용할 수 있습니다. 또한 다음을 참조하세요.
  - 프레임 처리된 Snappy는 Snappy용 공식 프레이밍 형식을 나타냅니다.
  - Deflate는 Glue 버전 2.0이 아닌 Glue 버전 3.0에서 지원됩니다.
- 스키마 감지를 사용하는 경우 스트리밍 데이터의 조인을 수행할 수 없습니다.
- AWS Glue 스트리밍 ETL 작업은 Avro 형식의 AWS Glue 스키마 레지스트리에 대한 Union 데이터 유형을 지원하지 않습니다.
- ETL 스크립트는 AWS Glue의 기본 제공 변환과 Apache Spark Structured Streaming에 대한 기본 변환을 사용할 수 있습니다. 자세한 내용은 Apache Spark 웹 사이트의 Operations on streaming DataFrames/Datasets 또는 AWS Glue PySpark 변환 참조 섹션을 참조하세요.
- AWS Glue 스트리밍 ETL 작업은 체크포인트를 사용하여 읽은 데이터를 추적합니다. 따라서 중지되고 다시 시작된 작업은 스트림에서 중단된 부분부터 다시 시작됩니다. 데이터를 다시 처리하려는 경우 스크립트에서 참조된 체크포인트 폴더를 삭제할 수 있습니다.
- 작업 북마크는 지원되지 않습니다.
- Kinesis Data Streams의 향상된 팬아웃 기능을 사용하려면 the section called “Kinesis 스트리밍 작업에서 향상된 팬아웃 사용” 섹션을 참조하세요.
- AWS Glue Schema Registry에서 생성된 Data Catalog 테이블을 사용하는 경우 새 스키마 버전을 사용할 수 있게 되면 새 스키마를 반영하기 위해 다음을 수행해야 합니다.

1. 테이블과 연결된 작업을 중지합니다.
2. Data Catalog 테이블의 스키마를 업데이트합니다.
3. 테이블과 연결된 작업을 다시 시작합니다.

AWS Lake Formation FindMatches로 레코드 매칭

> **Note**
>
> 현재 레코드 일치는 중동(UAE), 유럽(스페인), 아시아 태평양(자카르타) 및 유럽(취리히) 리전의 AWS Glue 콘솔에서 사용할 수 없습니다.

AWS Lake Formation에는 기계 학습 기능이 있어서 사용자 지정 변환을 만들어 데이터를 정리할 수 있습니다. 여기에는 FindMatches로 명명된 사용 가능 변환이 하나 있습니다. FindMatches 변환으로는 레코드에 공통된 고유 식별자가 없고 정확히 일치되는 필드 또한 없을 경우에도 데이터 세트에서 중복 레코드나 일치 레코드를 식별할 수 있습니다. 여기에서는 특정 코드 작성이나 기계 학습 진행 방법 숙지가 불필요합니다. FindMatches는 다음과 같은 다양한 문제에서 도움이 될 수 있습니다.

- 고객 매칭(Matching customers): 데이터베이스상에서 여러 고객 필드가 서로 정확히 일치하지 않을 때에도(예: 이름의 철자나 주소가 다른 경우, 데이터가 누락되거나 부정확한 경우 등) 고객의 레코드를 다양한 고객 데이터베이스에 연결할 수 있습니다.
- 제품 매칭(Matching products): 항목의 구조가 서로 다를 때에도 사용자 카탈로그의 제품과 다른 출처를 매칭할 수 있습니다(예: 경쟁사의 카탈로그와 매칭).
- 부정 행위 탐지 개선(Improving fraud detection): 중복된 고객 계정을 식별하고 새로 생성된 계정이 이전의 부정 행위 사용자와 일치하거나 일치할 소지가 있을 때 이를 확인할 수 있습니다.
- 기타 매칭 문제(Other matching problems): 주소나 영화, 파트 목록 등을 매칭할 수 있습니다. 일반적으로 사람이 데이터베이스의 행을 확인해서 일치 여부를 확인할 수 있다면 FindMatches 변환으로 업무에 도움이 될 만한 상황이 매우 많습니다.

작업을 생성할 때 이러한 변환을 만들 수 있습니다. 생성하는 변환은 레이블을 지정한 소스 데이터 세트의 예제 데이터 및 소스 데이터 스토어 스키마를 기반으로 합니다(이 프로세스를 변환 '교육'이라고 함). 레이블을 지정하는 레코드가 소스 데이터 세트에 있어야 합니다. 이러한 프로세스에서는 사용자가 레이블 지정하는 파일을 생성한 다음 변환이 어느 정도 학습이 가능한 사항을 다시 업로드합니다. 변환을 학습시킨 다음 Spark 기반 AWS Glue 작업(PySpark 또는 Scala Spark)에서 호출하고 원본 데이터 스토어와 함께 다른 스크립트에 사용할 수 있습니다.

생성된 변환은 AWS Glue에 저장됩니다. AWS Glue 콘솔에서 생성된 변환을 관리할 수 있습니다. 데이터 통합 및 ETL, 데이터 분류 도구 > 레코드 일치 아래의 탐색 창에서 기계 학습 변환을 편집하고 계속 가르칠 수 있습니다. 콘솔에서 변환을 관리하는 방법에 대한 자세한 내용은 기계 학습 변환 작업 단원을 참조하십시오.

> **Note**
>
> AWS Glue 버전 2.0 FindMatches 작업은 변환이 데이터를 처리하는 동안 Amazon S3 버킷 `aws-glue-temp-<accountID>-<region>`을 사용하여 임시 파일을 저장합니다. 실행이 완료된 후 수동으로 또는 Amazon S3 수명 주기 규칙을 설정하여 이 데이터를 삭제할 수 있습니다.

기계 학습 변환의 유형

기계 학습 변환을 만들어 데이터를 정리할 수 있습니다. 이런 변환을 ETL 스크립트에서 호출할 수 있습니다. 데이터가 DynamicFrame이라는 데이터 구조 내의 변환에서 변환으로 전달됩니다. 이는 Apache Spark SQL `DataFrame`을 확장한 것입니다. `DynamicFrame`은 데이터를 포함하고 데이터 스키마를 참조하여 데이터를 진행합니다.

다음과 같은 유형의 기계 학습 변환을 사용할 수 있습니다.

일치 항목 찾기

원본 데이터에서 중복 레코드를 찾습니다. 예제 데이터 세트에 일치하는 행을 나타내는 레이블을 지정하여 이 기계 학습 변환을 학습시킵니다. 레이블 지정된 예제 데이터로 학습을 많이 시킬수록 기계 학습 변환이 어떤 행을 매칭해야 하는지 배우게 됩니다. 변환의 구성 방식에 따라 다음 중 하나가 출력됩니다.

- 일치하는 레코드 세트를 나타내는 값이 채워진 `match_id` 열이 추가된 입력 테이블 사본 `match_id` 열은 임의 식별자입니다. `match_id`가 동일한 레코드는 서로 일치한다고 식별되었습니다. `match_id`가 서로 다른 레코드는 일치하지 않습니다.
- 중복 행이 제거된 입력 테이블 사본 중복 항목이 여러 개 발견되면 기본 키가 가장 낮은 레코드를 유지합니다.

증분 일치 항목 찾기

기존 프레임과 증분 프레임에서 일치 항목을 찾고 일치 그룹당 고유 ID를 포함하는 열을 출력으로 반환하도록 일치 항목 찾기 변환을 구성할 수도 있습니다.

자세한 내용은 증분 일치 항목 찾기 섹션을 참조하세요.

FindMatches 변환 사용

이 FindMatches 변환을 사용하여 원본 데이터에서 중복 레코드를 찾을 수 있습니다. 변환의 학습을 돕기 위한 레이블 지정 파일이 생성 또는 제공됩니다.

> **Note**
>
> 현재 사용자 지정 암호화 키를 사용하는 FindMatches 변환은 다음 리전에서 지원되지 않습니다.
>
> - 아시아 태평양(오사카) - ap-northeast-3

FindMatches 변환을 시작하려면 아래 단계를 수행합니다. 자세한 고급 예제는 AWS 빅 데이터 블로그의 Harmonize data using AWS Glue and AWS Lake Formation FindMatches ML to build a customer 360 view를 참조하세요.

일치 항목 찾기 변환을 사용하여 시작하기

FindMatches 변환을 시작하려면 다음 단계를 따르십시오.

1. AWS Glue Data Catalog에서 정리할 원본 데이터의 테이블을 만듭니다. 크롤러 생성 방법에 대한 자세한 내용은 AWS Glue 콘솔에서 크롤러 작업.을 참조하세요.

   원본 데이터가 CSV(쉼표로 구분된 값) 파일 등 텍스트 기반의 파일이라면 다음을 고려하십시오.

   - 입력 레코드 CSV 파일과 레이블 지정 파일을 별도의 파일에 보관합니다. 그렇지 않으면 AWS Glue 크롤러가 이를 동일한 테이블의 각 부분으로 간주하고 Data Catalog에 테이블을 잘못 생성할 수 있습니다.
   - CSV 파일에 ASCII 문자만 들어 있는 경우를 제외하고, CSV 파일에는 BOM(바이트 순서 표시) 없는 UTF-8 인코딩을 사용해야 합니다. Microsoft Excel은 흔히 UTF-8 CSV 파일의 시작 부분에 BOM을 추가합니다. 이를 제거하려면 텍스트 편집기에서 CSV 파일을 열고 파일을 UTF-8 without BOM(BOM 없는 UTF-8)으로 다시 저장하십시오.
2. AWS Glue 콘솔에서 작업을 만들고 Find matches(일치 항목 찾기) 변환 유형을 선택합니다.

> **Important**
>
> 작업을 위해 선택하는 데이터 원본 테이블의 열 수는 100개를 넘을 수 없습니다.

3. [레이블 지정 파일 생성(Generate labeling file)]을 선택하여 AWS Glue에 레이블 지정 파일을 생성하도록 지시합니다. AWS Glue가 각 `labeling_set_id`에 비슷한 레코드를 그룹화하기 위한 첫 번째 전달을 맡아 그러한 그룹화를 검토할 수 있게 해줍니다. 사용자의 레이블이 `label` 열과 일치합니다.
   - 일치하는 행을 나타내는 레코드 예시인 레이블 지정 파일이 이미 있는 경우, 해당 파일을 Amazon Simple Storage Service(Amazon S3)에 업로드합니다. 레이블 지정 파일의 형식에 대한 자세한 내용은 레이블 지정 파일 형식 단원을 참조하십시오. 4단계로 이동합니다.
4. 레이블 지정 파일을 다운로드하고 레이블링 단원에 설명된 대로 해당 파일에 레이블을 지정합니다.
5. 수정된 레이블 지정 파일을 업로드합니다. AWS Glue가 일치 항목 찾는 방법을 변환에 학습시키기 위한 작업을 실행합니다.

   Machine learning transforms(기계 학습 변환) 목록 페이지에서 기록 탭을 선택합니다. AWS Glue가 다음 작업을 수행하면 이 페이지에 표시됩니다.
   - 레이블 가져오기
   - 레이블 내보내기
   - 레이블 생성
   - 품질 예측
6. 더 나은 변환을 만들기 위해 레이블 지정 파일을 반복해서 다운로드하고, 레이블 지정하고, 업로드할 수 있습니다. 처음 실행할 때는 불일치 레코드가 훨씬 더 많을 수 있습니다. 그러나 레이블 지정 파일을 확인하면서 계속 학습시키면 AWS Glue가 학습을 합니다.
7. 일치 항목 찾기의 성능과 결과를 측정하여 변환을 평가하고 조정하십시오. 자세한 내용은 AWS Glue에서 기계 학습 변환 튜닝 섹션을 참조하세요.

레이블링

`FindMatches`가 레이블 지정 파일을 생성할 때 사용자의 소스 테이블에서 레코드가 선택됩니다. 이전의 학습을 토대로 `FindMatches`는 학습해야 할 가장 중요한 레코드를 파악합니다.

레이블 지정이란 레이블 지정 파일을 편집(Microsoft Excel 같은 스프레드시트 사용을 권장)하고 일치 및 불일치 레코드를 식별하는 `label` 열에 식별자나 레이블을 추가하는 것을 뜻합니다. 소스 데이터의 일치 항목을 명확하고 일관성 있게 정의해야 합니다. `FindMatches`는 사용자가 일치(또는 불일치) 항목으로 지정한 레코드에서 학습하고, 사용자의 결정을 토대로 중복 레코드 찾는 방법을 배웁니다.

`FindMatches`에서 레이블 지정 파일이 생성될 때 약 100개의 레코드가 생성됩니다. 이러한 100개의 레코드는 보통 10개의 레이블 지정 세트로 나뉘며, 각각의 레이블 지정 세트는 `FindMatches`에서 생성된 고유한 `labeling_set_id`으로 식별됩니다. 각 레이블 지정 세트는 다른 레이블 지정 세트와 독

립적인 별도의 레이블 지정 작업으로 간주되어야 합니다. 각각의 레이블 지정 세트 내에서 일치하는 레코드와 일치하지 않는 레코드를 식별하는 작업을 합니다.

스프레드시트에서의 레이블 지정 파일 편집 관련 팁입니다.

스프레드 시트 애플리케이션에서 레이블 지정 파일을 편집할 때는 다음을 고려하십시오.

- 열 필드가 전부 확장된 상태에서는 파일이 열리지 않을 수 있습니다. 원하는 셀의 내용을 보기 위해 `labeling_set_id` 및 `label` 열을 확장해야 할 수 있습니다.
- `long` 데이터 유형 등 기본 키 열이 숫자인 경우, 스프레드시트에서 이를 숫자로 해석하고 값을 변경할 수 있습니다. 이 키 값은 텍스트로 취급해야 합니다. 이 문제를 해결하려면 기본 키 열의 모든 셀을 텍스트 데이터로 서식 지정하십시오.

레이블 지정 파일 형식

`FindMatches` 변환을 가르치기 위해 AWS Glue에서 생성된 레이블 지정 파일은 다음 형식을 사용합니다. AWS Glue에 대한 자체 파일을 생성하는 경우에는 다음 형식도 따라야 합니다.

- CSV(쉼표로 구분된 값) 파일입니다.
- `UTF-8`로 인코딩해야 합니다 . Microsoft Windows에서 이 파일을 편집하면 `cp1252`로 인코딩될 수 있습니다.
- Amazon S3 위치에 있어야 AWS Glue로 전달할 수 있습니다.
- 각각의 레이블 지정 태스크마다 적당한 수의 행을 사용하세요. 태스크당 10~20행이 좋지만, 태스크당 2~30행도 허용 가능합니다. 행이 50개 이상인 작업은 권장되지 않으며 결과가 좋지 않거나 시스템 오류가 발생할 수 있습니다.
- “일치”또는 “불일치”로 레이블이 지정된 레코드 쌍으로 구성되고 이미 레이블이 지정된 데이터인 경우에는 괜찮습니다. 이렇게 레이블이 지정된 쌍은 크기가 2인 레이블 지정 세트로 표현될 수 있습니다. 이때 일치하는 경우에는 두 레코드를 모두 문자 “A”로 레이블 지정하고 일치하지 않는 경우에는 하나는 “A”로, 다른 하나는 “B”로 레이블 지정합니다.

> **Note**
>
> 레이블 지정 파일에는 추가 열이 있기 때문에 원본 데이터가 들어 있는 파일과는 스키마가 다릅니다. AWS Glue 크롤러가 Data Catalog에서 테이블을 생성할 때 레이블 지정 파일을 고려하는 일이 없도록 이 파일을 변환 입력 CSV 파일과 다른 폴더에 저장합니다. 그렇지 않으면 AWS Glue 크롤러가 만든 테이블에 데이터가 올바르게 표현되지 않을 수 있습니다.

- 처음 두 열(`labeling_set_id`, `label`)은 AWS Glue의 필수 열입니다. 나머지 열은 처리할 데이터의 스키마와 일치해야 합니다.
- 각 `labeling_set_id`에 대해 동일한 레이블을 사용하여 일치하는 레코드를 모두 식별합니다. 레이블이란 `label` 열에 배치된 고유한 문자열입니다. A, B, C 같이 단순한 문자로 된 레이블을 사용하는 것이 좋습니다. 레이블은 대소문자를 구분하며 `label` 열에 입력됩니다.
- 동일한 `labeling_set_id`과 동일한 레이블을 포함하는 행은 일치 항목으로 레이블이 지정되는 것으로 간주됩니다.
- 동일한 `labeling_set_id`과 다른 레이블을 포함하는 행은 일치 항목이 아닌 것으로 레이블이 지정됩니다.
- 다른 `labeling_set_id`를 포함하는 행은 일치하거나 일치하지 않은 정보를 전달하지 않는 것으로 간주됩니다.

  다음은 데이터 레이블 지정의 예입니다.

labeling_set_id	레이블	first_name	last_name	생일
ABC123	A	존	Doe	04/01/1980
ABC123	B	Jane	Smith	04/03/1980
ABC123	A	Johnny	Doe	04/01/1980
ABC123	A	Jon	Doe	04/01/1980
DEF345	A	Richard	Jones	12/11/1992
DEF345	A	Rich	Jones	11/12/1992
DEF345	B	Sarah	Jones	12/11/1992
DEF345	C	Richie	Jones Jr.	05/06/2017
DEF345	B	Sarah	Jones-Walker	12/11/1992
GHI678	A	Robert	Miller	1/3/1999
GHI678	A	Bob	Miller	1/3/1999
XYZABC	A	William	Robinson	2/5/2001

labeling_set_id	레이블	first_name	last_name	생일
XYZABC	B	Andrew	Robinson	2/5/1971

- 위의 예에서는 John/Johnny/Jon Doe를 일치 항목으로 식별하고 이러한 레코드들이 Jane Smith와 일치하지 않는다고 시스템을 학습시킵니다. 이와 별도로 Richard와 Rich Jones가 같은 사람이지만 이들 레코드가 Sarah Jones/Jones-Walker와 Richie Jones Jr와 일치하지 않는다고 시스템을 학습시킵니다.
- 보시다시피 레이블의 범위는 `labeling_set_id`로 제한됩니다 . 따라서 레이블은 `labeling_set_id` 경계를 벗어나지 않습니다. 예를 들어, `labeling_set_id` 1의 레이블 "A"는 `labeling_set_id` 2의 레이블 "A"와 아무런 관계가 없습니다.
- 레코드가 레이블 지정 세트 내에서 일치 항목이 없는 경우, 고유한 레이블을 지정합니다. 예를 들어, Jane Smith는 레이블 지정 세트 ABC123의 레코드와 일치하지 않으므로 B라는 레이블이 있는 레이블 지정 세트의 유일한 레코드입니다.
- 레이블 지정 세트 “GHI678"은 레이블 지정 세트가 일치한다는 것을 보여주기 위해 동일한 레이블이 지정된 두 개의 레코드로 구성 될 수 있음을 보여줍니다. 마찬가지로 “XYZABC”는 일치하지 않음을 나타내기 위해 서로 다른 레이블이 지정된 두 개의 레코드를 보여줍니다.
- 때로 레이블 지정 세트에 일치하는 항목이 없거나(예: 레이블 지정 세트의 모든 레코드에 다른 레이블을 지정하는 경우), 레이블 지정 세트가 모두 “동일"할 수 있다는(모두 동일한 레이블을 지정한 경우) 점에 유의하십시오. 레이블 지정 세트가 기준에 따라 “동일”하지 않은 레코드의 예들을 총체적으로 포함하는 경우에는 괜찮습니다.

> **Important**
>
> AWS Glue에 전달하는 IAM 역할이 레이블 지정 파일이 들어 있는 Amazon S3 버킷에 액세스할 수 있는지 확인합니다. 명명 규칙에 의해 AWS Glue 정책이 Amazon S3 버킷이나 폴더 등 이름에 aws-glue- 접두사가 붙은 곳에 권한을 부여합니다. 레이블 지정 파일이 다른 위치에 있는 경우, IAM 역할에 해당 위치에 대한 권한을 추가합니다.

AWS Glue에서 기계 학습 변환 튜닝

AWS Glue에서 기계 학습 변환을 튜닝하여 데이터 정리 작업의 결과를 개선함으로써 목표를 충족할 수 있습니다. 변환을 개선하려면 레이블 지정 세트를 생성하고, 레이블을 추가한 후, 원하는 결과를 얻을 때까지 이러한 단계를 여러 번 반복하여 교육할 수 있습니다. 일부 기계 학습 파라미터를 변경하여 튜닝할 수도 있습니다.

기계 학습 변환에 대한 자세한 내용은 AWS Lake Formation FindMatches로 레코드 매칭 단원을 참조하십시오.

주제

- 기계 학습 측정
- 정밀도와 재현율 중에서 결정
- 정확도와 비용 중에서 결정
- 일치 신뢰도 점수를 사용하여 일치 항목의 품질 추정
- 일치 항목 찾기 변환 교육

기계 학습 측정

기계 학습 변환을 튜닝하는 데 사용되는 측정을 이해하려면 다음 용어를 숙지해야 합니다.

참 긍정(TP)

변환이 올바르게 찾은 데이터의 일치로, 적중이라고도 합니다.

참 부정(TN)

변환이 올바르게 거부한 데이터의 불일치입니다.

거짓 긍정(FP)

변환이 일치로 잘못 분류한 데이터의 불일치로, 거짓 경보라고도 합니다.

거짓 부정(FN)

변환이 찾지 못한 데이터의 일치로, 누락이라고도 합니다.

기계 학습에 사용되는 용어에 대한 자세한 내용은 Wikipedia에서 Confusion matrix를 참조하십시오.

기계 학습 변환을 튜닝하려면 변환의 Advanced properties(고급 속성)에서 다음 측정 값을 변경하면 됩니다.

- [정밀도(Precision)]는 변환이 긍정으로 식별하는 총 레코드 수(참 긍정 및 거짓 긍정) 중에서 참 긍정을 얼마나 잘 찾는지 측정합니다. 자세한 내용은 Wikipedia의 정밀도 및 재현율을 참조하십시오.
- 재현율은 변환이 소스 데이터의 총 레코드에서 참 긍정을 얼마나 잘 찾는지 측정합니다. 자세한 내용은 Wikipedia의 정밀도 및 재현율을 참조하십시오.

- 정확도는 변환이 참 긍정과 참 부정을 얼마나 잘 찾는지 측정합니다. 정확도를 증가시키려면 더 많은 기계 리소스와 비용이 필요합니다. 하지만 이렇게 하면 재현율도 증가합니다. 자세한 내용은 Wikipedia의 Accuracy and precision을 참조하십시오.
- 비용은 변환을 실행하는 데 사용한 컴퓨팅 리소스(따라서 비용)의 양을 측정합니다.

정밀도와 재현율 중에서 결정

각 `FindMatches` 변환에는 `precision-recall` 파라미터가 포함되어 있습니다. 이 파라미터를 사용하여 다음 중 하나를 지정합니다.

- 변환이 실제로는 일치하지 않는 두 레코드가 일치한다고 잘못 보고할 것이 더 우려되는 경우, 정밀도를 강조해야 합니다.
- 변환이 정말로 일치하는 레코드를 감지하지 못할 것이 더 우려되는 경우, 재현율을 강조해야 합니다.

AWS Glue 콘솔에서 또는 AWS Glue 기계 학습 API 작업을 사용하여 균형을 이룰 수 있습니다.

정밀도를 추구해야 하는 경우

`FindMatches`가 실제로는 일치하지 않는 레코드 페어가 일치한다고 할 위험이 더 우려되는 경우 정밀도를 추구합니다. 정밀도를 추구하려면 더 높은 정밀도-재현율 균형 값을 선택합니다. 더 큰 값의 경우 `FindMatches` 변환에는 레코드 페어가 일치해야 한다고 결정하기 위해 더 많은 증거가 필요합니다. 이 변환은 레코드가 일치하지 않는다고 말하는 경향이 있습니다.

예를 들어 `FindMatches`를 사용하여 비디오 카탈로그에서 중복 항목을 감지하며, 변환에 더 큰 정밀도-재현율 값을 제공한다고 가정하겠습니다. 변환이 Star Wars: A New Hope가 Star Wars: The Empire Strikes Back과 같다고 잘못 감지하는 경우, A New Hope을 원하는 고객에게 The Empire Strikes Back이 표시될 수 있습니다. 이는 부정적인 고객 경험이 됩니다.

그러나 변환이 Star Wars: A New Hope와 Star Wars: Episode IV—A New Hope가 동일한 항목이라고 감지하지 못하면 고객은 처음에는 혼란스러울 수 있으나 결국에는 둘을 동일하게 인식할 수 있습니다. 이는 실수이지만, 이전 시나리오만큼 나쁘지는 않습니다.

재현율을 추구해야 하는 경우

`FindMatches` 변환 결과가 실제로는 일치하는 레코드 페어를 감지하지 못할 것이 더 우려되는 경우 재현율을 추구합니다. 재현율을 추구하려면 더 낮은 정밀도-재현율 균형 값을 선택합니다. 더 작은 값의 경우 `FindMatches` 변환에는 레코드 페어가 일치해야 한다고 결정하기 위해 더 적은 증거가 필요합니다. 이 변환은 레코드가 일치한다고 말하는 경향이 있습니다.

예를 들어 이는 보안 조직의 우선 순위일 수 있습니다. 고객을 알려진 사기꾼 목록과 대조해 보고 있으며, 고객이 사기꾼인지 여부를 확인하는 것이 중요하다고 가정하겠습니다. FindMatches를 사용하여 사기꾼 목록을 고객 목록과 대조하고 있습니다. FindMatches가 두 목록 간에 일치 항목을 감지할 때마다 해당 사람이 실제로 사기꾼인지 확인하기 위해 인간 감사자가 지정됩니다. 조직은 정밀도보다 재현율을 선택하는 것을 선호할 수 있습니다. 다시 말해 감사자가 수동으로 검토하고 고객이 사기꾼이 아닌 일부 사례를 거부하도록 하는 것입니다. 그러면 고객이 실제로 사기꾼 목록에 있음을 식별하지 못할 일이 없습니다.

정밀도와 재현율을 모두 추구하는 방법

정밀도와 재현율을 모두 개선하는 가장 좋은 방법은 더 많은 데이터에 레이블을 지정하는 것입니다. 더 많은 데이터에 레이블을 지정하면 FindMatches 변환의 전반적인 정확도가 개선되므로 정밀도와 재현율도 개선됩니다. 그럼에도 불구하고 가장 정확한 변환에서조차 정밀도 또는 재현율을 추구하는 실험이 필요하거나 중간 값을 선택해야 하는 애매한 상황이 항상 있습니다.

정확도와 비용 중에서 결정

각 FindMatches 변환에는 accuracy-cost 파라미터가 포함되어 있습니다. 이 파라미터를 사용하여 다음 중 하나를 지정할 수 있습니다.

- 변환이 두 레코드가 일치한다고 정확하게 보고하는 것이 더 우려되는 경우, 정확도를 강조해야 합니다.
- 변환 실행 비용이나 속도가 더 우려되는 경우 더 낮은 비용을 강조해야 합니다.

AWS Glue 콘솔에서 또는 AWS Glue 기계 학습 API 작업을 사용하여 균형을 이룰 수 있습니다.

정확도를 추구해야 하는 경우

find matches 결과에 일치 항목이 포함되지 않을 위험이 더 우려되는 경우 정확도를 추구합니다. 정확도를 추구하려면 더 높은 정확도-비용 균형 값을 선택합니다. 더 높은 값의 경우 FindMatches 변환에는 정확하게 일치하는 레코드를 더 철저히 검색하기 위해 더 많은 시간이 필요합니다. 이 파라미터는 일치하지 않는 레코드 페어를 일치한다고 잘못 호출할 가능성을 낮추지는 않습니다. 이 변환은 일치 항목을 찾는 데 더 많은 시간을 보내는 경향이 있습니다.

비용을 추구해야 하는 경우

find matches 변환을 실행하는 데 드는 비용을 더 우려하고 일치 항목의 수를 찾는 것에 덜 우려하는 경우 비용을 추구합니다. 비용을 추구하려면 더 낮은 정확도-비용 균형 값을 선택합니다. 더 낮은 값의 경우 FindMatches 변환은 실행할 리소스가 더 적게 필요합니다. 이 변환은 일치 항목을 더 적게

찾는 경향이 있습니다. 더 낮은 비용을 추구했을 때 결과가 수용 가능한 수준인 경우 이 설정을 사용합니다.

정확도와 더 낮은 비용을 모두 추구하는 방법

더 많은 레코드 쌍을 검토하여 일치 여부를 확인하려면 더 많은 시간이 걸립니다. 품질을 낮추지 않고 비용을 줄이려면 다음과 같은 단계를 수행하면 됩니다.

- 데이트 원본에서 일치에 대해 우려하지 않는 레코드를 제거합니다.
- 일치/불일치 결정을 내리는 것이 유용하지 않다고 확신하는 열을 데이터 원본에서 제거합니다. 이를 결정하는 좋은 방법은 레코드 세트가 “동일”한지 여부에 대한 자신의 결정에 영향을 미친다고 생각하지 않는 열을 제거하는 것입니다.

일치 신뢰도 점수를 사용하여 일치 항목의 품질 추정

일치 신뢰도 점수는 FindMatchs에서 찾은 일치 항목의 품질을 추정하여 기계 학습 모델의 신뢰도가 높거나 불확실하거나 낮은 일치 레코드를 구별합니다. 일치 신뢰도 점수는 0에서 1 사이이며, 점수가 높을수록 유사성이 높아집니다. 일치 신뢰도 점수를 검사하면 신뢰도가 높은 일치 항목 클러스터(병합하기로 결정할 수 있음), 불확실한 클러스터(사람이 검토하기로 결정할 수 있음), 신뢰도가 낮은 클러스터(거부하기로 결정할 수 있음)를 구별할 수 있습니다.

일치 신뢰도 점수가 높은데 일치 항목이 없거나 점수가 낮은데 실제로 일치 항목이 있는 경우 훈련 데이터를 조정하는 것이 좋습니다.

신뢰도 점수는 모든 FindMatches 결정을 검토하는 것이 불가능한 대규모 산업 데이터 집합이 있는 경우에 특히 유용합니다.

일치 신뢰도 점수는 AWS Glue 버전 2.0 이상에서 사용 가능합니다.

일치 신뢰도 점수 생성

`FindMatches` 또는 `FindIncrementalMatches` API를 호출할 때 `computeMatchConfidenceScores`의 부울 값을 True로 설정하면 일치 신뢰도 점수를 생성할 수 있습니다.

AWS Glue는 새 `column match_confidence_score`를 출력에 추가합니다.

일치 점수 매기기 예

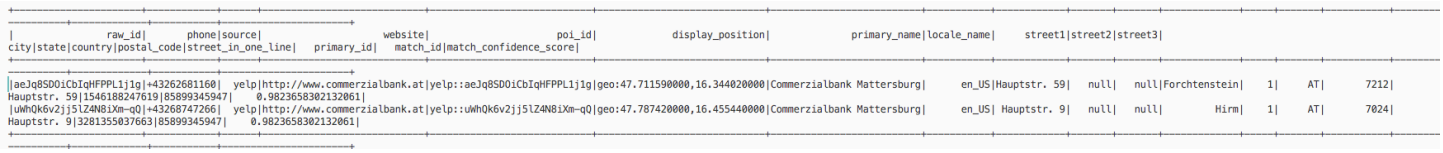
예를 들어 다음 일치 레코드를 고려해 보세요.

점수 >= 0.9

일치 레코드 요약:

```
  primary_id   |    match_id   | match_confidence_score

3281355037663    85899345947   0.9823658302132061
1546188247619    85899345947   0.9823658302132061
```

세부 정보:



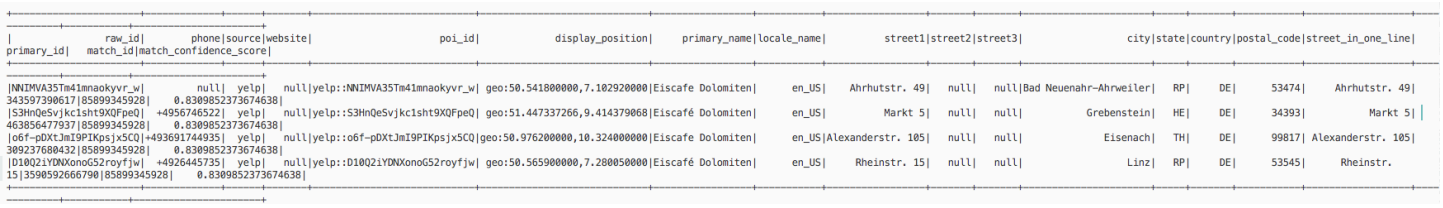
이 예제에서는 두 레코드가 매우 유사하고 `display_position`, `primary_name`, `street name`을 공유한다는 것을 확인할 수 있습니다.

점수 >= 0.8 및 점수 < 0.9

일치 레코드 요약:

```
  primary_id   |    match_id   | match_confidence_score

309237680432     85899345928   0.8309852373674638
3590592666790    85899345928   0.8309852373674638
343597390617     85899345928   0.8309852373674638
249108124906     85899345928   0.8309852373674638
463856477937     85899345928   0.8309852373674638
```

세부 정보:



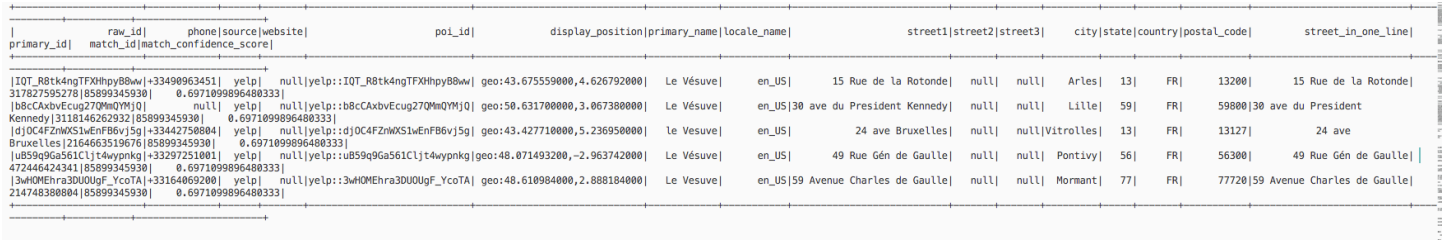
이 예제에서는 두 레코드가 동일한 `primary_name` 및 `country`를 공유한다는 것을 확인할 수 있습니다.

점수 >= 0.6 및 점수 < 0.7

일치 레코드 요약:

```
  primary_id   |   match_id   | match_confidence_score

2164663519676    85899345930   0.6971099896480333
 317827595278    85899345930   0.6971099896480333
 472446424341    85899345930   0.6971099896480333
3118146262932    85899345930   0.6971099896480333
 214748380804    85899345930   0.6971099896480333
```

세부 정보:



```
+-----------+-----------+-----------
```

"정밀도", "재현율" 및 "정밀도 재현율 곡선 아래 영역" 지표를 살펴보는 것입니다. 더 많은 작업 세트에 레이블을 지정한 후 이러한 지표를 다시 실행하고 개선되었는지 확인합니다. 작업 세트 몇 개에 레이블을 지정한 후 초점을 둔 지표에 개선이 보이지 않는 경우, 변환 품질이 정체 상태에 도달했을 수 있습니다.

참 긍정과 참 부정 레이블이 모두 필요한 이유는 무엇인가요?

FindMatches 변환은 사용자가 일치한다고 생각하는 항목을 학습하려면 긍정과 부정 예가 모두 필요합니다. FindMatches에서 생성하는 훈련 데이터에 레이블을 지정하는 경우(예: I do not have labels(레이블이 없음) 옵션 사용), FindMatches는 “레이블 세트 ID” 세트를 생성하려고 시도합니다. 각 작업 내에서 일부 레코드에는 동일한 “레이블”을, 다른 레코드에는 다른 “레이블”을 제공합니다. 다시 말해 이 작업은 일반적으로 모두 동일하지도 모두 다르지도 않습니다(그러나 특정 작업이 모두 “동일”하거나 모두 “동일하지 않은” 경우는 괜찮음).

[S3에서 레이블 업로드(Upload labels from S3)] 옵션을 사용하여 FindMatches 변환을 가르치는 경우 일치하는 레코드와 일치하지 않는 레코드의 예를 모두 포함합니다. 한 유형만 사용할 수 있습니다. 이러한 레이블을 사용하면 더 정확한 FindMatches 변환을 구축할 수 있지만, Generate labeling file(레이블 지정 파일 생성) 옵션을 사용하여 생성하는 일부 레코드에 레이블을 지정해야 합니다.

변환을 교육할 때 변환이 정확히 일치하도록 강제할 수 있나요?

FindMatches 변환은 사용자가 제공하는 레이블에서 학습하므로, 제공된 레이블을 따르지 않는 레코드 페어를 생성할 수 있습니다. FindMatches 변환이 레이블을 따르도록 하려면 FindMatchesParameter에서 EnforceProvidedLabels를 선택합니다.

ML 변환이 항목을 참이 아닌 일치 항목으로 식별하는 경우 어떤 기술을 사용할 수 있나요?

다음과 같은 기술을 사용할 수 있습니다.

- precisionRecallTradeoff을 더 큰 값으로 높입니다. 그러면 결국 더 적은 일치 항목을 찾게 되지만, 충분히 높은 값에 도달하면 큰 클러스터를 분리해야 합니다.
- 잘못된 결과에 해당하는 출력 행을 가져와서 레이블 지정 세트로 다시 포맷합니다(match_id 열을 제거하고, labeling_set_id 및 label 열 추가). 필요한 경우 여러 레이블 지정 세트로 분리(세분화)하여 레이블러가 레이블을 할당하는 동안 각 레이블 지정 세트를 염두에 둘 수 있도록 합니다. 그런 다음 일치하는 세트에 올바르게 레이블을 지정하고 레이블 파일을 업로드한 후 기존 레이블에 추가합니다. 그러면 패턴을 이해하기 위해 찾고 있는 사항에 대해 변환기를 충분히 교육할 수 있습니다.
- (고급) 마지막으로 데이터를 보고 시스템이 알아채지 못하고 있음을 감지할 수 있는 패턴이 있는지 확인합니다. 표준 AWS Glue 함수를 사용해 해당 데이터를 사전 처리하여 데이터를 정규화합니다.

중요한 데이터를 자체 열로 따로 분리하여 알고리즘이 학습하도록 할 내용을 강조 표시합니다. 또는 관련된 데이터가 있는 열에서 결합된 열을 생성합니다.

기계 학습 변환 작업

AWS Glue를 사용하여 데이터를 정리하는 데 사용할 수 있는 사용자 지정 기계 학습 변환을 생성할 수 있습니다. AWS Glue 콘솔에서 작업을 생성할 때 이러한 변환을 사용할 수 있습니다.

기계 학습 변환을 생성하는 방법에 대한 자세한 내용은 AWS Lake Formation FindMatches로 레코드 매칭 단원을 참조하십시오.

주제

- 변환 속성
- 기계 학습 변환 추가 및 편집
- 변환 세부 정보 보기
- 레이블을 사용하여 변환 학습

변환 속성

기존 기계 학습 변환을 보려면 AWS Management Console에 로그인하고 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다. 탐색 창의 데이터 통합 및 ETL에서 데이터 분류 도구 > 레코드 일치를 선택합니다.

각 변환의 속성:

변환 이름

생성할 때 변환에 부여한 고유 이름입니다.

ID

변환의 고유 식별자입니다.

레이블 수

변환 학습에 도움이 되도록 지정된 레이블 지정 파일의 레이블 수입니다.

상태 표시기

변환이 [준비(Ready)] 상태인지 또는 [훈련 필요(Needs training)] 상태인지를 나타냅니다. 작업에서 기계 학습 변환을 성공적으로 실행하려면 변환이 [준비(Ready)] 상태여야 합니다.

Created

변환이 생성된 날짜입니다.

수정됨

변환이 마지막으로 업데이트된 날짜입니다.

설명

변환에 대해 제공된 설명(제공된 경우)입니다.

AWS Glue 버전

사용된 AWS Glue 버전입니다.

실행 ID

생성할 때 변환에 부여한 고유 이름입니다.

작업 유형

기계 학습 변환의 유형입니다. 예: Find matching records(일치하는 레코드 찾기)

상태 표시기

작업 실행 상태를 나타냅니다. 가능한 상태는 다음과 같습니다.

- Starting(시작 중)
- 실행 중
- Stopping(중지 중)
- Stopped(중지됨)
- Succeeded(성공)
- Failed(실패)
- 제한 시간

Error

상태가 실패인 경우 실패의 원인을 설명하는 오류 메시지가 표시됩니다.

기계 학습 변환 추가 및 편집

AWS Glue 콘솔에서 변환을 보기, 삭제, 설정 및 교육 또는 튜닝할 수 있습니다. 목록에서 변환 옆의 확인란을 선택하고, Action(작업)을 선택한 다음, 수행할 작업을 선택합니다.

새 ML 변환 생성

새 기계 학습 변환을 추가하려면 변환 생성을 선택합니다. 크롤로 추가 마법사의 지시에 따릅니다. 자세한 내용은 AWS Lake Formation FindMatches로 레코드 매칭 단원을 참조하십시오.

단계 1. 변환 속성을 설정합니다.

1. 이름 및 설명(선택 사항)을 입력합니다.
2. 선택적으로 보안 구성을 설정합니다. 기계 학습 변환에 데이터 암호화 사용 섹션을 참조하세요.
3. 선택적으로 작업 실행 설정을 설정합니다. 작업 실행 설정을 통해 작업 실행 방식을 사용자 지정할 수 있습니다. 작업자 유형, 작업자 수, 작업 제한 시간(분), 재시도 횟수, AWS Glue 버전을 선택합니다.
4. 선택적으로 태그를 설정합니다. 태그는 AWS 리소스에 할당할 수 있는 레이블입니다. 각 태그는 키와 값(선택 사항)으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

2단계. 테이블과 프라이머리 키를 선택합니다.

1. AWS Glue 카탈로그 데이터베이스 및 테이블을 선택합니다.
2. 선택한 테이블에서 프라이머리 키를 선택합니다. 프라이머리 키 열에는 일반적으로 데이터 소스의 모든 레코드에 대한 고유 식별자가 포함됩니다.

3단계. 튜닝 옵션을 선택합니다.

1. 재현율 대 정밀도의 경우 튜닝 값을 선택하여 재현율 또는 정밀도에 유리하도록 변환을 튜닝합니다. 기본적으로 균형이 선택되지만 재현율에 유리하거나 정밀도에 유리하도록 선택할 수 있습니다. 또는 사용자 지정을 선택하고 0.0에서 1.0(포함) 사이의 값을 입력할 수 있습니다.
2. 낮은 비용 및 정확도의 경우 낮은 비용에 유리하거나 정확도에 유리하도록 튜닝 값을 선택합니다. 또는 사용자 지정을 선택하고 0.0에서 1.0(포함) 사이의 값을 입력합니다.
3. 일치 적용의 경우 사용하는 레이블과 일치하도록 강제 출력함으로서 ML 변환을 학습시키려면 레이블과 일치하도록 강제 출력을 선택합니다.

4단계. 검토 및 생성.

1. 1~3단계의 옵션을 검토합니다.

2. 수정이 필요한 모든 단계에서 편집을 선택합니다. 변환 생성을 선택하여 변환 생성 마법사를 완료합니다.

기계 학습 변환에 데이터 암호화 사용

기계 학습 변환을 AWS Glue에 추가할 때 데이터 원본 또는 데이터 대상과 연결된 보안 구성을 선택적으로 지정할 수 있습니다. 데이터를 저장하는 데 사용된 Amazon S3 버킷이 보안 구성으로 암호화된 경우 변환을 생성할 때 동일한 보안 구성을 지정합니다.

또한 AWS KMS(SSE-KMS)로 서버 측 암호화를 사용하여 모델과 레이블을 암호화하여 권한이 없는 사람이 모델을 검사하지 못하도록 할 수 있습니다. 이 옵션을 선택하면 이름으로 AWS KMS key를 선택하라는 메시지가 나타나거나 [키 ARN 입력(Enter a key ARN)]을 선택할 수 있습니다. KMS 키에 대한 ARN을 입력하도록 선택하면 KMS 키 ARN을 입력할 수 있는 두 번째 필드가 나타납니다.

> **Note**
>
> 현재 사용자 지정 암호화 키를 사용하는 ML 변환은 다음 리전에서 지원되지 않습니다.
>
> - 아시아 태평양(오사카) - `ap-northeast-3`

변환 세부 정보 보기

변환 속성 보기

변환 속성 페이지에는 변환 속성이 포함되어 있습니다. 이 탭은 다음을 포함하여 변환 정의에 대한 세부 정보를 보여 줍니다.

- Transform name(변환 이름)은 변환의 이름을 보여 줍니다.
- Type(유형)에는 변환의 유형이 나열됩니다.
- Status(상태)는 변환이 스크립트 또는 작업에 사용할 준비가 되었는지 여부를 표시합니다.
- Force output to match labels(레이블과 일치하도록 출력 강제)는 변환이 사용자가 제공한 레이블과 일치하도록 출력을 강제하는지 여부를 표시합니다.
- [Spark 버전(Spark version)]은 변환을 추가할 때 [태스크 실행 속성(Task run properties)]에서 선택한 AWS Glue 버전과 관련이 있습니다. AWS Glue 1.0 및 Spark 2.4는 대부분의 고객에게 권장됩니다. 자세한 내용은 AWS Glue 버전을 참조하세요.

기록, 품질 예측 및 태그 탭

변환 세부 정보에는 변환을 생성할 때 정의한 정보가 포함되어 있습니다. 변환에 대한 세부 정보를 보려면 Machine learning transforms(기계 학습 변환) 목록에서 변환을 선택하고 다음 탭에서 정보를 검토합니다.

- 기록
- 품질 예측
- Tags

기록

History(기록) 탭에는 변환 작업 실행 기록이 표시됩니다. 변환을 교육하기 위해 여러 유형의 작업이 실행됩니다. 각 작업에 대한 실행 지표에는 다음이 포함됩니다.

- Run ID(실행 ID)는 이 작업의 각 실행에 대해 AWS Glue에서 생성된 식별자입니다.
- Task type(작업 유형)은 작업 실행의 유형을 보여 줍니다.
- Run status(실행 상태)에는 맨 위에 가장 최근 실행부터 순서대로 나열된 각 작업의 성공 여부를 보여 줍니다.
- 오류는 성공적이지 못한 실행의 결과로서 오류 메시지의 세부 정보를 보여줍니다.
- Start time(시작 시간)은 작업이 시작된 날짜와 시간(로컬 시간)을 보여줍니다.
- 종료 시간은 작업이 종료되는 날짜와 시간(로컬 시간)을 보여줍니다.
- Logs(로그)는 이 작업을 실행할 때 `stdout`에 작성되는 로그와 연결됩니다.

  [로그(Logs)] 링크를 클릭하면 Amazon CloudWatch Logs로 이동됩니다. 그곳에서는 AWS Glue Data Catalog에서 생성된 테이블 및 발생한 오류에 대한 모든 세부 정보를 볼 수 있습니다. CloudWatch 콘솔에서 로그 보존 기간을 관리할 수 있습니다. 기본 로그 보관은 `Never Expire`. 보존 기간 변경 방법에 대한 자세한 내용은 Amazon CloudWatch Logs User Guide의 Change Log Data Retention in CloudWatch Logs를 참조하세요.
- 레이블 파일은 Amazon S3에 대한 생성된 레이블 지정 파일의 링크를 보여줍니다.

품질 예측

Estimate quality(예상 품질) 탭은 변환의 품질을 측정하는 데 사용할 수 있는 지표를 보여 줍니다. 추정치는 레이블 지정 데이터의 하위 세트를 사용하는 변환 일치 예측과 사용자가 제공한 레이블을 비교해

서 계산합니다. 이 추정치는 근사값입니다. 이 탭에서 Estimate quality(예상 품질) 작업 실행을 호출할 수 있습니다.

Estimate quality(예상 품질) 탭에는 다음 속성이 포함된 마지막 Estimate quality(예상 품질) 실행의 지표가 표시됩니다.

- Area under the Precision-Recall curve(정밀도-재현율 곡선 아래 면적)는 전체적 변환 품질의 상한값을 예측하는 단일 숫자입니다. 이 항목은 정밀도-재현율 파라미터에 대해 수행한 선택과 무관합니다. 값이 높을수록 더 매력적인 정밀도-재현율 트레이드오프가 있음을 나타냅니다.
- Precision(정밀도)은 변환이 일치를 정확하게 예측하는 빈도를 예측치로 표시합니다.
- Recall upper limit(재현율 상한)는 실제 일치에 대해 변환이 일치를 예측하는 빈도를 예측치로 표시합니다.
- F1은 0~1의 범위에서 변환의 정확도를 예측합니다. 여기서 1은 최상의 정확도입니다. 자세한 내용은 Wikipedia의 F1 점수를 참조하십시오.
- [열 중요도(Column importance)] 테이블에는 각 열의 열 이름과 중요도 점수가 표시됩니다. 열 중요도는 일치 작업을 수행하는 데 가장 많이 사용되는 레코드의 열을 식별하여 열이 모델에 기여하는 방식을 이해하는 데 도움이 됩니다. 이 데이터는 열 중요도를 높이거나 낮추기 위해 레이블 집합을 추가하거나 변경하라는 메시지를 표시할 수 있습니다.

  중요도 열은 1.0 이하의 십진수로 각 열에 대한 숫자 점수를 제공합니다.

품질 예측과 실제 품질 비교 이해에 대한 자세한 내용은 품질 예측과 엔드 투 엔드(실제) 품질 비교 단원을 참조하십시오.

변환 튜닝에 대한 자세한 내용은 AWS Glue에서 기계 학습 변환 튜닝 단원을 참조하십시오.

품질 예측과 엔드 투 엔드(실제) 품질 비교

AWS Glue에서는 사용자가 일치하는 레이블을 제공했지만 이전에 모델에서는 본 적이 없는 수많은 레코드 페어로 기계 학습된 내부 모델을 제시하여 변환의 품질을 예측합니다. 이러한 품질 예측은 기계 학습된 모델의 품질 함수입니다(변환을 “교육”하기 위해 레이블 지정하는 수많은 레코드의 영향을 받음). 엔드 투 엔드 또는 실제 재현율(`ML transform`으로 자동 계산되지 않음)은 있을 수 있는 다양한 일치 항목을 기계 학습된 모델에 제안하는 `ML transform` 필터링 메커니즘의 영향도 받습니다.

주로 낮은 비용 대비 정확도 튜닝 값을 사용하여 이 필터링 방법을 튜닝할 수 있습니다. 튜닝 값이 정확도 쪽으로 이동할수록 시스템은 일치 항목일 수 있는 레코드 페어에 대해 더 철저하고 비용이 많이 드는 검색을 수행합니다. 더 많은 레코드 페어가 기계 학습된 모델에 공급되며, `ML transform`의 엔드

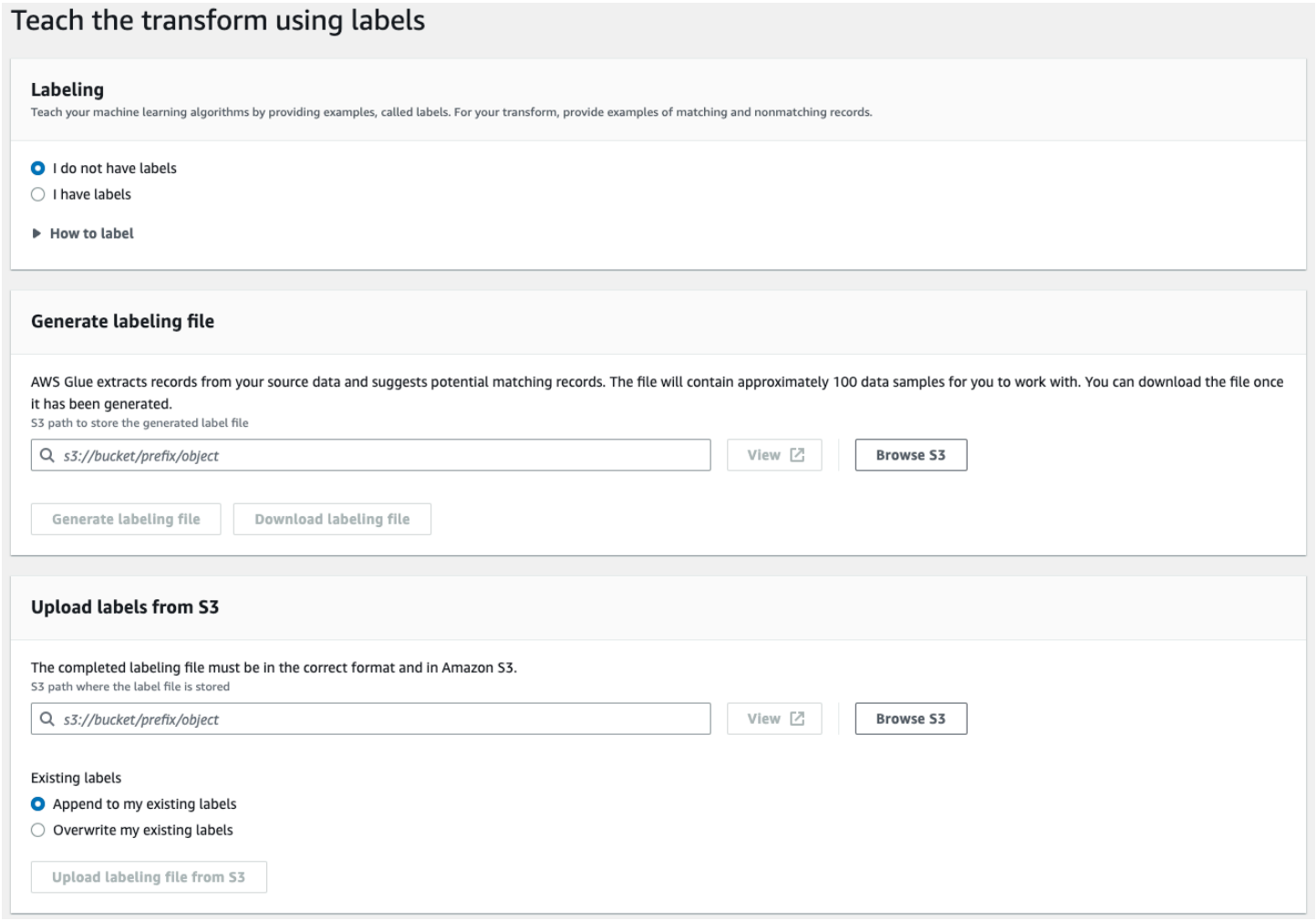
투 엔드 또는 실제 재현율은 예상된 재현율 지표에 더 가깝게 접근합니다. 그 결과 일치 항목에 대한 비용/정확성 트레이드오프 변화에 따른 일치 항목의 엔드 투 엔드 품질 변화는 일반적으로 품질 예측치에 반영되지 않습니다.

Tags

태그는 AWS 리소스에 할당할 수 있는 레이블입니다. 각 태그는 키와 값(선택 사항)으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

레이블을 사용하여 변환 학습

ML 변환 세부 정보 페이지에서 변환 학습을 선택하여 레이블(예제)을 사용한 ML 변환을 학습할 수 있습니다. 예제(레이블이라고도 함)를 제공하여 기계 학습 알고리즘을 학습할 때 사용할 기존 레이블을 선택하거나 레이블 지정 파일을 생성할 수 있습니다.



- 레이블 지정 - 레이블이 있는 경우 레이블 있음을 선택합니다. 레이블이 없으면 레이블 지정 파일을 생성하는 다음 단계를 계속할 수 있습니다.

- 레이블 지정 파일 생성 - AWS Glue는 소스 데이터에서 레코드를 추출하고 일치 가능성이 있는 레코드를 제안합니다. 생성된 레이블 파일을 저장할 Amazon S3 버킷을 선택합니다. 레이블 지정 파일 생성을 선택하여 프로세스를 시작합니다. 완료되면 레이블 지정 파일 다운로드를 선택합니다. 다운로드한 파일에는 레이블에 입력할 수 있는 레이블 열이 있습니다.
- Amazon S3에서 레이블 업로드 - 레이블 파일이 저장되어 있는 Amazon S3 버킷에서 완성된 레이블 지정 파일을 선택합니다. 그런 다음 기존 레이블에 레이블을 추가하거나 기존 레이블을 덮어쓸지 선택합니다. Amazon S3에서 레이블 지정 파일 업로드를 선택합니다.

자습서: AWS Glue로 기계 학습 변환 생성

이 자습서에서는 AWS Glue를 사용하여 ML(기계 학습) 변환을 생성하고 관리하는 작업을 안내합니다. 이 자습서를 사용하기 전에, AWS Glue 콘솔을 사용하여 크롤러 및 작업을 추가하고 스크립트를 편집하는 방법을 숙지해야 합니다. 또한 Amazon Simple Storage Service(Amazon S3) 콘솔에서 파일을 찾고 다운로드하는 방법도 잘 알아야 합니다.

이 예제에서는 `FindMatches` 변환을 만들어 일치하는 레코드를 찾고, 일치 레코드 및 불일치 레코드를 식별하는 방법을 변환에 학습시키고, AWS Glue 작업에 이 변환을 사용해 봅니다. AWS Glue 작업은 `match_id`라는 추가 열을 포함하는 새 Amazon S3 파일을 작성합니다.

이 자습서에서 사용하는 원본 데이터는 `dblp_acm_records.csv`라는 파일입니다. 이 파일은 원래의 DBLP ACM 데이터 세트에서 사용 가능한 교육용 게시물(DBLP 및 ACM)을 수정한 버전입니다. `dblp_acm_records.csv` 파일은 BOM(바이트 순서 표시) 없는 UTF-8 형식의 CSV(쉼표로 구분된 값) 파일입니다.

두 번째 파일인 `dblp_acm_labels.csv`는 이 자습서에 따라 변환을 학습시키는 데 사용할 일치 레코드와 불일치 레코드가 둘 다 들어 있는 레이블 지정 예제 파일입니다.

주제

- 1단계: 소스 데이터 크롤링
- 2단계: 기계 학습 변환 추가
- 3단계: 기계 학습 변환 학습
- 4단계: 기계 학습 변환의 품질 예측
- 5단계: 기계 학습 변환으로 작업 추가 및 실행
- 6단계: Amazon S3의 출력 데이터 확인

1단계: 소스 데이터 크롤링

우선, 원본 Amazon S3 CSV 파일을 크롤링하여 Data Catalog에 그와 상응하는 메타데이터 테이블을 생성합니다.

> **Important**
>
> 크롤러가 이 CSV 파일 전용의 테이블을 만들게 하기 위해 CSV 원본 데이터를 나머지 파일과 다른 Amazon S3 폴더에 저장합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 크롤러, 크롤러 추가를 선택합니다.
3. 마법사를 따라 `demo-db-dblp-acm` 데이터베이스로 출력되는 `demo-crawl-dblp-acm`이라는 크롤러를 만들고 실행합니다. `demo-db-dblp-acm` 데이터베이스가 아직 없으면 이 마법사를 실행하는 동안 만듭니다. 현재 AWS 리전에 있는 샘플 데이터로 이동하는 Amazon S3 포함 경로를 선택합니다. 예를 들어 `us-east-1`의 경우, Amazon S3에서 소스 파일로 이동하는 포함 경로는 `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/records/dblp_acm_records.csv`입니다.

   작업이 성공하면 크롤러가 id, title, authors, venue, year 및 source 열이 있는 `dblp_acm_records_csv` 테이블을 만듭니다.

2단계: 기계 학습 변환 추가

다음으로, `demo-crawl-dblp-acm`이라는 크롤러에서 만든 데이터 원본 테이블의 스키마를 토대로 한 기계 학습 변환을 추가합니다.

1. AWS Glue Console의 탐색 창의 데이터 통합 및 ETL 아래에서 데이터 분류 도구 > 레코드 일치를 선택한 다음 변환 추가를 선택합니다. 마법사를 따라 다음 속성으로 `Find matches` 변환을 생성합니다.

   a. 변환 이름으로 **`demo-xform-dblp-acm`**을 입력합니다. 이것은 원본 데이터에서 일치 항목을 찾을 때 사용하는 변환의 이름입니다.

   b. IAM 역할(IAM role)로는 Amazon S3 원본 데이터, 레이블 지정 파일 및 AWS Glue API 작업에 대한 권한이 있는 IAM 역할을 선택합니다. 자세한 내용은 AWS Glue Developer Guide의 Create an IAM Role for AWS Glue를 참조하세요.

c. [데이터 원본(Data source)]의 경우, 데이터베이스 [demo-db-dblp-acm]의 [dblp_acm_records_csv]라는 이름의 테이블을 선택합니다.

d. 기본 키로는 테이블의 기본 키 열(id)을 선택합니다.

2. 마법사에서 완료를 선택하고 ML transforms(ML 변환) 목록으로 돌아갑니다.

3단계: 기계 학습 변환 학습

이제 자습서의 레이블 지정 샘플 파일을 사용하여 기계 학습 변환을 학습시켜야 합니다.

Ready for use(사용 준비 완료) 상태가 되기 전에는 기계어 변환을 ETL(추출, 변환, 로드) 작업에 사용할 수 없습니다. 변환을 준비시키려면 일치 레코드와 불일치 레코드의 예제를 제공하여 일치 및 불일치 레코드를 식별하는 방법을 가르쳐야 합니다. 변환을 학습시키기 위해 레이블 파일을 생성하고, 레이블을 추가하고, 그 레이블 파일을 업로드할 수 있습니다. 이 자습서에서는 `dblp_acm_labels.csv`라는 레이블 지정 예제 파일을 사용합니다. 레이블 지정 프로세스에 대한 자세한 내용은 레이블링 단원을 참조하십시오.

1. AWS Glue Console의 탐색 창에서 레코드 일치를 선택합니다.
2. `demo-xform-dblp-acm` 변환을 선택한 다음 Action(작업)과 Teach(학습)를 선택합니다. 마법사를 따라 `Find matches` 변환을 학습시킵니다.
3. 변환 속성 페이지에서 I have labels(레이블 있음)를 선택합니다. 현재 AWS 리전에 있는 샘플 레이블 지정 파일로 이동하는 Amazon S3 경로를 선택합니다. 예를 들어 `us-east-1`의 경우, Amazon S3 경로 `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/labels/dblp_acm_labels.csv`에서 제공한 레이블 지정 파일을 업로드하되 기존 레이블 [덮어쓰기(overwrite)] 옵션을 포함합니다. 레이블 지정 파일이 AWS Glue 콘솔과 같은 리전의 Amazon S3에 있어야 합니다.

   레이블 지정 파일을 업로드하면 해당 변환에 데이터 원본 처리 방법을 학습시키는 데 사용할 레이블을 추가하거나 덮어쓰는 작업이 AWS Glue에서 시작됩니다.
4. 마법사의 마지막 페이지에서 완료를 선택하고 ML transforms(ML 변환) 목록으로 돌아옵니다.

4단계: 기계 학습 변환의 품질 예측

이제 기계 학습 변환의 품질을 예측할 수 있습니다. 품질은 레이블 지정 작업을 얼마나 많이 했는지에 따라 달라집니다. 품질 예측에 대한 자세한 내용은 품질 예측 단원을 참조하십시오.

1. AWS Glue Console의 탐색 창의 데이터 통합 및 ETL에서 데이터 분류 도구 > 레코드 일치를 선택합니다.

2. demo-xform-dblp-acm 변환을 선택하고 Estimate quality(품질 예측) 탭을 선택합니다. 해당 변환의 현재 품질이 예측되어 있으면 이 탭에 표시됩니다.
3. Estimate quality(품질 예측)를 선택하여 변환의 품질 예측 작업을 시작합니다. 품질 예측의 정확성은 원본 데이터의 레이블 지정에 달려 있습니다.
4. History(기록) 탭으로 이동합니다. 이 창에는 Estimating quality(품질 예측) 작업을 비롯하여 해당 변환에 실행한 작업이 나열되어 있습니다. 작업 실행에 대한 자세한 내용은 로그를 참조하십시오. 완료된 작업의 실행 상태가 성공인지 확인합니다.

5단계: 기계 학습 변환으로 작업 추가 및 실행

이 단계에서는 기계 학습 변환을 사용하여 AWS Glue에서 작업을 추가하고 실행해 봅니다. demo-xform-dblp-acm 변환이 Ready for use(사용 준비 완료) 상태이면 ETL 작업에 사용할 수 있습니다.

1. AWS Glue 콘솔의 탐색 창에서 작업을 선택합니다.
2. 작업 추가를 선택하고, 마법사의 단계에 따라 생성된 스크립트로 ETL Spark 작업을 만듭니다. 변환에 대해 다음 속성 값을 선택하십시오.
   a. 이름으로 이 자습서의 예제 작업인 demo-etl-dblp-acm을 선택합니다.
   b. [IAM 역할(IAM role)]로는 Amazon S3 원본 데이터, 레이블 지정 파일 및 AWS Glue API 작업에 대한 권한이 있는 IAM 역할을 선택합니다. 자세한 내용은 AWS Glue Developer Guide의 Create an IAM Role for AWS Glue를 참조하세요.
   c. ETL 언어로 Scala를 선택합니다. 이것은 ETL 스크립트의 프로그래밍 언어입니다.
   d. 스크립트 파일 이름으로 demo-etl-dblp-acm을 선택합니다. 이것은 Scala 스크립트의 파일 이름입니다(작업 이름과 동일).
   e. 데이터 원본으로 dblp_acm_records_csv를 선택합니다. 선택한 데이터 원본이 기계 학습 변환의 데이터 원본 스키마와 일치해야 합니다.
   f. Transform type(변환 유형)으로 Find matching records(일치 레코드 찾기)를 선택하여 기계 학습 변환을 사용하는 작업을 만듭니다.
   g. Remove duplicate records(중복 레코드 제거)를 선택 취소합니다. 작성되는 출력 레코드에 match_id 필드가 하나 더 추가되므로 중복 레코드를 제거하지 않아도 됩니다.
   h. 변환으로 이 작업에 사용할 기계 학습 변환인 demo-xform-dblp-acm을 선택합니다.
   i. 데이터 대상에서 테이블 생성에서 다음 속성의 테이블을 만들도록 선택합니다.
      - [데이터 스토어 유형(Data store type)] - **Amazon S3**
      - [포맷(Format)] - **CSV**

- [압축 유형(Compression type) - **None**
- [대상 경로(Target path)] - 작업의 출력이 쓰이는 Amazon S3 경로(현재 콘솔 AWS 리전에서)

3. 작업 저장 및 스크립트 편집을 선택하여 스크립트 편집기 페이지를 표시합니다.
4. 대상 경로에 대한 작업 출력을 파티션 파일 하나에 쓰도록 하는 문을 추가하여 스크립트를 편집합니다. FindMatches 변환을 실행하는 문 바로 뒤에 이 문을 추가합니다. 이 문은 다음과 비슷합니다.

```
val single_partition = findmatches1.repartition(1)
```

출력을 .writeDynamicFrame(single_partion)으로 쓰도록 .writeDynamicFrame(findmatches1) 문을 수정해야 합니다.

5. 스크립트를 편집한 다음 저장을 선택합니다. 수정된 스크립트는 다음 코드와 비슷한 모양이지만 사용자 환경에 맞게 조정되어 있습니다.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.ml.FindMatches
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    // @type: DataSource
    // @args: [database = "demo-db-dblp-acm", table_name = "dblp_acm_records_csv",
 transformation_ctx = "datasource0"]
    // @return: datasource0
    // @inputs: []
    val datasource0 = glueContext.getCatalogSource(database = "demo-db-dblp-acm",
 tableName = "dblp_acm_records_csv", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
```

```
    // @type: FindMatches
    // @args: [transformId = "tfm-123456789012", emitFusion = false,
 survivorComparisonField = "<primary_id>", transformation_ctx = "findmatches1"]
    // @return: findmatches1
    // @inputs: [frame = datasource0]
    val findmatches1 = FindMatches.apply(frame = datasource0, transformId
 = "tfm-123456789012", transformationContext = "findmatches1",
 computeMatchConfidenceScores = true)


    // Repartition the previous DynamicFrame into a single partition.
    val single_partition = findmatches1.repartition(1)


    // @type: DataSink
    // @args: [connection_type = "s3", connection_options = {"path": "s3://aws-
glue-ml-transforms-data/sal"}, format = "csv", transformation_ctx = "datasink2"]
    // @return: datasink2
    // @inputs: [frame = findmatches1]
    val datasink2 = glueContext.getSinkWithFormat(connectionType =
 "s3", options = JsonOptions("""{"path": "s3://aws-glue-ml-transforms-
data/sal"}"""), transformationContext = "datasink2", format =
 "csv").writeDynamicFrame(single_partition)
    Job.commit()
  }
}
```

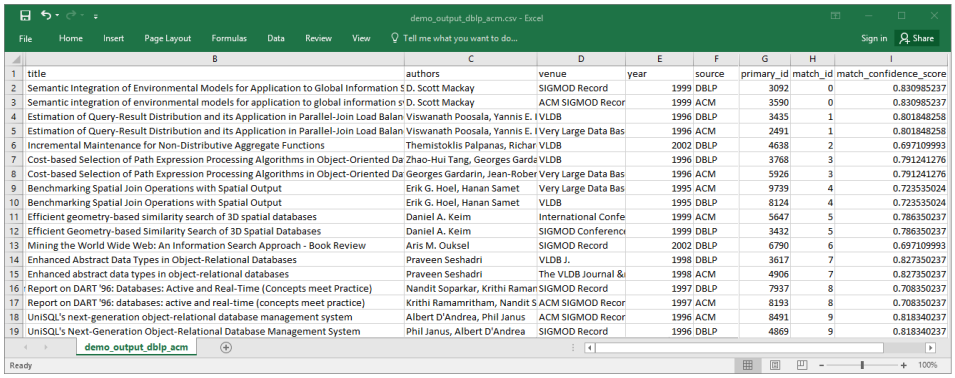
6. 작업 실행을 선택하여 작업을 실행하기 시작합니다. 작업 목록에서 작업의 상태를 확인합니다. 작업이 완료되면 ML transform(ML 변환)의 History(기록) 탭에 새로운 실행 ID 행이 ETL 작업 유형으로 추가됩니다.
7. 작업, History(기록) 탭으로 이동합니다. 이 창에는 작업 실행이 나열되어 있습니다. 작업 실행에 대한 자세한 내용은 로그를 참조하십시오. 완료된 작업의 실행 상태가 성공인지 확인합니다.

6단계: Amazon S3의 출력 데이터 확인

이 단계에서는 작업을 추가할 때 선택한 Amazon S3 버킷에서 작업 실행의 출력을 확인합니다. 출력 파일을 로컬 시스템에 다운로드하고, 일치하는 레코드가 식별되었는지 확인할 수 있습니다.

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. `demo-etl-dblp-acm` 작업의 대상 출력 파일을 다운로드합니다. 해당 파일을 스프레드시트 애플리케이션에서 엽니다. (파일을 제대로 열려면 파일 확장명 `.csv`를 추가해야 할 수 있음)

아래 그림은 Microsoft Excel의 출력을 발췌한 것입니다.



	title	authors	venue	year	source	primary_id	match_id	match_confidence_score
2	Semantic Integration of Environmental Models for Application to Global Information S	D. Scott Mackay	SIGMOD Record	1999	DBLP	3092	0	0.830985237
3	Semantic integration of environmental models for application to global information s	D. Scott Mackay	ACM SIGMOD Recor	1999	ACM	3590	0	0.830985237
4	Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balan	Viswanath Poosala, Yannis E. I	VLDB	1996	DBLP	3435	1	0.801848258
5	Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balan	Viswanath Poosala, Yannis E. I	Very Large Data Bas	1996	ACM	2491	1	0.801848258
6	Incremental Maintenance for Non-Distributive Aggregate Functions	Themistoklis Palpanas, Richar	VLDB	2002	DBLP	4638	2	0.697109993
7	Cost-based Selection of Path Expression Processing Algorithms in Object-Oriented Da	Zhao-Hui Tang, Georges Gard	VLDB	1996	DBLP	3768	3	0.791241276
8	Cost-based Selection of Path Expression Processing Algorithms in Object-Oriented Da	Georges Gardarin, Jean-Rober	Very Large Data Bas	1996	ACM	5926	3	0.791241276
9	Benchmarking Spatial Join Operations with Spatial Output	Erik G. Hoel, Hanan Samet	Very Large Data Bas	1995	ACM	9739	4	0.723535024
10	Benchmarking Spatial Join Operations with Spatial Output	Erik G. Hoel, Hanan Samet	VLDB	1995	DBLP	8124	4	0.723535024
11	Efficient geometry-based similarity search of 3D spatial databases	Daniel A. Keim	International Confe	1999	ACM	5647	5	0.786350237
12	Efficient Geometry-based Similarity Search of 3D Spatial Databases	Daniel A. Keim	SIGMOD Conferenc	1999	DBLP	3432	5	0.786350237
13	Mining the World Wide Web: An Information Search Approach - Book Review	Aris M. Ouksel	SIGMOD Record	2002	DBLP	6790	6	0.697109993
14	Enhanced Abstract Data Types in Object-Relational Databases	Praveen Seshadri	VLDB J.	1998	DBLP	3617	7	0.827350237
15	Enhanced abstract data types in object-relational databases	Praveen Seshadri	The VLDB Journal &	1998	ACM	4906	7	0.827350237
16	Report on DART '96: Databases: Active and Real-Time (Concepts meet Practice)	Nandit Soparkar, Krithi Raman	SIGMOD Record	1997	DBLP	7937	8	0.708350237
17	Report on DART '96: databases: active and real-time (concepts meet practice)	Krithi Ramamritham, Nandit S	ACM SIGMOD Recor	1997	ACM	8193	8	0.708350237
18	UniSQL's next-generation object-relational database management system	Albert D'Andrea, Phil Janus	ACM SIGMOD Recor	1996	ACM	8491	9	0.818340237
19	UniSQL's Next-Generation Object-Relational Database Management System	Phil Janus, Albert D'Andrea	SIGMOD Record	1996	DBLP	4869	9	0.818340237

데이터 원본과 대상 파일 모두 레코드가 4,911개 있습니다. 그러나 `Find matches` 변환은 출력에서 일치하는 레코드를 식별하기 위해 `match_id`라는 열을 하나 더 추가합니다. `match_id`가 동일한 행은 일치하는 레코드로 간주됩니다. `match_confidence_score`는 `Find matches`에서 찾은 일치 항목의 품질을 추정하는, 0에서 1 사이의 숫자입니다.

3. 일치하는 레코드를 보기 쉽도록 출력 파일을 `match_id`로 정렬합니다. 나머지 열의 값을 비교하여 `Find matches` 변환의 결과에 동의할 만한지 알아봅니다. 동의할 수 없으면 계속해서 레이블을 더 추가하여 변환을 학습시킬 수 있습니다.

   또한 `title` 등 다른 필드를 기준으로 파일을 정렬하여 제목이 비슷한 레코드의 `match_id`가 동일한지 확인할 수도 있습니다.

증분 일치 항목 찾기

일치 항목 찾기 기능을 사용하면 레코드에 공통된 고유 식별자가 없고 정확히 일치하는 필드가 없는 경우에도 데이터 집합에서 중복 레코드나 일치 레코드를 식별할 수 있습니다. 일치 항목 찾기 변환의 초기 릴리스에서는 단일 데이터 집합 내의 일치 레코드를 식별했습니다. 데이터 집합에 새 데이터를 추가하는 경우 정리된 기존 데이터 집합과 병합하고 병합된 전체 데이터 집합에 대해 일치를 다시 실행해야 했습니다.

증분 일치 기능을 사용하면 기존 일치 데이터 집합과 증분 레코드를 더 간단하게 일치시킬 수 있습니다. 기존 고객 데이터 집합과 잠재 고객 데이터를 일치시키려는 경우를 가정합니다. 증분 일치 기능을 사용하면 결과를 단일 데이터베이스나 테이블에 병합하여 잠재 고객 및 고객의 기존 데이터베이스와 수십만 명의 신규 잠재 고객을 유연성 있게 일치시킬 수 있습니다. 증분 일치 항목 찾기 최적화는 새 데이터 집합과 기존 데이터 집합 간에만 일치시켜 계산 시간을 단축하므로 비용도 절감됩니다.

증분 일치 사용법은 자습서: AWS Glue로 기계 학습 변환 생성에 설명된 일치 항목 찾기와 유사합니다. 이 주제에서는 증분 일치와의 차이점만 설명합니다.

자세한 내용은 증분 데이터 일치에 대한 블로그 게시물을 참조하세요.

증분 일치 작업 실행

다음 절차에서는 다음과 같이 가정합니다.

- 기존 데이터 세트를 first_records 테이블로 크롤링했습니다. first_records 데이터 세트는 일치하는 데이터 세트이거나 일치하는 작업의 출력이어야 합니다.
- AWS Glue 버전 2.0을 사용하여 일치 항목 찾기 변환을 생성하고 훈련시켰습니다. 증분 일치는 이 버전의 AWS Glue에서만 지원됩니다.
- ETL 언어는 Scala입니다. Python도 지원됩니다.
- `demo-xform`이라는 모델이 이미 생성되어 있습니다.

1. 증분 데이터 집합을 second_records 테이블로 크롤링합니다.
2. AWS Glue 콘솔의 탐색 창에서 작업을 선택합니다.
3. 작업 추가를 선택하고, 마법사의 단계에 따라 생성된 스크립트로 ETL Spark 작업을 만듭니다. 변환에 대해 다음 속성 값을 선택하십시오.
   a. 이름(Name)에서 demo-etl을 선택합니다.
   b. IAM 역할(IAM role)에서 Amazon S3 소스 데이터, 레이블 지정 파일, AWS Glue API 작업에 대한 권한이 있는 IAM 역할을 선택합니다.
   c. ETL 언어로 Scala를 선택합니다.
   d. 스크립트 파일 이름(Script file name)에서 demo-etl을 선택합니다. Scala 스크립트의 파일 이름입니다.
   e. 데이터 원본(Data source)에서 first_records를 선택합니다. 선택한 데이터 원본이 기계 학습 변환의 데이터 원본 스키마와 일치해야 합니다.
   f. Transform type(변환 유형)으로 Find matching records(일치 레코드 찾기)를 선택하여 기계 학습 변환을 사용하는 작업을 만듭니다.
   g. 증분 일치 옵션을 선택하고 데이터 원본(Data Source)에서 second_records라는 테이블을 선택합니다.
   h. 변환(Transform)에서 이 작업에 사용할 기계 학습 변환인 demo-xform을 선택합니다.
   i. 데이터 대상에 테이블 생성(Create tables in your data target) 또는 데이터 카탈로그 내 테이블 사용 및 데이터 대상 업데이트(Use tables in the data catalog and update your data target)를 선택합니다.
4. 작업 저장 및 스크립트 편집을 선택하여 스크립트 편집기 페이지를 표시합니다.

5. 작업 실행을 선택하여 작업을 실행하기 시작합니다.

시각적 작업에서 FindMatches 사용

AWS Glue Studio에서 FindMatches 변환을 사용하려면 FindMatches API를 간접 호출하는 사용자 지정 변환 노드를 사용하면 됩니다. 사용자 지정 변환을 사용하는 방법에 대한 자세한 내용은 사용자 지정 변환 생성을 참조하세요.

> **Note**
>
> 현재 FindMatches API는 `Glue 2.0`에서만 작동합니다. FindMatches API를 간접 호출하는 사용자 지정 변환을 포함하는 작업을 실행하려면 AWS Glue 버전이 작업 세부 정보 탭에서 `Glue 2.0`인지 확인합니다. AWS Glue의 버전이 `Glue 2.0`이 아닌 경우 작업이 런타임에 실패하고 'cannot import name 'FindMatches' from 'awsglueml.transforms' 오류 메시지가 표시됩니다.

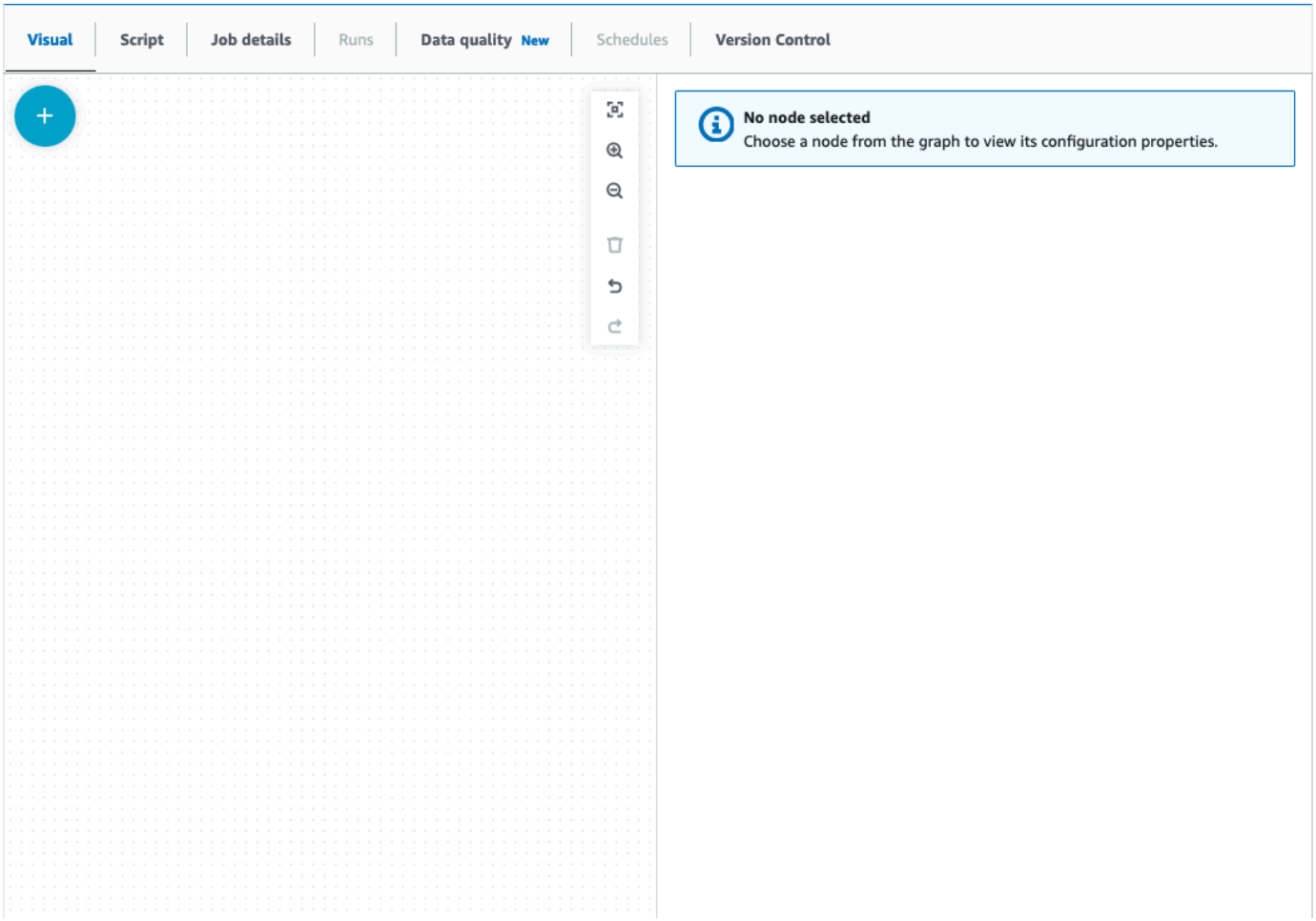
사전 조건

- 일치 항목 찾기 변환을 사용하려면 https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.
- 기계 학습 변환을 생성합니다. 생성되면 transformId가 생성됩니다. 아래 단계를 수행하려면 이 ID가 필요합니다. 기계 학습 변환을 생성하는 방법에 대한 자세한 내용은 기계 학습 변환 추가 및 편집을 참조하세요.

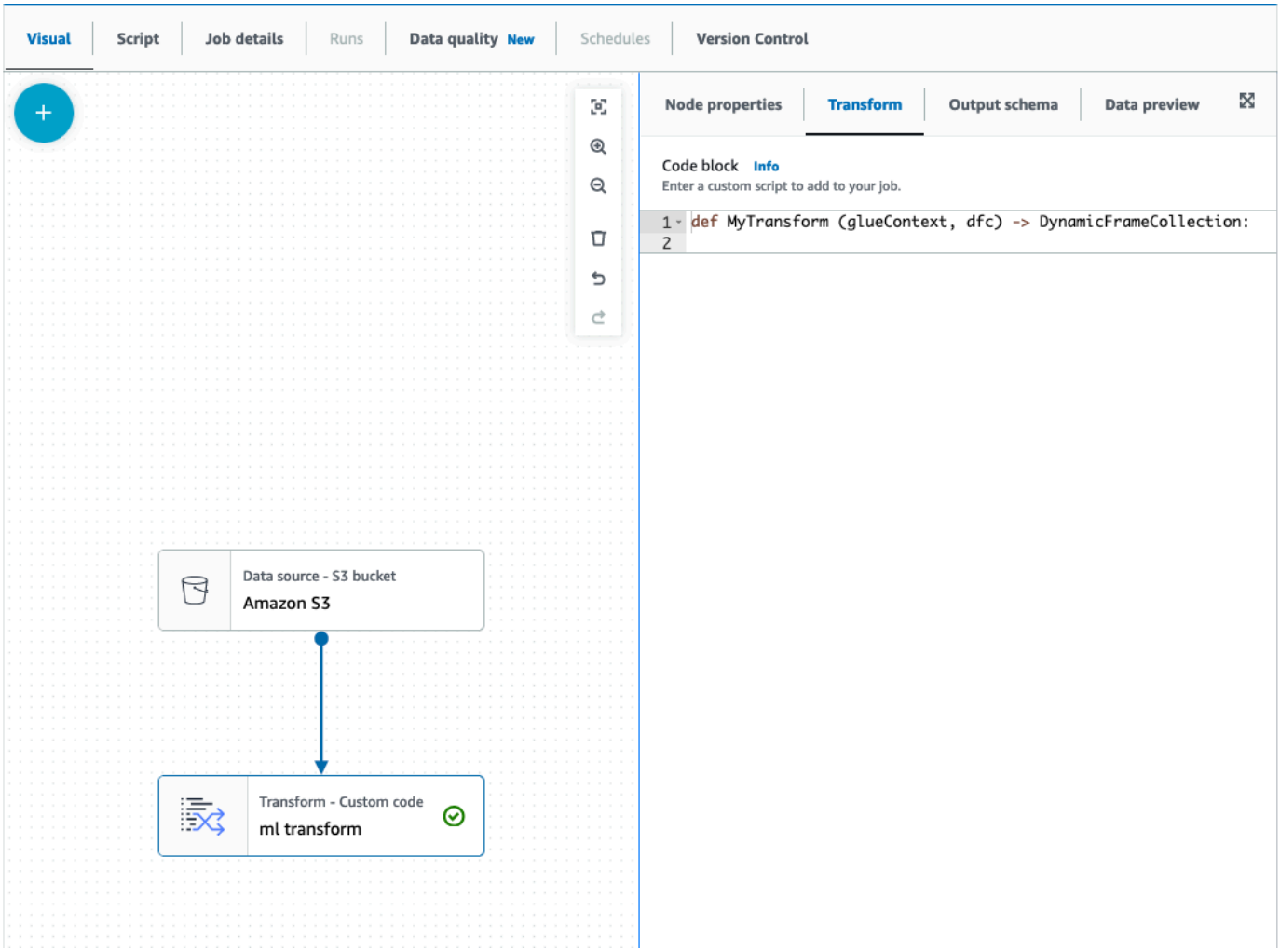
FindMatches 변환 추가

FindMatches 변환을 추가하려면:

1. AWS Glue Studio 작업 편집기에서 시각적 작업 그래프의 왼쪽 상단에 있는 십자 기호를 클릭하여 리소스 패널을 열고 데이터 탭을 선택하여 데이터 소스를 선택합니다. 이는 일치하는지 확인하려는 데이터 소스입니다.



2. 데이터 소스 노드를 선택한 다음 시각적 작업 그래프의 왼쪽 상단에 있는 십자 기호를 클릭하여 리소스 패널을 열고 '사용자 지정 변환'을 검색합니다. 사용자 지정 변환 노드를 선택하여 그래프에 추가합니다. 사용자 지정 변환이 데이터 소스 노드에 링크됩니다. 링크되지 않은 경우 사용자 지정 변환 노드를 클릭하고 노드 속성 탭을 선택한 다음 노드 상위 아래에서 데이터 소스를 선택할 수 있습니다.
3. 시각적 그래프에서 사용자 지정 변환 노드를 클릭한 다음 노드 속성 탭을 선택하고 사용자 지정 변환의 이름을 지정합니다. 시각적 그래프에서 변환 이름을 쉽게 식별할 수 있도록 변환의 이름을 바꾸는 것이 좋습니다.
4. 변환 탭을 선택합니다. 여기에서 코드 블록을 편집할 수 있습니다. 여기에서 FindMatches API를 간접 호출하는 코드를 추가할 수 있습니다.



코드 블록에는 미리 채워진 코드가 포함되어 있어 시작하는 데 도움이 됩니다. 미리 채워진 코드를 아래 템플릿으로 덮어씁니다. 템플릿에는 사용자가 제공할 수 있는 transformId의 자리 표시자가 있습니다.

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    dynf = dfc.select(list(dfc.keys())[0])
    from awsglueml.transforms import FindMatches
    findmatches = FindMatches.apply(frame = dynf, transformId = "<your id>")
    return(DynamicFrameCollection({"FindMatches": findmatches}, glueContext))
```

5. 시각적 그래프에서 사용자 지정 변환 노드를 클릭하고 시각적 작업 그래프의 왼쪽 상단에 있는 십자 기호를 클릭하여 리소스 패널을 연 후 '컬렉션에서 선택'을 검색합니다. 컬렉션에는 DynamicFrame 하나만 있으므로 기본 선택을 변경할 필요가 없습니다.

6. 계속해서 변환을 추가하거나 결과를 저장할 수 있습니다. 이제 일치 항목 찾기와 같은 추가된 열로 보강되었습니다. 다운스트림 변환에서 이러한 새 열을 참조하려면 변환 출력 스키마에 해당 열을 추가해야 합니다. 가장 쉬운 방법은 데이터 미리 보기 탭을 선택한 다음 스키마 탭에서 '데이터 미리보기 스키마 사용'을 선택하는 것입니다.
7. FindMatches를 사용자 지정하려면 'apply' 메서드에 전달할 추가 파라미터를 추가할 수 있습니다. FindMatches 클래스를 참조하세요.

FindMatches 증분 변환 추가

증분 일치의 경우 프로세스는 다음 차이점을 제외하고 FindMatches 변환 추가와 동일합니다.

- 사용자 지정 변환의 상위 노드 대신, 두 개의 상위 노드가 필요합니다.
- 첫 번째 상위 노드는 데이터 세트여야 합니다.
- 두 번째 상위 노드는 증분 데이터 세트여야 합니다.

  템플릿 코드 블록에서 `transformId`를 사용자의 `transformId`로 바꿉니다.

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    dfs = list(dfc.values())
    dynf = dfs[0]
    inc_dynf = dfs[1]
    from awsglueml.transforms import FindIncrementalMatches
    findmatches = FindIncrementalMatches.apply(existingFrame = dynf, incrementalFrame
 = inc_dynf,
                                       transformId = "<your id>")
    return(DynamicFrameCollection({"FindMatches": findmatches}, glueContext))
```

- 선택적 파라미터는 FindIncrementalMatches 클래스를 참조하세요.

Apache Spark 프로그램을 AWS Glue로 마이그레이션

Apache Spark는 대규모 데이터 세트에서 수행되는 분산 컴퓨팅 워크로드를 위한 오픈 소스 플랫폼입니다. AWS Glue에서는 Spark의 기능을 활용하여 ETL에 최적화된 환경을 제공합니다. Spark 프로그램을 AWS Glue로 마이그레이션하여 기능을 활용할 수 있습니다. AWS Glue에서는 Amazon EMR의 Apache Spark에서 기대하는 것과 동일한 성능 향상을 제공합니다.

Spark 코드 실행

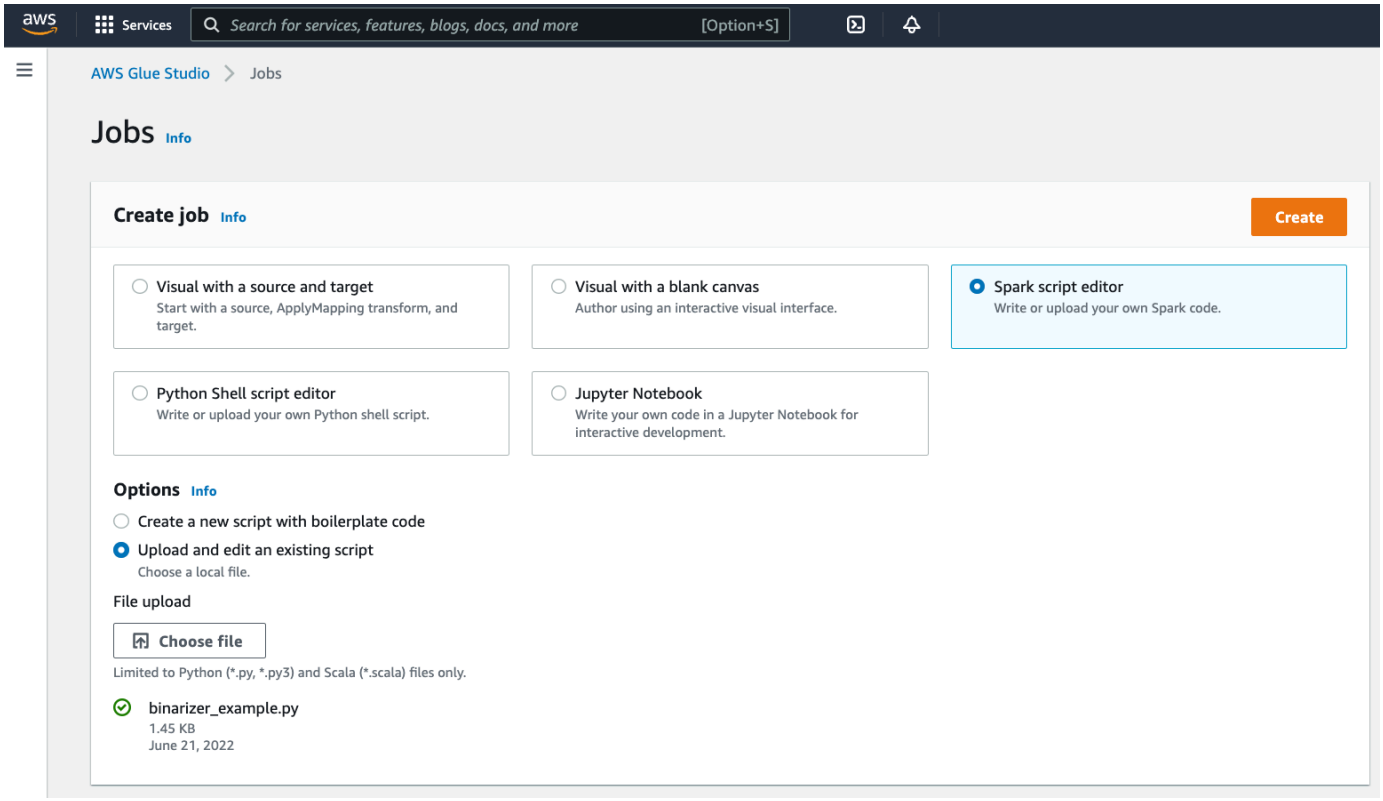
기본 Spark 코드는 기본 제공 AWS Glue 환경에서 실행될 수 있습니다. 스크립트는 대화형 세션에 적합한 워크플로우인 코드를 반복적으로 변경하여 개발되는 경우가 많습니다. 그러나 기존 코드는 AWS Glue 작업에서 실행되기에 더 적합하며 이를 통해 각 스크립트 실행에 대한 로그와 지표를 예약하고 일관되게 가져올 수 있습니다. 콘솔을 통해 기존 스크립트를 업로드하고 편집할 수 있습니다.

1. 스크립트의 소스를 확보합니다. 이 예에서는 Apache Spark 리포지토리의 스크립트 예를 사용합니다. 이진화 메서드 예
2. AWS Glue 콘솔에서 왼쪽 탐색 창을 확장하고 ETL > 작업(Jobs)을 선택합니다.

   작업 생성(Create job) 패널에서 Spark 스크립트 편집기(Spark script editor)를 선택합니다. 옵션(Options) 섹션이 나타납니다. 옵션(Options)에서 기존 스크립트 업로드 및 편집(Upload and edit an existing script)을 선택합니다.

   파일 업로드(File upload) 섹션이 나타납니다. 파일 업로드(File upload)에서 파일 선택(Choose file)을 클릭합니다. 시스템 파일 선택기가 나타납니다. `binarizer_example.py`를 저장한 위치로 이동하여 선택하고 선택을 확인합니다.

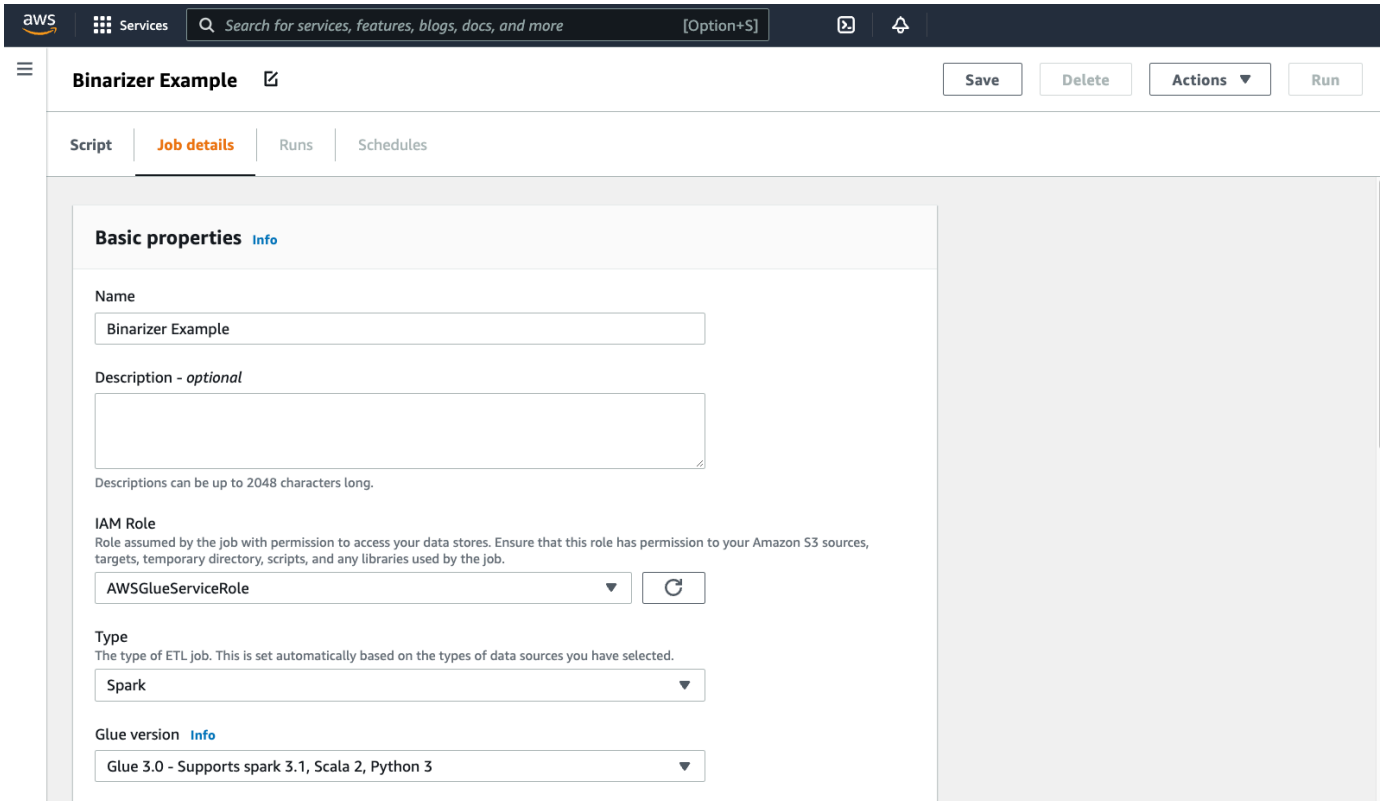
   생성(Create) 버튼이 작업 생성(Create job) 패널의 머리글에 나타납니다. 이 버튼을 클릭합니다.



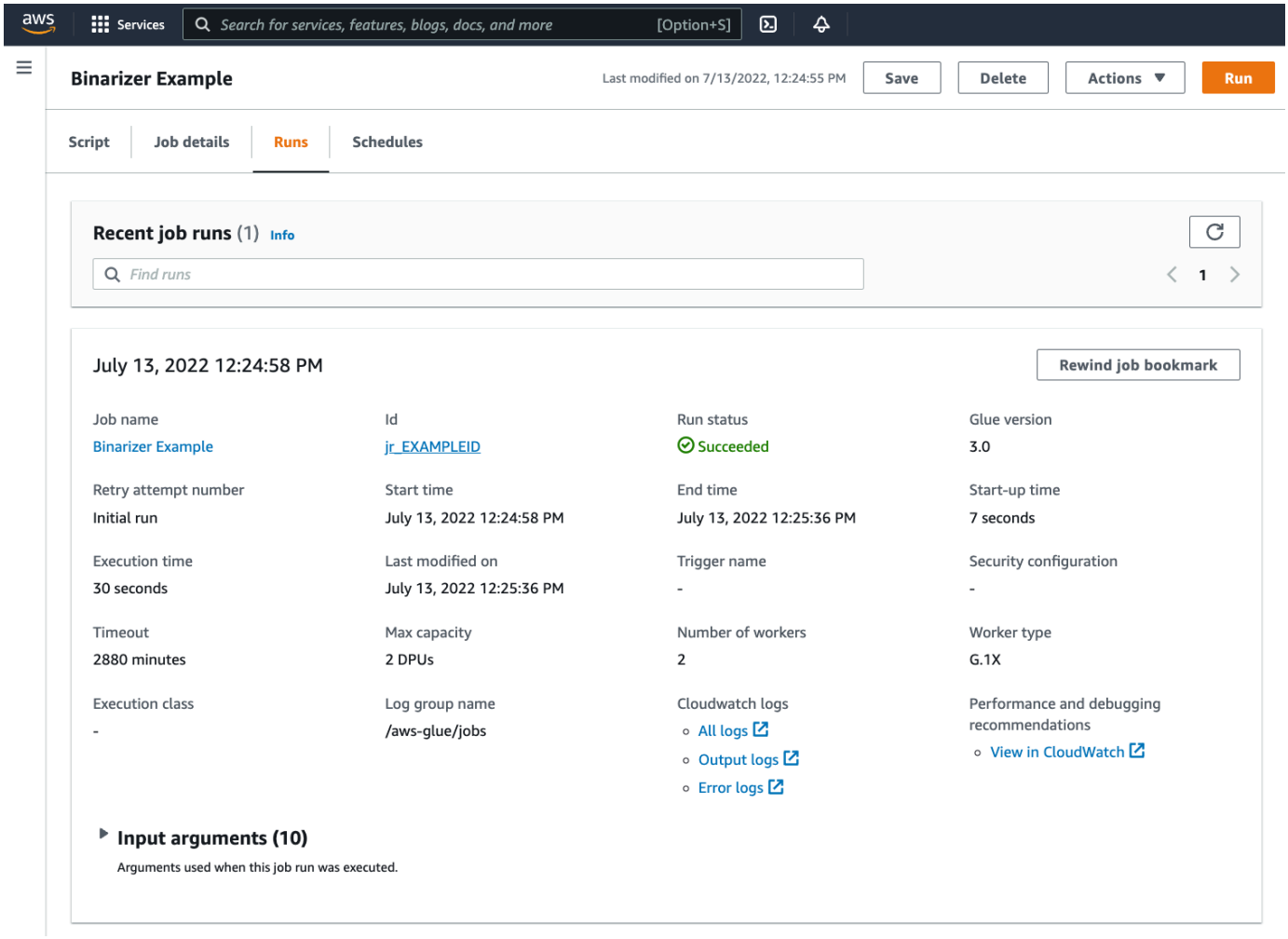
3. 브라우저가 스크립트 편집기로 이동합니다. 머리글에서 Job 세부 정보(Job details) 탭을 클릭합니다. 이름 및 IAM 역할을 설정합니다. AWS Glue IAM 역할에 대한 지침은 the section called “IAM 권한 설정” 섹션을 참조하세요.

   옵션 사항 - 요청된 작업자 수(Requested number of workers)를 2로, 재시도 횟수(Number of retries)를 1로 설정합니다. 이러한 옵션은 프로덕션 작업을 실행할 때 유용하지만 이 옵션을 끄면 기능을 테스트하는 동안 사용 환경이 간소화됩니다.

   제목 표시줄에서 저장(Save)을 클릭한 다음 실행(Run)을 클릭합니다.

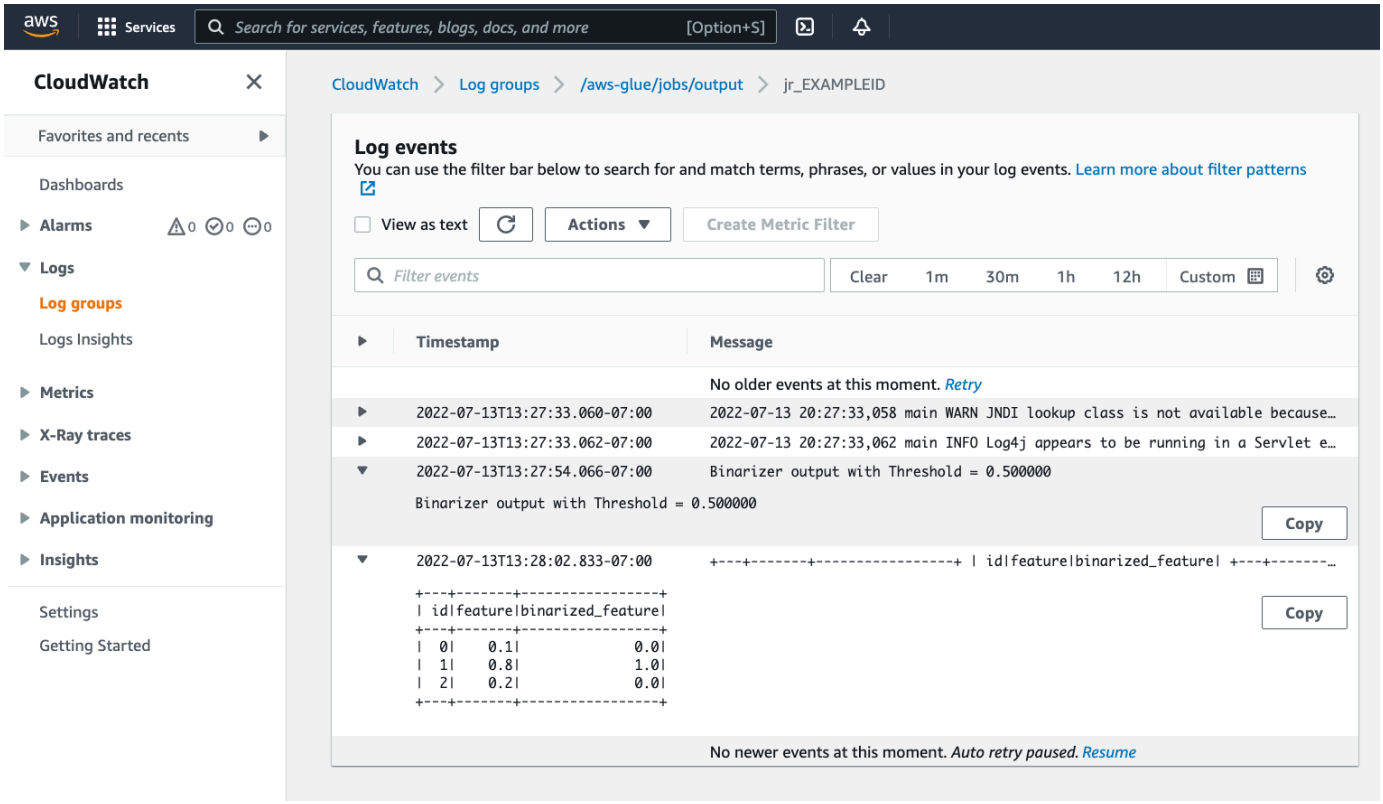


4. 실행(Runs) 탭으로 이동합니다. 작업 실행에 해당하는 패널이 표시됩니다. 몇 분간 기다리면 페이지가 자동으로 새로 고침되어 실행 상태(Run status) 아래에 성공(Succeeded)이라고 표시됩니다.



5. 출력을 검사하여 Spark 스크립트가 의도한 대로 실행되었는지 확인하려고 합니다. 이 Apache Spark 샘플 스크립트는 출력 스트림에 문자열을 기록해야 합니다. 성공적인 작업 실행에 대한 패널의 Cloudwatch 로그(Cloudwatch logs)에서 출력 로그(Output logs)로 이동하면 해당 문자열이 표시됩니다. `jr_`로 시작하는 ID 레이블에서 생성된 ID인 작업 실행 ID를 기록해 둡니다.

   그러면 CloudWatch 콘솔이 열리고 작업 실행 ID에 대한 로그 스트림의 내용으로 필터링된 기본 AWS Glue 로그 그룹 `/aws-glue/jobs/output`의 내용을 시각화하도록 설정됩니다. 각 작업자는 로그 스트림(Log streams) 아래 행으로 표시된 로그 스트림을 생성하게 됩니다. 한 작업자가 요청된 코드를 실행했어야 합니다. 올바른 작업자를 식별하려면 모든 로그 스트림을 열어야 합니다. 적합한 작업자를 찾으면 다음 이미지에 표시된 것과 같이 스크립트의 출력이 표시되어야 합니다.



Spark 프로그램을 마이그레이션하는 데 필요한 일반적인 절차

Spark 버전 지원 평가

AWS Glue 릴리즈 버전에 따라 AWS Glue 작업에 사용할 수 있는 Apache Spark와 Python의 버전이 정해집니다. AWS Glue 버전 및 the section called “AWS Glue 버전”의 지원 대상을 찾을 수 있습니다. 특정 AWS Glue 기능에 액세스하려면 최신 버전의 Spark와 호환되도록 Spark 프로그램을 업데이트해야 할 수 있습니다.

타사 라이브러리 포함

기존의 많은 Spark 프로그램에는 프라이빗 및 퍼블릭 아티팩트 모두에 대한 종속성이 있습니다. AWS Glue에서는 Scala 작업에 대한 JAR 스타일 종속성뿐만 아니라 Python 작업에 대한 휠 및 소스 퓨어-Python 종속성을 지원합니다.

Python - Python 종속성에 대한 자세한 정보는 the section called “Python 라이브러리” 섹션을 참조하세요.

공통 Python 종속성은 일반적으로 요청되는 Pandas 라이브러리를 포함한 AWS Glue 환경에서 제공됩니다. 종속성은 AWS Glue 버전 2.0+에 포함되어 있습니다. 제공된 모듈에 대한 자세한 내용을 알아보려면 the section called “AWS Glue에서 이미 제공되는 Python 모듈” 섹션을 참조하세요. 기본적으로

포함된 다른 버전의 종속성을 가진 작업을 제공해야 하는 경우 `--additional-python-modules`를 사용할 수 있습니다. 작업 인수에 대한 자세한 정보는 the section called “작업 파라미터” 섹션을 참조하세요.

`--extra-py-files` 작업 인수를 추가 Python 종속성에 제공할 수 있습니다. Spark 프로그램에서 작업을 마이그레이션하는 경우 이 파라미터는 PySpark의 `--py-files` 플래그와 기능적으로 동일하고 동일한 제한이 적용되므로 좋은 옵션입니다. `--extra-py-files` 파라미터에 대한 자세한 내용을 알아보려면 the section called “PySpark 네이티브 기능으로 Python 파일 포함” 섹션을 참조하세요.

새 작업의 경우 `--additional-python-modules` 작업 인수를 사용하여 Python 종속성을 관리할 수 있습니다. 이 인수를 사용하면 종속성 관리를 보다 철저하게 수행할 수 있습니다. 이 파라미터는 Amazon Linux 2와 호환되는 기본 코드 바인딩이 있는 종속성을 포함하여 Wheel 스타일 종속성을 지원합니다.

Scala

`--extra-jars` 작업 인수를 추가 Scala 종속성에 제공할 수 있습니다. 종속성은 Amazon S3에서 호스팅되어야 하며, 인수 값은 공백 없이 쉼표로 구분된 Amazon S3 경로 목록이어야 합니다. 종속성을 호스팅하고 구성하기 전에 다시 번들링하여 구성을 관리하는 것이 더 쉬울 수 있습니다. AWS Glue JAR 종속성에는 모든 JVM 언어에서 생성될 수 있는 Java 바이트 코드가 포함됩니다. Java와 같은 다른 JVM 언어를 사용하여 사용자 정의 종속성을 작성할 수 있습니다.

데이터 소스 자격 증명 관리

기존 Spark 프로그램에는 데이터 소스에서 데이터를 가져오기 위한 복잡한 구성 또는 사용자 정의 구성이 제공될 수 있습니다. 일반적인 데이터 소스 인증 흐름은 AWS Glue 연결에서 지원됩니다. AWS Glue 연결에 대한 자세한 정보는 데이터에 연결 섹션을 참조하세요.

AWS Glue 연결은 라이브러리에 대한 메서드 호출 및 AWS 콘솔에서 추가 네트워크 연결(Additional network connection) 설정과 같은 두 가지 기본 방법으로 작업을 다양한 유형의 데이터 저장소에 쉽게 연결할 수 있도록 합니다. 작업 내에서 AWS SDK를 호출하여 연결에서 정보를 검색할 수도 있습니다.

메서드 호출(Method calls) – AWS Glue 연결은 데이터 세트 에 대한 정보를 큐레이팅할 수 있는 서비스인 AWS Glue Data Catalog와 밀접히 통합되며, AWS Glue 연결과 상호 작용할 수 있는 메서드가 그것을 반영합니다. 재사용하려는 기존 인증 구성이 있으면 JDBC 연결의 경우 `GlueContext`의 `extract_jdbc_conf` 메서드를 통해 AWS Glue 연결 구성에 액세스할 수 있습니다. 자세한 정보는 the section called “extract_jdbc_conf” 섹션을 참조하세요.

콘솔 구성(Console configuration) – AWS Glue 작업은 관련된 AWS Glue 연결을 사용하여 Amazon VPC 서브넷에 대한 연결을 구성할 수 있습니다. 보안 자료를 직접 관리하는 경우 라우팅을 구성하는

AWS 콘솔에서 NETWORK 유형의 추가 네트워크 연결(Additional network connection)을 제공해야 할 수 있습니다. AWS Glue 연결에 대한 자세한 정보는 the section called “연결” 섹션을 참조하세요.

Spark 프로그램에 사용자 지정 또는 일반적이지 않은 인증 흐름이 있는 경우 보안 자료를 직접 관리해야 할 수 있습니다. AWS Glue 연결이 적합하지 않은 것 같은 경우 Secrets Manager에서 보안 자료를 안전하게 호스팅하고 작업에서 제공되는 boto3 또는 AWS SDK를 통해 해당 자료에 액세스 할 수 있습니다.

Apache Spark 구성

복잡한 마이그레이션은 Spark 구성을 변경하여 워크로드를 수용하는 경우가 많습니다. 최신 버전의 Apache Spark를 사용하면 런타임 구성을 SparkSession으로 설정할 수 있습니다. AWS Glue 3.0+ 작업에 런타임 구성을 설정하기 위해 수정할 수 있는 SparkSession이 제공됩니다. Apache Spark 구성. Spark 튜닝은 복잡하고, AWS Glue에서 모든 Spark 구성 설정에 대한 지원을 보장하지는 않습니다. 마이그레이션에 상당한 Spark 수준의 구성이 필요한 경우 지원 팀에 문의하세요.

사용자 지정 구성 설정

마이그레이션된 Spark 프로그램은 사용자 지정 구성을 사용하도록 설계될 수 있습니다. AWS Glue에서는 작업 인수를 통해 작업 및 작업 실행 수준에서 구성을 설정할 수 있습니다. 작업 인수에 대한 자세한 정보는 the section called “작업 파라미터” 섹션을 참조하세요. 라이브러리를 통해 작업 컨텍스트 내에서 작업 인수에 액세스할 수 있습니다. AWS Glue에서는 유틸리티 함수를 제공하여 작업에 설정된 인수와 작업 실행 시 설정된 인수 간에 일관된 보기를 유지합니다. Python의 the section called “getResolvedOptions 옵션 ” 섹션 및 Scala의 the section called “GlueArgParser” 섹션을 참조하세요.

Java 코드 마이그레이션

the section called “타사 라이브러리”에서 설명한 대로 종속성에는 Java 또는 Scala와 같은 JVM 언어로 생성된 클래스가 포함될 수 있습니다. 종속성에는 main 메서드가 포함될 수 있습니다. 종속성의 main 메소드를 AWS Glue Scala 작업에 대한 진입점으로 사용할 수 있습니다. 이를 통해 Java의 main 메서드를 작성하거나 자체 라이브러리 표준에 맞게 패키징된 main 메서드를 재사용할 수 있습니다.

종속성에서 main 메서드를 사용하려면 다음을 수행합니다. 기본 GlueApp 객체를 제공하는 편집 창의 내용을 삭제합니다. 종속성에 있는 클래스의 정규화된 이름을 키 --class를 통해 작업 인수로 제공합니다. 그러면 작업 실행을 트리거할 수 있습니다.

인수 AWS Glue의 순서나 구조가 main 메서드에 전달되도록 구성할 수 없습니다. 기존 코드에서 AWS Glue에 설정된 구성을 읽어야 하는 경우 이전 코드와 호환되지 않을 수 있습니다. getResolvedOptions를 사용하는 경우 이 메소드를 호출하기에 적합한 장소도 없을 것입니다. AWS Glue에 의해 생성된 기본 메서드에서 직접 종속성을 호출하는 것이 좋습니다. 이에 대한 예는 다음 AWS Glue ETL 스트립트에 표시됩니다.

```
import com.amazonaws.services.glue.util.GlueArgParser

object GlueApp {
  def main(sysArgs: Array[String]) {
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)

    // Invoke static method from JAR. Pass some sample arguments as a String[], one
 defined inline and one taken from the job arguments, using getResolvedOptions
    com.mycompany.myproject.MyClass.myStaticPublicMethod(Array("string parameter1",
 args("JOB_NAME")))

    // Alternatively, invoke a non-static public method.
    (new com.mycompany.myproject.MyClass).someMethod()
  }
}
```

AWS Glue에서 Ray 작업 사용

이 섹션에서는 AWS Glue for Ray 작업 사용에 대한 정보를 제공합니다. AWS Glue for Ray 스크립트 작성에 대한 자세한 내용은 the section called “AWS Glue for Ray” 섹션을 참조하세요.

주제

- AWS Glue for Ray 시작하기
- 지원되는 Ray 런타임 환경
- Ray 작업에서 작업자 고려
- Ray 작업에서 작업 파라미터 사용
- 지표를 통한 Ray 작업 모니터링

AWS Glue for Ray 시작하기

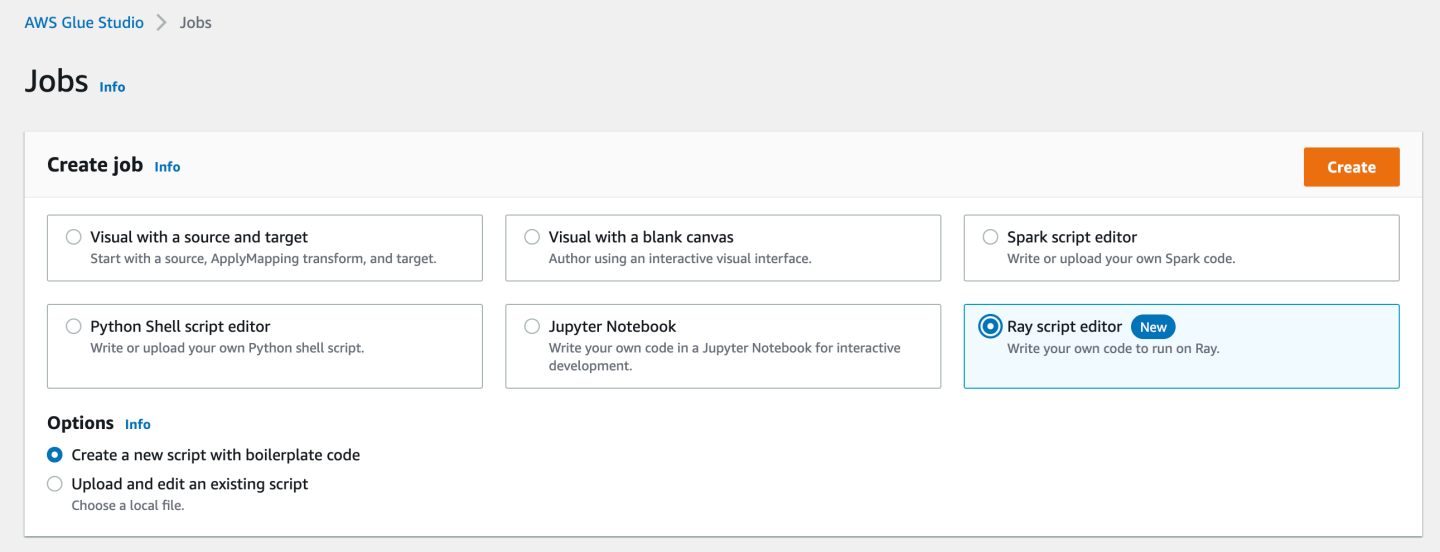
AWS Glue for Ray를 사용하려면 AWS Glue for Spark를 사용할 때와 동일한 AWS Glue 작업 및 대화형 세션을 사용합니다. AWS Glue 작업은 동일한 스크립트를 반복적으로 실행하도록 설계된 반면, 대화형 세션은 프로비저닝된 동일한 리소스에 대해 순차적으로 코드 조각을 실행할 수 있도록 설계되었습니다.

AWS Glue ETL과 Ray는 기본적으로 다르므로 스크립트에서 다른 도구, 기능 및 구성에 액세스할 수 있습니다. AWS Glue에서 관리하는 새로운 계산 프레임워크인 Ray는 아키텍처가 다르며 해당 기능을 설명하기 위해 다양한 어휘를 사용합니다. 자세한 내용은 Ray 설명서의 아키텍처 백서를 참조하세요.

> **Note**
>
> AWS Glue for Ray는 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤), 아시아 태평양(도쿄), 유럽(아일랜드)에서 사용 가능합니다.

AWS Glue Studio 콘솔에서 Ray 작업

AWS Glue Studio에서 작업을 생성할 때 AWS Glue Studio 콘솔의 작업 페이지에서 새 옵션(Ray 스크립트 편집기)을 선택할 수 있습니다. 콘솔에서 Ray 작업을 생성하려면 이 옵션을 선택합니다. 작업에 대한 자세한 내용과 사용 방법은 AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 섹션을 참조하세요.



AWS CLI 및 SDK에서 Ray 작업

AWS CLI에서 Ray 작업은 다른 작업과 동일한 SDK 작업 및 파라미터를 사용합니다. AWS Glue for Ray에는 특정 파라미터에 대한 새로운 값이 도입되었습니다. 작업 API에 대한 자세한 내용은 the section called “작업” 섹션을 참조하세요.

지원되는 Ray 런타임 환경

Spark 작업에서는 `GlueVersion`에서 AWS Glue for Spark 작업에 대해 사용할 수 있는 Apache Spark 및 Python 버전을 결정합니다. Python의 버전으로 Spark 유형의 작업에 대해 지원되는 버전을 확인할 수 있습니다. Ray 런타임 환경은 이렇게 구성되지 않습니다.

Ray 작업의 경우 `GlueVersion`을 `4.0` 이상으로 설정해야 합니다. 하지만 Ray 작업에서 사용할 수 있는 Ray, Python 및 추가 라이브러리 버전은 작업 정의의 `Runtime` 필드에 의해 결정됩니다.

Ray2.4 런타임 환경은 출시 후 최소 6개월 동안 사용할 수 있습니다. Ray가 빠르게 진화함에 따라 향후 런타임 환경 릴리스를 통해 Ray 업데이트 및 개선 사항을 통합할 수 있습니다.

유효값: Ray2.4

런타임 값	Ray 및 Python 버전
Ray2.4(AWS Glue 4.0 이상)	Ray 2.4.0 Python 3.9

추가 정보

- Ray 릴리스에서 AWS Glue를 함께 제공하는 릴리스 정보는 the section called “AWS Glue 버전” 섹션을 참조하세요.
- 런타임 환경에서 제공되는 Python 라이브러리는 the section called “Ray 작업과 함께 제공되는 모듈” 섹션을 참조하세요.

Ray 작업에서 작업자 고려

AWS Glue에서는 새로운 Graviton 기반 EC2 작업자 유형(Ray 작업에만 사용 가능)에서 Ray 작업을 실행합니다. Ray의 설계 목표였던 워크로드에 맞게 이러한 작업자를 적절하게 프로비저닝하기 위해 대부분의 작업자와는 다른 비율로, 메모리 리소스에 대한 컴퓨팅 리소스를 제공합니다. 이러한 리소스를 고려하기 위해 표준 데이터 처리 장치(DPU) 대신, 메모리에 최적화된 데이터 처리 장치(M-DPU)를 사용합니다.

- 1개의 M-DPU는 vCPU 4개와 32GB의 메모리에 해당합니다.
- 1개의 DPU가 vCPU 4개와 16GB 메모리에 해당합니다. AWS Glue에서 Spark 작업 및 해당 작업자로 리소스를 처리하려는 경우 DPU가 사용됩니다.

Ray 작업은 현재 하나의 작업자 유형(Z.2X)에 액세스할 수 있습니다. Z.2X 작업자는 2개의 M-DPU(vCPU 8개, 메모리 64GB)에 매핑되며 128GB의 디스크 공간을 보유합니다. Z.2X 시스템은 8개의 Ray 작업자(vCPU당 한 개)를 제공합니다.

계정에서 동시에 사용할 수 있는 M-DPU 수에는 서비스 할당량이 적용됩니다. AWS Glue 계정 제한에 대한 자세한 내용은 AWS Glue 엔드포인트 및 할당량을 참조하세요.

작업 정의에서 `--number-of-workers (NumberOfWorkers)`를 사용하여 Ray 작업에 사용할 수 있는 워커 노드 수를 지정합니다. 작업 API에서 Ray 값에 대한 자세한 내용은 the section called “작업” 섹션을 참조하세요.

`--min-workers` 작업 파라미터를 사용하여 Ray 작업에서 할당해야 하는 최소 작업자 수를 추가로 지정할 수 있습니다. 작업 파라미터에 대한 자세한 내용을 알아보려면 the section called “레퍼런스” 섹션을 참조하세요.

Ray 작업에서 작업 파라미터 사용

AWS Glue for Spark 작업과 동일한 방식으로 AWS Glue Ray 작업에 대한 인수를 설정합니다. AWS Glue API에 대한 자세한 내용은 the section called “작업”을 참조하십시오. 이 참조에 나열된 다양한 인수로 AWS Glue Ray 작업을 구성할 수 있습니다. 내 인수를 제공할 수도 있습니다.

Job Parameters(작업 파라미터) 제목 아래의 Job details(작업 세부 정보) 탭에서 콘솔을 통해 작업을 구성할 수 있습니다. 작업에서 `DefaultArguments`를 설정하거나 작업 실행에서 `Arguments`를 설정하여 AWS CLI를 통해 작업을 구성할 수도 있습니다. 기본 인수 및 작업 파라미터는 여러 번 실행해도 작업과 함께 유지됩니다.

예를 들어 다음은 특수 파라미터를 설정하기 위해 `--arguments`를 사용하여 작업을 실행하는 구문입니다.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py",--test-environment="true"'
```

인수를 설정한 후에는 환경 변수를 통해 Ray 작업 내에서 작업 파라미터에 액세스할 수 있습니다. 그러면 각 실행에 대한 작업을 구성할 수 있습니다. 환경 변수의 이름은 `--` 접두사가 없는 작업 인수 이름입니다.

예를 들어 이전 예제에서 변수 이름은 `scriptLocation` 및 `test-environment`입니다. 그런 다음 표준 라이브러리(`test_environment = os.environ.get('test-environment')`)에서 사용할 수 있는 메서드를 통해 인수를 검색합니다. Python으로 환경 변수에 액세스하는 방법에 대한 자세한 내용은 Python 설명서의 os 모듈을 참조하세요.

Ray 작업이 로그를 생성하는 방법 구성

기본적으로 Ray 작업은 CloudWatch와 Amazon S3로 전송되는 로그와 지표를 생성합니다. `--logging_configuration` 파라미터를 사용하여 로그 생성 방식을 변경할 수 있으며, 현재 이를 사용하여 Ray 작업에서 다양한 유형의 로그를 생성하지 못하게 할 수 있습니다. 이 파라미터는 변경하려는 로그/동작에 해당하는 키를 갖는 JSON 객체를 사용하며, 다음 키를 지원합니다.

- `CLOUDWATCH_METRICS` - 작업 상태를 시각화하는 데 사용할 수 있는 CloudWatch 지표 시리즈를 구성합니다. 지표에 대한 자세한 정보는 the section called “Ray 작업 지표” 섹션을 참조하세요.
- `CLOUDWATCH_LOGS` - 작업 실행 상태에 대한 Ray 애플리케이션 수준 세부 정보를 제공하는 CloudWatch 로그를 구성합니다. 로그에 대한 자세한 내용은 the section called “Ray 오류 해결” 섹션을 참조하세요.
- `S3` - AWS Glue가 Amazon S3에 작성하는 내용을 구성합니다. 주로 CloudWatch 로그와 유사한 정보이지만 로그 스트림이 아닌 파일로 작성합니다.

Ray 로깅 동작을 비활성화하려면 `{\"IS_ENABLED\": \"False\"}` 값을 제공합니다. 예를 들어, CloudWatch 지표와 CloudWatch 로그를 비활성화하려면 다음 구성을 제공합니다.

```
"--logging_configuration": "{\"CLOUDWATCH_METRICS\": {\"IS_ENABLED\": \"False\"},
 \"CLOUDWATCH_LOGS\": {\"IS_ENABLED\": \"False\"}}"
```

레퍼런스

Ray 작업은 Ray 작업 및 작업 실행에 대한 스크립트 환경을 설정하는 데 사용할 수 있는 다음과 같은 인수 이름을 인식합니다.

- `--logging_configuration` - Ray 작업에서 생성되는 다양한 로그의 생성을 중지하는 데 사용됩니다. 이러한 로그는 모든 Ray 작업에서 기본적으로 생성됩니다. 형식은 문자열 이스케이프 JSON 객체입니다. 자세한 내용은 the section called “Ray 작업이 로그를 생성하는 방법 구성” 단원을 참조하십시오.
- `--min-workers` - Ray 작업에 할당되는 작업자 노드의 최소 수입니다. 작업자 노드는 가상 CPU당 하나씩 여러 복제본을 실행할 수 있습니다. 형식: 정수 최소값: 0 최대값: 작업 정의의 `--number-of-workers` (`NumberOfWorkers`)에 지정된 값입니다. 워커 노드 고려 사항에 대한 자세한 내용은 the section called “Ray 작업에서 작업자 고려” 섹션을 참조하세요.
- `--object_spilling_config` - AWS Glue for Ray는 Ray의 객체 스토어로 사용 가능한 공간을 확장하는 방법으로 Amazon S3를 사용할 수 있도록 지원합니다. 이 동작을 활성화하려면 이 파라미터와 함께 객체 유출 JSON 구성 객체를 Ray에 제공할 수 있습니다. Ray 객체 유출 구성에 대한 자세한 내용은 Ray 설명서의 Object Spilling을 참조하세요. 형식: JSON 객체.

  AWS Glue for Ray는 디스크로 유출 또는 Amazon S3로 한 번에 유출하는 것만 지원합니다. 이러한 제한을 준수하는 유출에 대해 여러 위치를 제공할 수 있습니다. Amazon S3로 유출하는 경우 이 버킷의 작업에 IAM 권한도 추가해야 합니다.

CLI에서 JSON 객체를 구성으로 제공할 때는 JSON 객체를 문자열로 이스케이프 처리하여 문자열로 제공해야 합니다. 예를 들어 하나의 Amazon S3 경로로 유출하기 위한 문자열 값은 `"{\"type\": \"smart_open\", \"params\": {\"uri\":\"s3path\"}}"`과 같습니다. AWS Glue Studio에서는 이 파라미터를 추가 형식 지정 없이 JSON 객체로 제공합니다.

- `--object_store_memory_head` - Ray 헤드 노드의 Plasma 객체 스토어에 할당된 메모리입니다. 이 인스턴스는 작업자 복제본뿐만 아니라 클러스터 관리 서비스를 실행합니다. 값은 부팅 후 인스턴스에서 사용 가능한 메모리의 백분율을 나타냅니다. 메모리를 많이 사용하는 워크로드를 조정하는 데 이 파라미터를 사용합니다. 기본값은 대부분의 사용 사례에 적합합니다. 형식: 양의 정수 최소값: 1 최대값: 100

  Plasma에 대한 자세한 내용은 Ray 설명서의 Plasma 메모리 내 객체 스토어를 참조하세요.

- `--object_store_memory_worker` - Ray 작업자 노드의 Plasma 객체 스토어에 할당된 메모리입니다. 이러한 인스턴스는 작업자 복제본만 실행합니다. 값은 부팅 후 인스턴스에서 사용 가능한 메모리의 백분율을 나타냅니다. 이 파라미터는 메모리를 많이 사용하는 워크로드를 조정하는 데 사용됩니다. 기본값은 대부분의 사용 사례에 허용됩니다. 형식: 양의 정수 최소값: 1 최대값: 100

  Plasma에 대한 자세한 내용은 Ray 설명서의 Plasma 메모리 내 객체 스토어를 참조하세요.

- `--pip-install` - 설치할 Python 패키지 세트입니다. 이 인수를 사용하여 PyPI에서 패키지를 설치할 수 있습니다. 형식: 쉼표로 구분된 목록.

  PyPI 패키지 항목은 `package==version` 형식(대상 패키지의 PyPI 이름 및 버전)입니다. 항목은 단일 등호 =가 아닌 ==와 같은 Python 버전 매칭을 사용하여 패키지와 버전을 매칭합니다. 다른 버전 일치 연산자도 있습니다. 자세한 내용은 Python 웹사이트의 PEP 440을 참조하세요. `--s3-py-modules`에서 사용자 지정 모듈을 제공할 수도 있습니다.

- `--s3-py-modules` - Python 모듈 배포를 호스팅하는 Amazon S3 경로 세트입니다. 형식: 쉼표로 구분된 목록.

  이를 사용하여 Ray 작업에 자체 모듈을 배포할 수 있습니다. `--pip-install`에서 PyPI의 모듈을 제공할 수도 있습니다. AWS Glue ETL과 달리, 사용자 지정 배포는 pip를 통해 설정되지 않고 배포를 위해 Ray로 전달됩니다. 자세한 내용은 the section called “Ray 작업을 위한 추가 Python 모듈” 단원을 참조하십시오.

- `--working-dir` - Amazon S3에 호스팅된.zip 파일의 경로로, Ray 작업을 실행하는 모든 노드에 배포할 파일을 포함합니다. 형식: 문자열. 자세한 내용은 the section called “Ray 작업에 파일 제공” 단원을 참조하십시오.

지표를 통한 Ray 작업 모니터링

AWS Glue Studio 및 Amazon CloudWatch를 사용하여 Ray 작업을 모니터링할 수 있습니다. CloudWatch는 Ray를 통해 AWS Glue에서 원시 데이터를 수집하고 처리합니다. 이러한 데이터는 분석에 사용할 수 있습니다. 이러한 지표는 AWS Glue Studio 콘솔에서 시각화되므로 작업을 실행하면서 모니터링할 수 있습니다.

AWS Glue 모니터링 방법에 대한 일반적인 개요는 the section called “CloudWatch 지표 사용” 섹션을 참조하세요. AWS Glue에서 게시한 CloudWatch 지표를 사용하는 방법에 대한 일반적인 개요는 the section called “CloudWatch를 사용하여 모니터링” 섹션을 참조하세요.

AWS Glue 콘솔에서 Ray 작업 모니터링

작업 실행에 대한 세부 정보 페이지의 실행 세부 정보 섹션 아래에서 사용 가능한 작업 지표를 시각화하는 사전 구축된 집계된 그래프를 볼 수 있습니다. AWS Glue Studio에서는 모든 작업 실행 시 작업 지표를 Cloud Watch에 전송합니다. 이를 통해 클러스터 및 작업의 프로파일을 구축하고 각 노드에 대한 세부 정보에 액세스할 수 있습니다.

사용 가능한 지표에 대한 자세한 내용은 the section called “Ray 작업 실행에 대한 Amazon CloudWatch 지표 보기” 섹션을 참조하십시오.

CloudWatch의 Ray 작업 지표 개요

CloudWatch에서 세부 모니터링이 활성화되면 Ray 지표를 게시합니다. 지표는 `Glue/Ray` CloudWatch 네임스페이스에 게시됩니다.

- 인스턴스 지표

  작업에 할당된 인스턴스의 CPU, 메모리 및 디스크 사용률에 대한 지표를 게시합니다. 이러한 지표는 `ExecutorId`, `ExecutorType`, `host`와 같은 기능으로 식별됩니다. 이러한 지표는 표준 Linux CloudWatch 에이전트 지표의 하위 세트입니다. CloudWatch 설명서에서 지표 이름 및 기능에 대한 정보를 확인할 수 있습니다. 자세한 내용은 CloudWatch 에이전트가 수집하는 지표를 참조하세요.

- Ray 클러스터 지표

  스크립트를 실행하는 Ray 프로세스에서 이 네임스페이스로 지표를 전달한 후 사용자에게 가장 중요한 지표를 제공합니다. 사용 가능한 지표는 Ray 버전에 따라 다를 수 있습니다. 작업이 실행 중인 Ray 버전에 대한 자세한 내용은 the section called “AWS Glue 버전” 섹션을 참조하세요.

  Ray는 인스턴스 수준에서 지표를 수집합니다. 또한 작업 및 클러스터에 대한 지표도 제공합니다. Ray의 기본 지표 전략에 대한 자세한 내용은 Ray 설명서의 Metrics를 참조하세요.

> **Note**
>
> AWS Glue ETL 작업에만 사용되는 `Glue/Job Metrics/` 네임스페이스에는 Ray 지표를 게시하지 않습니다.

AWS Glue에서 Python 쉘 작업에 대한 작업 속성 구성

AWS Glue에서 Python 셸 작업을 사용하여 Python 스크립트를 셸로 실행할 수 있습니다. Python 쉘 작업을 사용하면 Python 3.6 또는 Python 3.9와 호환되는 스크립트를 실행할 수 있습니다.

> **Note**
>
> Pyshell v3.6에 대한 지원은 2026년 3월 1일에 종료됩니다. 워크로드를 마이그레이션하려면 AWS Glue Python 쉘 작업에서 마이그레이션을 참조하세요. Python 쉘 v3.9를 계속 사용하려면 Python 쉘 3.6에서 Python 쉘 3.9로 마이그레이션 섹션을 참조하세요.

주제

- 제한 사항
- Python 셸 작업에 대한 작업 속성 정의
- Python 셸 작업에 지원되는 라이브러리
- 자체 Python 라이브러리 제공
- AWS Glue에서 Python 쉘 작업과 함께 AWS CloudFormation 사용
- Python 쉘 3.6에서 Python 쉘 3.9로 마이그레이션
- AWS Glue Python 쉘 작업에서 마이그레이션

제한 사항

Python 셸 작업의 다음 제한 사항에 유의합니다.

- Python 셸 작업에는 작업 북마크를 사용할 수 없습니다.
- Python 3.9 이상에서는 어떤 Python 라이브러리도 `.egg` 파일로 패키징할 수 없습니다. 대신 `.whl`를 사용합니다.
- S3 데이터의 임시 복사본에 대한 제한 사항 때문에 이 `--extra-files` 옵션을 사용할 수 없습니다.

Python 셸 작업에 대한 작업 속성 정의

이 섹션에서는 AWS Glue Studio에서 또는 AWS CLI를 사용하여 작업 속성을 정의하는 것을 설명합니다.

AWS Glue Studio

AWS Glue Studio에서 Python 셸 작업을 정의할 때 다음 속성 중 일부를 입력합니다.

IAM 역할

사용된 리소스에 대한 권한 부여에 사용되는 AWS Identity and Access Management(IAM) 역할을 지정하여 작업을 실행하고 데이터 스토어에 액세스합니다. AWS Glue 작업을 실행하는 권한에 대한 자세한 내용은 AWS Glue의 Identity and Access Management 단원을 참조하십시오.

유형

Python shell(Python 셸)을 선택하여 `pythonshell` 작업 명령으로 Python 스크립트를 실행합니다.

Python 버전

Python 버전을 선택합니다. 기본값은 Python 3.9입니다. 유효한 버전은 Python 3.6과 Python 3.9입니다.

공통 분석 라이브러리 로드(권장)

Python 셸에 Python 3.9용 공통 라이브러리를 포함하려면 이 옵션을 선택합니다.

라이브러리가 사용자 지정 라이브러리이거나 사전 설치된 라이브러리와 충돌하는 경우 공통 라이브러리를 설치하지 않을 수 있습니다. 하지만 공통 라이브러리 외에 추가 라이브러리를 설치할 수 있습니다.

이 옵션을 선택하는 경우 `library-set` 옵션이 `analytics`로 설정됩니다. 이 옵션을 선택 취소하는 경우 `library-set` 옵션이 `none`으로 설정됩니다.

스크립트 파일 이름 및 스크립트 경로

스크립트의 코드는 작업 절차 논리를 정의합니다. Amazon Simple Storage Service(Amazon S3)에서 스크립트 이름과 위치를 제공합니다. 경로의 스크립트 디렉터리와 동일한 이름의 파일이 없다는 것을 확인합니다. 스크립트 사용에 대한 자세한 내용은 AWS Glue 프로그래밍 안내서 단원을 참조하십시오.

Script

스크립트의 코드는 작업 절차 논리를 정의합니다. Python 3.6 또는 Python 3.9에서 스크립트를 코딩할 수 있습니다. AWS Glue Studio에서 스크립트를 편집할 수 있습니다.

데이터 처리 단위

이 작업이 실행될 때 할당할 수 있는 AWS Glue 데이터 처리 단위(DPU)의 최대 수입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금을 참조하십시오.

값을 0.0625 또는 1로 설정할 수 있습니다. 기본값은 0.0625입니다. 어느 경우든 인스턴스의 로컬 디스크는 20GB가 됩니다.

CLI

다음 예와 같이 AWS CLI를 사용하여 [Python 셸(Python shell)] 작업을 생성할 수도 있습니다.

```
aws glue create-job --name python-job-cli --role Glue_DefaultRole
    --command '{"Name" :  "pythonshell", "PythonVersion": "3.9", "ScriptLocation" :
"s3://amzn-s3-demo-bucket/scriptname.py"}'
    --max-capacity 0.0625
```

> **Note**
>
> `--glue-version` 파라미터는 AWS Glue 쉘 작업에 적용되지 않으므로 AWS Glue 버전을 지정할 필요가 없습니다. 지정된 버전은 모두 무시됩니다.

AWS CLI를 사용하여 생성된 작업은 Python 3으로 기본 설정됩니다. 유효한 Python 버전은 3(3.6에 해당)과 3.9입니다. Python 3.6을 지정하려면 이 튜플을 `--command` 파라미터 `"PythonVersion":"3"`에 추가합니다.

Python 3.9를 지정하려면 이 튜플을 `--command` 파라미터 `"PythonVersion":"3.9"`에 추가합니다.

Python 셸 작업에서 사용하는 최대 용량을 설정하려면 `--max-capacity` 파라미터를 제공합니다. Python 셸 작업에서 `--allocated-capacity` 파라미터는 사용할 수 없습니다.

Python 셸 작업에 지원되는 라이브러리

Python 3.9를 사용하는 Python 셸에서 필요에 따라 사전 패키징된 라이브러리 세트를 사용하도록 라이브러리 세트를 선택할 수 있습니다. `library-set` 옵션을 사용하여 라이브러리 세트를 선택할 수 있습니다. 유효한 값은 `analytics` 및 `none`입니다.

Python 셸 작업의 실행 환경은 다음 라이브러리를 지원합니다.

Python 버전	Python 3.6	Python 3.9	
라이브러리 세트	해당 사항 없음	분석	none
avro		1.11.0	
awscli	116.242	1.23.5	1.23.5
awswrangler		2.15.1	
botocore	1.12.232	1.24.21	1.23.5
boto3	1.9.203	1.21.21	
elasticsearch		8.2.0	
numpy	1.16.2	1.22.3	
pandas	0.24.2	1.4.2	
psycopg2		2.9.3	
pyathena		2.5.3	
PyGreSQL	5.0.6		
PyMySQL		1.0.2	
pyodbc		4.0.32	
pyorc		0.6.0	
redshift-connector		2.0.907	

Python 버전	Python 3.6	Python 3.9	
요청	2.22.0	2.27.1	
scikit-learn	0.20.3	1.0.2	
scipy	1.2.1	1.8.0	
SQLAlchemy		1.4.36	
s3fs		2022.3.0	

NumPy 라이브러리는 공학용 컴퓨팅의 Python 셸 작업에서 사용할 수 있습니다. 자세한 내용은 NumPy를 참조하십시오. 다음 예제는 Python 셸 작업에 사용할 수 있는 NumPy 스크립트를 보여줍니다. 이 스크립트는 "Hello world" 및 몇 가지 수학적 계산 결과를 프린트합니다.

```
import numpy as np
print("Hello world")

a = np.array([20,30,40,50])
print(a)

b = np.arange( 4 )

print(b)

c = a-b

print(c)

d = b**2

print(d)
```

자체 Python 라이브러리 제공

PIP 사용

Python 3.9를 사용하는 Python 셸을 사용하면 작업 수준에서 추가 Python 모듈 또는 다른 버전을 제공할 수도 있습니다. `--additional-python-modules` 옵션을 쉼표로 구분된 Python 모듈 목록과 함께 사용하여 새 모듈을 추가하거나 기존 모듈의 버전을 변경할 수 있습니다. Python 쉘 작업을 사용할 때는 이 파라미터로 Amazon S3에 호스팅된 사용자 지정 Python 모듈을 제공할 수 없습니다.

예를 들어 새 `scikit-learn` 모듈을 업데이트하거나 추가하려면 `"--additional-python-modules", "scikit-learn==0.21.3"` 키 및 값을 사용합니다.

AWS Glue는 Python 패키지 설치 프로그램(pip3)을 사용하여 추가 모듈을 설치합니다. `--additional-python-modules` 값 내에 추가 pip3 옵션을 전달할 수 있습니다. 예를 들어 `"scikit-learn==0.21.3 -i https://pypi.python.org/simple/"`입니다. pip3의 비호환성 또는 제한 사항이 적용됩니다.

> **Note**
>
> 향후에 비호환성을 방지하려면 Python 3.9용으로 빌드된 라이브러리를 사용하는 것이 좋습니다.

Egg 또는 Whl 파일 사용

이미 하나 이상의 `.egg` 또는 `.whl` 파일로 패키징된 하나 이상의 Python 라이브러리가 있을 수 있습니다. 그렇다면 다음 예와 같이 AWS Command Line Interface(AWS CLI)를 사용하여 `"--extra-py-files"` 플래그 하에서 작업에 이를 지정할 수 있습니다.

```
aws glue create-job --name python-redshift-test-cli --role role --command '{"Name" :
 "pythonshell", "ScriptLocation" : "s3://MyBucket/python/library/redshift_test.py"}'
    --connections Connections=connection-name --default-arguments '{"--extra-py-
files" : ["s3://amzn-s3-demo-bucket/EGG-FILE", "s3://amzn-s3-demo-bucket/WHEEL-FILE"]}'
```

Python 라이브러리에서 `.egg` 파일 또는 `.whl` 파일을 생성하는 방법을 모르는 경우 다음 단계를 따르십시오. 이 예제는 macOS, Linux 및 Windows Subsystem for Linux(WSL)에 적용됩니다.

Python .egg 또는 .whl 파일을 만들려면

1. Virtual Private Cloud(VPC)에서 Amazon Redshift 클러스터를 생성하고 일부 데이터를 테이블에 추가합니다.
2. 클러스터를 만드는 데 사용한 VPC-SecurityGroup-Subnet 조합에 대해 AWS Glue 연결을 만듭니다. 연결이 성공인지 테스트합니다.
3. `redshift_example`이라는 디렉터리를 생성하고 `setup.py`라는 파일을 생성합니다. `setup.py`에 다음 코드를 붙여넣습니다.

```
from setuptools import setup

setup(
    name="redshift_module",
    version="0.1",
    packages=['redshift_module']
)
```

4. `redshift_example` 디렉터리에 `redshift_module` 디렉터리를 만듭니다. `redshift_module` 디렉터리에서 `__init__.py` 및 `pygresql_redshift_common.py`라는 파일을 만듭니다.
5. `__init__.py` 파일은 비워둡니다. `pygresql_redshift_common.py`에 다음 코드를 붙여넣습니다. *`port`*, *`db_name`*, *`user`* 및 *`password_for_user`*를 Amazon Redshift 클러스터에 해당하는 세부 정보로 바꿉니다. *`table_name`*을 Amazon Redshift의 테이블 이름으로 바꿉니다.

```
import pg


def get_connection(host):
    rs_conn_string = "host=%s port=%s dbname=%s user=%s password=%s" % (
        host, port, db_name, user, password_for_user)

    rs_conn = pg.connect(dbname=rs_conn_string)
    rs_conn.query("set statement_timeout = 1200000")
    return rs_conn


def query(con):
    statement = "Select * from table_name;"
    res = con.query(statement)
    return res
```

6. 아직 이동하지 않은 경우 `redshift_example` 디렉터리로 변경합니다.
7. 다음 중 하나를 수행합니다.

   - `.egg` 파일을 생성하려면 다음 명령을 실행합니다.

     ```
     python setup.py bdist_egg
     ```

   - `.whl` 파일을 생성하려면 다음 명령을 실행합니다.

     ```
     python setup.py bdist_wheel
     ```

8. 이전 명령에 필요한 종속성을 설치합니다.
9. 해당 명령은 `dist` 디렉터리에서 파일을 생성합니다.

   - egg 파일을 만든 경우, 이름은 `redshift_module-0.1-py2.7.egg`입니다.
   - wheel 파일을 만든 경우, 이름은 `redshift_module-0.1-py2.7-none-any.whl`입니다.

   이 파일을 Amazon S3에 업로드합니다.

   이 예제에서 업로드된 파일 경로는 *s3://amzn-s3-demo-bucket/EGG-FILE* 또는 *s3://amzn-s3-demo-bucket/WHEEL-FILE*입니다.
10. AWS Glue 작업의 스크립트로 사용할 Python 파일을 생성하고 다음 코드를 파일에 추가합니다.

    ```
    from redshift_module import pygresql_redshift_common as rs_common

    con1 = rs_common.get_connection(redshift_endpoint)
    res = rs_common.query(con1)

    print "Rows in the table cities are: "

    print res
    ```

11. 이전 파일을 Amazon S3에 업로드합니다. 이 예제에서 업로드된 파일 경로는 *s3://amzn-s3-demo-bucket/scriptname.py*입니다.
12. 이 스크립트를 사용하여 Python 셸 작업을 생성합니다. AWS Glue 콘솔의 [작업 속성(Job properties)] 페이지에서 Python 라이브러리 경로(Python library path) 상자에 `.egg/.whl` 파일의

경로를 지정합니다. 여러 `.egg/.whl` 파일 및 Python 파일이 있으면 이 상자에 쉼표로 구분된 목록을 제공합니다.

`.egg` 파일을 수정하거나 이름을 바꿀 때 파일 이름은 “python setup.py bdist_egg” 명령으로 생성된 기본 이름을 사용하거나 Python 모듈 명명 규칙을 준수해야 합니다. 자세한 내용은 Python 코드 스타일 가이드를 참조하십시오.

AWS CLI를 사용하여 다음 예와 같이 명령으로 작업을 생성하십시오.

```
aws glue create-job --name python-redshift-test-cli --role Role --command
 '{"Name" :  "pythonshell", "ScriptLocation" : "s3://amzn-s3-demo-bucket/
scriptname.py"}'
     --connections Connections="connection-name" --default-arguments '{"--extra-
py-files" : ["s3://amzn-s3-demo-bucket/EGG-FILE", "s3://amzn-s3-demo-bucket/WHEEL-
FILE"]}'
```

작업이 실행되면 스크립트는 Amazon Redshift 클러스터의 *table_name* 테이블에 생성된 행을 인쇄합니다.

AWS Glue에서 Python 쉘 작업과 함께 AWS CloudFormation 사용

AWS Glue에서 Python 쉘 작업과 함께 AWS CloudFormation을 사용할 수 있습니다. 다음은 예제입니다.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Python39Job:
    Type: 'AWS::Glue::Job'
    Properties:
      Command:
        Name: pythonshell
        PythonVersion: '3.9'
        ScriptLocation: 's3://bucket/location'
      MaxRetries: 0
      Name: python-39-job
      Role: RoleName
```

Python 셸 작업 출력에 대한 Amazon CloudWatch Logs 그룹은 `/aws-glue/python-jobs/output`입니다. 오류의 경우 로그 그룹 `/aws-glue/python-jobs/error`를 확인합니다.

Python 쉘 3.6에서 Python 쉘 3.9로 마이그레이션

Python 쉘 작업을 최신 AWS Glue 버전으로 마이그레이션:

1. AWS Glue 콘솔(https://console.aws.amazon.com/glue/)에서 기존 Python 쉘 작업을 선택합니다.
2. 작업 세부 정보 탭에서 Python 버전을 `Python 3.9`로 설정하고 저장을 선택합니다.
3. 작업 스크립트가 Python 3.9와 호환되고 성공적으로 실행되는지 확인합니다.

AWS Glue Python 쉘 작업에서 마이그레이션

AWS는 2018년 고객에게 중소 규모의 ETL 작업에서 손쉽게 Python 스크립트를 실행하고 SQL 쿼리를 트리거할 수 있는 방법을 제공하기 위해 AWS Glue Python 쉘 작업을 출시했습니다. 그러나 이제 PythonShell에서 현재 실행 중인 워크로드를 해결하기 위한 보다 현대적이고 유연한 옵션이 있습니다. 이 항목에서는 사용 가능한 최신 기능을 활용하기 위해 워크로드를 AWS Glue Python 쉘 작업에서 이러한 대체 옵션 중 하나로 마이그레이션하는 방법을 설명합니다.

이 항목에서는 AWS Glue Python 쉘 작업에서 대체 옵션으로 마이그레이션하는 방법에 관해 설명합니다.

워크로드를 AWS Glue Spark 작업으로 마이그레이션

AWS Glue Spark 및 PySpark 작업을 사용하면 분산 방식으로 워크로드를 실행할 수 있습니다. AWS Glue Python 쉘 작업과 AWS Glue Spark 작업은 모두 동일한 플랫폼에서 실행되므로 쉽게 마이그레이션할 수 있으며 AWS Glue 워크플로, AWS Glue 트리거, AWS Glue의 Amazon EventBridge 통합, \ PIP 기반 패키지 설치 등 Python Shell 작업과 함께 사용 중인 기존 AWS Glue 기능을 계속 사용할 수 있습니다.

그러나 AWS Glue Spark 작업은 Spark 워크로드를 실행하도록 설계되었으며 최소 워커 수는 2개입니다. 스크립트를 수정하지 않고 Python 쉘 작업에서 마이그레이션하는 경우 실제로 1개의 워커만 사용되며 다른 워커는 유휴 상태로 유지됩니다. 이로 인해 비용이 증가합니다.

효율성을 높이려면 Python 작업 스크립트를 다시 작성하여 Spark의 기능을 활용하고 워크로드를 여러 워커에 분산합니다. Python 스크립트가 Pandas 기반인 경우 Spark의 새 Pandas API를 사용하여 쉽게 마이그레이션할 수 있습니다. 이에 대한 자세한 내용은 AWS 빅 데이터 블로그: Apache Spark용 AWS Glue 4.0 자세히 알아보기에서 확인하세요.

워크로드를 AWS Lambda로 마이그레이션

AWS Lambda는 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 서버리스 컴퓨팅 서비스입니다. AWS Lambda는 시작 시간이 짧고 컴퓨팅 용량 옵션이 유연하기 때문에 이러한 이점을 활용할 수 있습니다. 추가 Python 라이브러리를 관리하기 위해 AWS Glue Python 쉘 작업은 PIP 기반 설치를 사용합니다. 그러나 AWS Lambda의 경우 zip 아카이브, 컨테이너 이미지 또는 Lambda 계층 옵션 중 하나를 선택해야 합니다.

반면 AWS Lambda의 최대 제한 시간은 900초(15분)입니다. 기존 AWS Glue Python 쉘 작업 워크로드의 작업 기간이 그보다 크거나 워크로드에 작업 기간이 길어질 수 있는 급증 패턴이 있는 경우 AWS Lambda 대신 다른 옵션을 탐색하는 것이 좋습니다.

워크로드를 Amazon ECS/Fargate로 마이그레이션

Amazon Elastic Container Service(Amazon ECS)는 컨테이너화된 애플리케이션의 배포, 관리 및 규모 조정을 간소화하는 완전관리형 서비스입니다. AWS Fargate는 Amazon ECS 및 Amazon Elastic Kubernetes Service(Amazon EKS)에서 실행되는 컨테이너화된 워크로드를 위한 서버리스 컴퓨팅 엔진입니다. Amazon ECS 및 Fargate에는 최대 제한 시간이 없으므로 장기 실행 작업에 적합한 옵션입니다. 컨테이너 이미지를 완전히 제어할 수 있으므로 기존 Python 스크립트와 추가 Python 라이브러리를 컨테이너로 가져와 사용할 수 있습니다. 그러나 이 접근 방식을 사용하려면 Python 스크립트를 컨테이너화해야 합니다.

워크로드를 Amazon Managed Workflows for Apache Airflow Python 연산자로 마이그레이션

Amazon Managed Workflows for Apache Airflow(Apache Airflow용 관리형 워크플로)는 Apache Airflow에 대한 관리형 오케스트레이션 서비스로 클라우드에서 종단 간 데이터 파이프라인을 대규모로 쉽게 설정하고 운영할 수 있습니다. 이미 MWAA 환경이 있는 경우 AWS Glue Python 쉘 작업 대신 Python 연산자를 사용하는 것이 간단합니다. Python 연산자는 Airflow 워크플로 내에서 Python 코드를 실행하는 연산자입니다. 그러나 기존 MWAA 환경이 없는 경우 다른 옵션을 탐색하는 것이 좋습니다.

워크로드를 Amazon SageMaker AI AI 훈련 작업으로 마이그레이션

Amazon SageMaker AI Training은 Amazon SageMaker AI에서 제공하는 완전관리형 기계 학습(ML) 서비스로, 광범위한 ML 모델을 대규모로 효율적으로 훈련하는 데 도움이 됩니다. Amazon SageMaker AI AI 작업의 핵심은 ML 워크로드의 컨테이너화와 AWS 컴퓨팅 리소스 관리 기능입니다. 최대 제한 시간이 없는 서버리스 환경을 선호하는 경우 Amazon SageMaker AI AI 훈련 작업이 적합할 수 있습니다. 그러나 시작 지연 시간은 AWS Glue Python 쉘 작업보다 긴 경향이 있습니다. 지연 시간에 민감한 작업의 경우 다른 옵션을 탐색하는 것이 좋습니다.

AWS Glue 모니터링

모니터링은 AWS Glue 및 사용자의 다른 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어서 중요한 부분입니다. AWS은 AWS Glue를 보면서, 이상이 있을 때 이를 보고하고, 적절할 때 자동으로 조치를 취할 수 있게 해주는 모니터링 도구를 제공합니다.

다음과 같은 자동 모니터링 도구를 사용하여 AWS Glue를 관찰하고 문제 발생 시 보고할 수 있습니다.

- Amazon CloudWatch Events는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 제공합니다. CloudWatch Events는 자동화된 이벤트 중심 컴퓨팅을 지원합니다. 특정 이벤트를 감시하는 규칙을 작성하고 이러한 이벤트가 발생할 때 다른 AWS 서비스에서 자동화된 작업을 트리거할 수 있습니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, AWS CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.
- AWS CloudTrail은 직접 수행하거나 AWS 계정을 대신하여 수행한 API 직접 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 호출하는지, 어떤 소스 IP 주소에 호출이 이루어지는지, 언제 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

또한 AWS Glue 콘솔에서 다음과 같은 인사이트에 액세스하여 작업을 디버깅하고 프로파일링하는 데 도움을 받을 수 있습니다.

- Spark 작업 - 선택된 CloudWatch 지표 시리즈의 시각화를 볼 수 있으며, 새로운 작업은 Spark UI에 액세스할 수 있습니다. 자세한 내용은 the section called “Spark 작업 모니터링” 단원을 참조하십시오.
- Ray 작업 - 선택된 CloudWatch 지표 시리즈의 시각화를 볼 수 있습니다. 자세한 내용은 the section called “Ray 작업 지표” 단원을 참조하십시오.

주제

- AWS Glue의 AWS 태그
- EventBridge를 사용한 AWS Glue 자동화
- AWS Glue 리소스 모니터링
- AWS CloudTrail을 사용하여 AWS Glue API 호출 로깅

AWS Glue의 AWS 태그

AWS Glue 리소스를 쉽게 관리할 수 있도록 선택에 따라 일부 AWS Glue 리소스 유형에 자체 태그를 할당할 수 있습니다. 태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. AWS Glue의 태그를 사용하여 리소스를 정리하고 식별할 수 있습니다. 태그는 비용 회계 보고서를 생성하고 리소스에 대한 액세스를 제한하는 데 사용할 수 있습니다. AWS Identity and Access Management를 사용하는 경우 AWS 계정에서 태그를 생성, 편집 또는 삭제할 수 있는 권한이 있는 사용자를 제어할 수 있습니다. 태그 관련 API를 직접 호출하는 권한 외에도 연결에서 태그 지정 API를 직접 호출할 수 있는 `glue:GetConnection` 권한과 데이터베이스에서 태그 지정 API를 직접 호출할 수 있는 `glue:GetDatabase` 권한도 필요합니다. 자세한 내용은 ABAC AWS Glue 사용 단원을 참조하십시오.

AWS Glue에서 다음 리소스에 태그를 지정할 수 있습니다.

- 연결
- 데이터베이스
- 크롤러
- 대화형 세션
- 개발 엔드포인트
- 작업
- 트리거
- 워크플로
- 청사진
- 기계 학습 변환
- 데이터 품질 규칙 세트
- 스트림 스키마
- 스트림 스키마 레지스트리

> **Note**
>
> 이러한 AWS Glue 리소스의 태그 지정이 가능하도록 모범 사례로서 정책에 항상 `glue:TagResource` 작업을 포함시킵니다.

AWS Glue에서 태그를 사용할 때 다음을 고려하십시오.

- 엔터티당 최대 50개의 태그가 지원됩니다.
- AWS Glue에서 `{"string": "string" ...}` 포맷의 키-값 페어 목록으로 태그를 지정합니다.
- 객체에서 태그를 생성할 때 태그 키는 필수이고 태그 값은 선택 사항입니다.
- 태그 키와 태그 값은 대/소문자를 구분합니다.
- 태그 키와 태그 값은 접두사 aws를 포함하지 않아야 합니다. 이러한 태그에서 어떤 작업도 허용되지 않습니다.
- 태그 키의 최대 길이는 UTF-8 형식의 유니코드 문자 128자입니다. 태그 키는 비어 있거나 null이면 안 됩니다.
- 태그 값의 최대 길이는 UTF-8 형식의 유니코드 문자 256자입니다. 태그 값은 비어 있거나 null일 수 있습니다.

AWS Glue 연결에 대한 태깅 지원

리소스 태그를 기준으로 `CreateConnection`, `UpdateConnection`, `GetConnection` 및 `DeleteConnection` 작업 권한을 제한할 수 있습니다. 이렇게 하면 Data Catalog에서 JDBC 연결 정보를 가져와야 하는 JDBC 데이터 원본을 사용하는 AWS Glue 작업에 최소 권한 액세스 제어를 구현할 수 있습니다.

사용 예

["connection-category", "dev-test"] 태그를 사용하여 AWS Glue 연결을 생성합니다.

IAM 정책에서 `GetConnection` 작업에 대한 태그 조건을 지정합니다.

```
{
   "Effect": "Allow",
   "Action": [
     "glue:GetConnection"
     ],
   "Resource": "*",
   "Condition": {
     "ForAnyValue:StringEquals": {
       "aws:ResourceTag/tagKey": "dev-test"
     }
   }
 }
```

예시

다음 예는 태그가 할당된 작업을 생성합니다.

AWS CLI

```
aws glue create-job --name job-test-tags --role MyJobRole --command
 Name=glueetl,ScriptLocation=S3://aws-glue-scripts//prod-job1
--tags key1=value1,key2=value2
```

AWS CloudFormation JSON

```
{
  "Description": "AWS Glue Job Test Tags",
  "Resources": {
    "MyJobRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "glue.amazonaws.com"
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
        "Policies": [
          {
            "PolicyName": "root",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
```

```
                  "Action": "*",
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    },
    "MyJob": {
      "Type": "AWS::Glue::Job",
      "Properties": {
        "Command": {
          "Name": "glueetl",
          "ScriptLocation": "s3://aws-glue-scripts//prod-job1"
        },
        "DefaultArguments": {
          "--job-bookmark-option": "job-bookmark-enable"
        },
        "ExecutionProperty": {
          "MaxConcurrentRuns": 2
        },
        "MaxRetries": 0,
        "Name": "cf-job1",
        "Role": {
           "Ref": "MyJobRole",
           "Tags": {
                "key1": "value1",
                "key2": "value2"
           }
        }
      }
    }
  }
}
```

AWS CloudFormation YAML

```
Description: AWS Glue Job Test Tags
Resources:
  MyJobRole:
    Type: AWS::IAM::Role
    Properties:
```

```
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - glue.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: "/"
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action: "*"
                Resource: "*"
  MyJob:
    Type: AWS::Glue::Job
    Properties:
      Command:
        Name: glueetl
        ScriptLocation: s3://aws-glue-scripts//prod-job1
      DefaultArguments:
        "--job-bookmark-option": job-bookmark-enable
      ExecutionProperty:
        MaxConcurrentRuns: 2
      MaxRetries: 0
      Name: cf-job1
      Role:
        Ref: MyJobRole
        Tags:
           key1: value1
           key2: value2
```

자세한 내용은 AWS 태그 지정 전략을 참조하세요.

태그를 이용한 액세스 제어 방법에 대한 자세한 내용은 ABAC AWS Glue 사용 단원을 참조하십시오.

EventBridge를 사용한 AWS Glue 자동화

Amazon EventBridge를 사용하면 AWS 서비스를 자동화하고 애플리케이션 가용성 문제나 리소스 변경 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. AWS 서비스의 이벤트는 거의 실시간으로 EventBridge로 전송됩니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다. 자동으로 트리거할 수 있는 작업은 다음과 같습니다.

- AWS Lambda 함수 호출
- Amazon EC2 Run Command 호출
- Amazon Kinesis Data Streams로 이벤트 릴레이
- AWS Step Functions 상태 머신 활성화
- Amazon SNS 주제 또는 Amazon SQS 대기열 알림

다음은 EventBridge를 AWS Glue에 사용하는 몇 가지 예시입니다.

- ETL 작업이 성공하면 Lambda 함수를 활성화합니다.
- ETL 작업이 실패하면 Amazon SNS 주제를 알립니다.

다음 EventBridge는 AWS Glue에 의해 생성됩니다.

- `"detail-type":"Glue Job State Change"` 이벤트는 `SUCCEEDED`, `FAILED`, `TIMEOUT` 및 `STOPPED`에 대해 생성됩니다.
- 작업 지연 알림 임곗값을 초과한다면 `RUNNING`, `STARTING` 및 `STOPPING` 작업 실행에 `"detail-type":"Glue Job Run Status"`용 이벤트가 생성됩니다. 이러한 이벤트를 수신하려면 작업 지연 알림 임계값 속성을 설정해야 합니다.

  작업 지연 알림 임계값을 초과한 경우 작업 실행 상태당 하나의 이벤트만 생성됩니다.
- `"detail-type":"Glue Crawler State Change"` 이벤트는 `Started`, `Succeeded` 및 `Failed`을 위해 발생합니다.
- `“detail_type”:“Glue Scheduled Crawler Invocation Failure”`에 대한 이벤트는 예약된 크롤러가 시작되지 않을 때 생성됩니다. 알림 세부 정보:
  - `customerId`에는 고객의 계정 ID가 포함됩니다.
  - `crawlerName`에는 시작하지 못한 크롤러의 이름이 포함됩니다.
  - `errorMessage`에는 간접 호출 실패의 예외 메시지가 포함됩니다.

- “detail_type”:“Glue Auto Statistics Invocation Failure“에 대한 이벤트는 자동 관리형 열 통계 작업 실행이 시작되지 않을 때 생성됩니다. 알림 세부 정보:
  - catalogId에는 카탈로그와 연결된 ID가 포함됩니다.
  - databaseName에는 영향을 받는 데이터베이스 이름이 포함됩니다.
  - tableName에는 영향을 받는 테이블 이름이 포함됩니다.
  - errorMessage에는 간접 호출 실패의 예외 메시지가 포함됩니다.
- “detail_type”:“Glue Scheduled Statistics Invocation Failure”에 대한 이벤트는 (cron) 예약 열 통계 작업 실행이 시작되지 않을 때 생성됩니다. 알림 세부 정보:
  - catalogId에는 카탈로그와 연결된 ID가 포함됩니다.
  - databaseName에는 영향을 받는 데이터베이스 이름이 포함됩니다.
  - tableName에는 영향을 받는 테이블 이름이 포함됩니다.
  - errorMessage에는 간접 호출 실패의 예외 메시지가 포함됩니다.
- “detail_type”:“Glue Statistics Task Started”에 대한 이벤트는 열 통계 작업 실행이 시작될 때 생성됩니다.
- “detail_type”:“Glue Statistics Task Succeeded”에 대한 이벤트는 열 통계 작업 실행이 성공할 때 생성됩니다.
- “detail_type”:“Glue Statistics Task Failed”에 대한 이벤트는 열 통계 작업 실행이 실패할 때 생성됩니다.
- "detail-type":"Glue Data Catalog Database State Change" 이벤트는 CreateDatabase, DeleteDatabase, CreateTable, DeleteTable 및 BatchDeleteTable에 대해 생성됩니다. 예를 들어 테이블이 생성되거나 삭제되면 EventBridge에 알림이 전송됩니다. 순서가 잘못되었거나 누락되는 등 알림 이벤트의 순서 또는 존재 여부에 따라 프로그램을 작성할 수 없을 수 있습니다. 이벤트는 최선의 작업을 기반으로 발생됩니다. 알림 세부 정보:
  - typeOfChange에는 API 작업의 이름이 포함되어 있습니다.
  - databaseName에는 영향을 받는 데이터베이스 이름이 포함되어 있습니다.
  - changedTables는 알림당 영향을 받는 테이블의 이름을 최대 100개까지 포함합니다. 테이블 이름이 길면 여러 알림이 생성될 수 있습니다.
- "detail-type":"Glue Data Catalog Table State Change"에 대한 이벤트는 UpdateTable, CreatePartition, BatchCreatePartition, UpdatePartition, DeletePartition, BatchUpdatePartition 및 BatchDeletePartition에 대해 생성됩니다. 예를 들어 테이블이 생성되거나 파티션이 업데이트되면 EventBridge에 알림이 전송됩니다. 순서가 잘못되었거나 누락되는 등 알림 이벤트의 순서 또는 존재 여부에 따라 프로그램을 작성할 수 없을 수 있습니다. 이벤트는 최선의 작업을 기반으로 발생됩니다. 알림 세부 정보:

- typeOfChange에는 API 작업의 이름이 포함되어 있습니다.
- databaseName은 영향을 받는 리소스가 포함된 데이터베이스의 이름을 포함합니다.
- tableName에는 영향을 받는 테이블 이름이 포함되어 있습니다.
- changedPartitions는 하나의 알림에서 최대 100개의 영향을 받는 파티션을 지정합니다. 파티션 이름이 길면 여러 알림이 생성될 수 있습니다.

  예를 들어 Year 및 Month와 같은 두 개의 파티션 키가 있는 경우"2018,01", "2018,02"는 "Year=2018" and "Month=01" 및 "Year=2018" and "Month=02"가 있는 파티션을 수정합니다.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Glue Data Catalog Table State Change",
    "source":"aws.glue",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":["arn:aws:glue:us-west-2:123456789012:database/default/foo"],
    "detail":{
        "changedPartitions": [
            "2018,01",
            "2018,02"
        ],
        "databaseName": "default",
        "tableName": "foo",
        "typeOfChange": "BatchCreatePartition"
        }
}
```

자세한 내용은 Amazon CloudWatch Events 사용 설명서를 참조하세요. AWS Glue에 한정되는 이벤트는 AWS Glue 이벤트를 참조하십시오.

AWS Glue 리소스 모니터링

AWS Glue에는 예상치 못한 과도한 프로비저닝과 요금 증가를 의도한 악의적인 행동으로부터 고객을 보호하기 위한 서비스 한도가 있습니다. 또한 이러한 한도는 서비스도 보호합니다. AWS Service Quota 콘솔에 로그인하면 고객은 현재 리소스 한도를 확인하고 필요한 경우 리소스 증가를 요청할 수 있습니다.

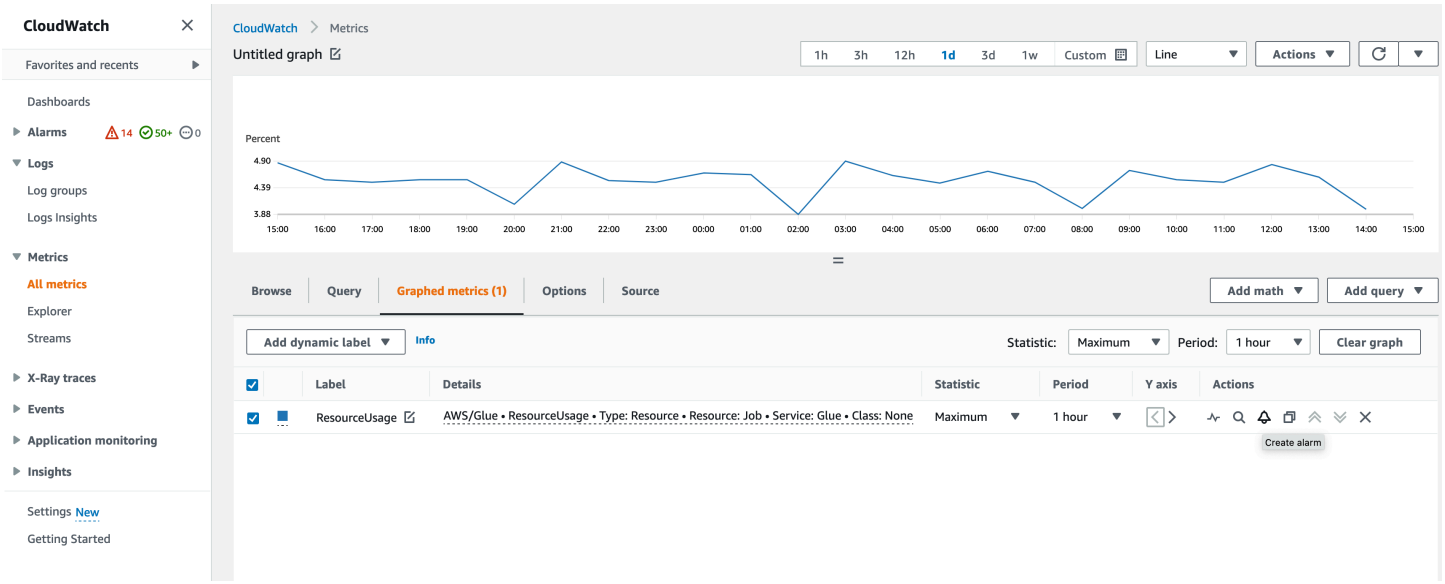
AWS Glue에서는 Amazon CloudWatch에서 서비스의 리소스 사용량을 백분율로 보고 사용량을 모니터링하도록 CloudWatch 경보를 구성할 수 있습니다. Amazon CloudWatch는 Amazon 인프라에서 실행되는 AWS 리소스 및 고객 애플리케이션에 대한 모니터링을 제공합니다. 지표는 무료로 제공됩니다. 다음과 같은 지표가 지원됩니다.

- 계정당 워크플로우 수
- 계정당 트리거 수
- 계정당 작업 수
- 계정당 동시 작업 실행 개수
- 계정당 청사진 수
- 계정당 대화형 세션 수

리소스 지표 구성 및 사용

이 기능을 사용하려면 Amazon CloudWatch 콘솔로 이동하여 지표를 보고 경보를 구성하면 됩니다. 지표는 AWS/Glue 네임스페이스 아래에 있으며 실제 리소스 사용량 수치를 리소스 할당량으로 나눈 백분율입니다. CloudWatch 지표는 사용자 계정으로 전송되므로 비용이 청구되지 않습니다. 예를 들어, 10개의 워크플로를 생성하고 서비스 할당량으로 최대 200개의 워크플로를 사용할 수 있는 경우 사용량은 10/200 = 5%이고, 그래프에는 5의 데이터 포인트가 백분율로 표시됩니다. 구체적으로 설명하면 다음과 같습니다.

```
Namespace: AWS/Glue
Metric name: ResourceUsage
Type: Resource
Resource: Workflow (or Trigger, Job, JobRun, Blueprint, InteractiveSession)
Service: Glue
Class: None
```



CloudWatch 콘솔에서 지표에 대한 경보를 생성하려면:

1. 지표를 찾은 후에 그래프로 표시된 지표로 이동합니다.
2. 작업에서 경보 생성을 클릭합니다.
3. 필요에 따라 경보를 구성합니다.

리소스 사용량이 변경될 때마다(예: 증가 또는 감소) 지표를 생성합니다. 그러나 리소스 사용량이 변하지 않는 경우에는 매시간 지표를 생성하므로 지속적인 CloudWatch 그래프가 생성됩니다. 데이터 포인트가 누락되는 것을 방지하기 위해 1시간 미만의 기간을 구성하는 것은 권장하지 않습니다.

또한 다음 예제에서처럼 AWS CloudFormation을 사용하여 경보를 구성할 수 있습니다. 이 예제에서 워크플로 리소스 사용량이 80%에 도달하면 기존 SNS 주제에 메시지를 보내는 경보가 트리거되며, 이 주제를 구독하면 알림을 받을 수 있습니다.

```
{
 "Type": "AWS::CloudWatch::Alarm",
 "Properties": {
  "AlarmName": "WorkflowUsageAlarm",
  "ActionsEnabled": true,
  "OKActions": [],
  "AlarmActions": [
   "arn:aws:sns:af-south-1:085425700061:Default_CloudWatch_Alarms_Topic"
  ],
  "InsufficientDataActions": [],
  "MetricName": "ResourceUsage",
```

```
  "Namespace": "AWS/Glue",
  "Statistic": "Maximum",
  "Dimensions": [{
    "Name": "Type",
    "Value": "Resource"
   },
   {
    "Name": "Resource",
    "Value": "Workflow"
   },
   {
    "Name": "Service",
    "Value": "Glue"
   },
   {
    "Name": "Class",
    "Value": "None"
   }
  ],
  "Period": 3600,
  "EvaluationPeriods": 1,
  "DatapointsToAlarm": 1,
  "Threshold": 80,
  "ComparisonOperator": "GreaterThanThreshold",
  "TreatMissingData": "notBreaching"
 }
}
```

AWS CloudTrail을 사용하여 AWS Glue API 호출 로깅

AWS Glue는 AWS Glue에서 사용자, 역할, 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail와 통합됩니다. CloudTrail은 AWS Glue에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 AWS Glue 콘솔로부터의 직접 호출과 AWS Glue API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 AWS Glue 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 AWS Glue에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

CloudTrail의 AWS Glue 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. AWS Glue에서 활동이 발생하면 해당 활동이 [이벤트 기록]의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 설명은 CloudTrail 이벤트 이력을 사용하여 이벤트 보기를 참조하십시오.

AWS Glue에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 지역에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 지역의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 추가적으로, CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오.

- AWS 계정에 대한 추적 생성
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 AWS Glue 작업이 CloudTrail에서 로깅되고 AWS Glue API에서 문서화됩니다. 예를 들어 `CreateDatabase`, `CreateTable`, `CreateScript` 작업을 직접 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지.

자세한 설명은 CloudTrail userIdentity 요소를 참조하십시오.

그러나 CloudTrail은 호출에 따른 모든 정보를 로그하지 않습니다. 예를 들어, 연결에 사용된 `ConnectionProperties`와 같은 특정 민감한 정보는 로그하지 않고 다음 API에 의해 반환된 응답 대신 `null`을 로그합니다.

```
BatchGetPartition       GetCrawlers          GetJobs          GetTable
CreateScript            GetCrawlerMetrics    GetJobRun        GetTables
```

```
GetCatalogImportStatus  GetDatabase          GetJobRuns       GetTableVersions
GetClassifier           GetDatabases         GetMapping       GetTrigger
GetClassifiers          GetDataflowGraph     GetObjects       GetTriggers
GetConnection           GetDevEndpoint       GetPartition     GetUserDefinedFunction
GetConnections          GetDevEndpoints      GetPartitions    GetUserDefinedFunctions
GetCrawler              GetJob               GetPlan
```

AWS Glue 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `DeleteCrawler` 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/johndoe",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "johndoe"
  },
  "eventTime": "2017-10-11T22:29:49Z",
  "eventSource": "glue.amazonaws.com",
  "eventName": "DeleteCrawler",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "72.21.198.64",
  "userAgent": "aws-cli/1.11.148 Python/3.6.1 Darwin/16.7.0 botocore/1.7.6",
  "requestParameters": {
    "name": "tes-alpha"
  },
  "responseElements": null,
  "requestID": "b16f4050-aed3-11e7-b0b3-75564a46954f",
  "eventID": "e73dd117-cfd1-47d1-9e2f-d1271cad838c",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

다음은 CreateConnection 작업을 설명하는 CloudTrail 로그 항목을 보여주는 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/johndoe",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "johndoe"
  },
  "eventTime": "2017-10-13T00:19:19Z",
  "eventSource": "glue.amazonaws.com",
  "eventName": "CreateConnection",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "72.21.198.66",
  "userAgent": "aws-cli/1.11.148 Python/3.6.1 Darwin/16.7.0 botocore/1.7.6",
  "requestParameters": {
    "connectionInput": {
      "name": "test-connection-alpha",
      "connectionType": "JDBC",
      "physicalConnectionRequirements": {
        "subnetId": "subnet-323232",
        "availabilityZone": "us-east-1a",
        "securityGroupIdList": [
          "sg-12121212"
        ]
      }
    }
  },
  "responseElements": null,
  "requestID": "27136ebc-afac-11e7-a7d6-ab217e5c3f19",
  "eventID": "e8b3baeb-c511-4597-880f-c16210c60a4a",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

AWS Glue 작업 실행 상태

AWS Glue 추출, 변환, 로드 작업이 실행되는 동안 또는 중지된 후 작업 상태를 볼 수 있습니다. AWS Glue 콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS Glue API의 GetJobRun 작업을 사용하여 상태를 볼 수 있습니다.

가능한 작업 실행 상태는 `STARTING`, `RUNNING`, `STOPPING`, `STOPPED`, `SUCCEEDED`, `FAILED`, `ERROR`, `WAITING` 및 `TIMEOUT`입니다.

다음 표에는 비정상적인 작업 종료를 나타내는 상태가 나열되어 있습니다.

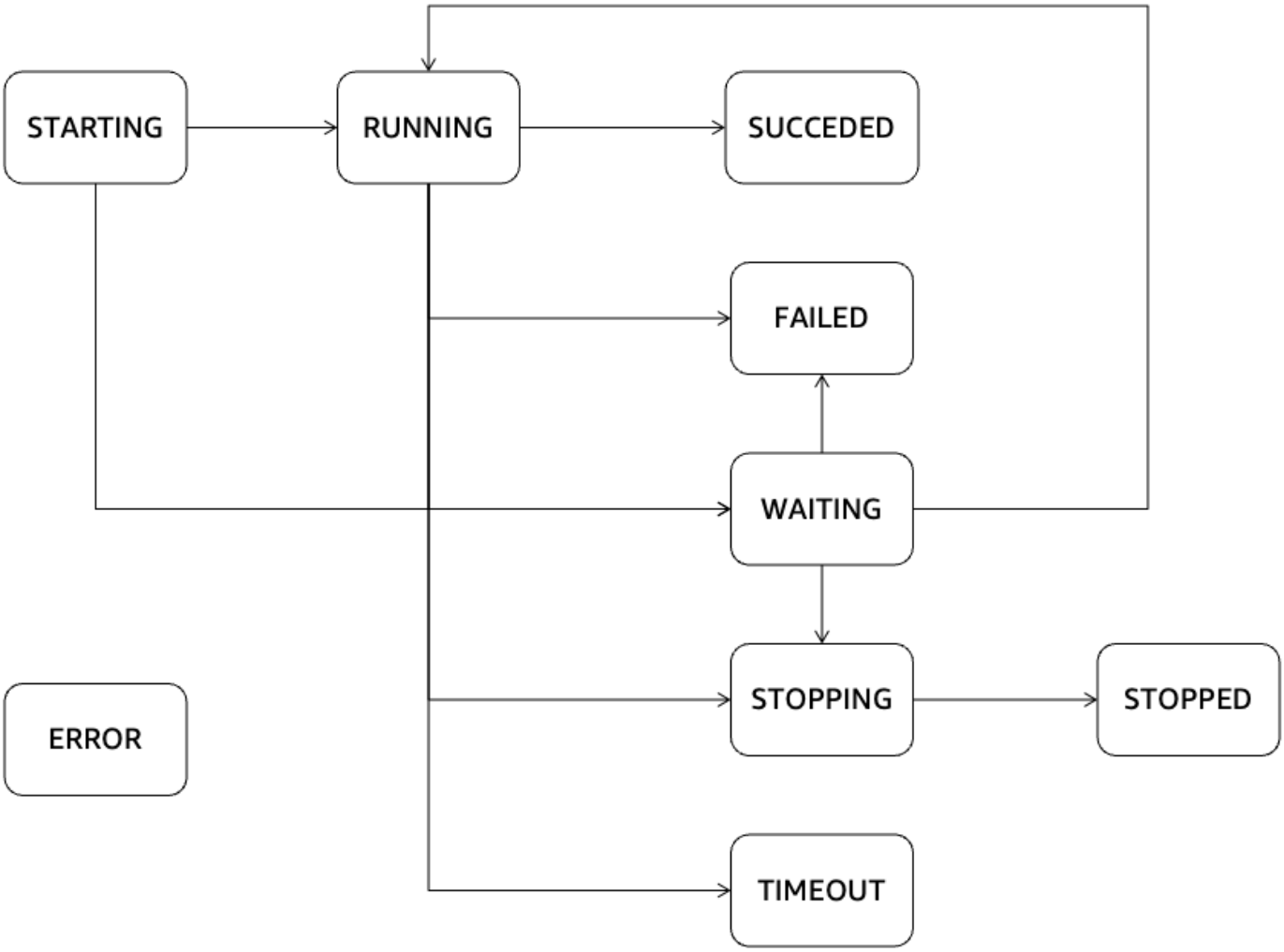
작업 실행 상태	설명
`FAILED`	작업이 허용되는 최대 동시 실행을 초과했거나 알 수 없는 종료 코드로 종료되었습니다.
`ERROR`	워크플로, 일정 트리거 또는 이벤트 트리거가 삭제된 작업을 실행하려고 했습니다.
`TIMEOUT`	작업 런타임이 지정된 시간 제한 값을 초과했습니다.
`WAITING`	리소스를 기다리는 작업 실행 상태입니다.

`WAITING` 상태는 작업 실행에서 리소스를 기다리고 있음을 나타냅니다. 다음 표에서는 여러 작업 클래스의 대기 동작을 설명합니다.

작업 유형	동작
Spark 작업(표준)	AWS Glue에서 Spark 작업에 대한 작업 속성 구성에 설명된 대로 AWS Glue 작업 실행 대기열을 선택하면 작업이 WAITING 상태가 될 수 있습니다. 작업 실행은 계정의 서비스 할당량 또는 해당 지역의 용량 제한으로 인해 다음과 같은 오류 사례 중 하나가 발생한 경우 WAITING 상태일 수 있습니다.

작업 유형	동작
	• 계정당 최대 동시 작업 실행 수를 초과했습니다 • 작업당 최대 동시 작업 실행 수를 초과했습니다(계정 수준 서비스 할당량 및 작업에 `MaxConcurrentRuns` 로 지정한 한도 포함) • 최대 동시 컴퓨팅(DPU 사용량) 초과 • 사용 가능하지 않는 리소스 서비스 할당량에 관계없이 작업 실행을 시작할 리소스가 충분하지 않은 경우에도 작업이 WAITING 상태로 전환될 수 있습니다.
Spark 잡(플렉스)	서비스에서 실행을 시작하기에 충분한 리소스를 얻을 수 없는 경우 새 작업 실행은 WAITING 상태가 되며, 이로 인해 실행 시작이 지연됩니다. 실행은 최대 20분 동안 WAITING 상태가 됩니다(서비스에 의해 시간 초과가 제어됨). 15분 후 서비스가 강제 시작을 시도하며 사용 가능한 용량에 따라 적절한 오류 메시지와 함께 실행이 시작되거나 실패할 수 있습니다.
Python 셸 작업	Spark를 사용하는 표준 작업과 동일한 동작입니다.

다음 상태 다이어그램은 AWS Glue 작업의 수명 주기 동안 예상되는 상태 전환을 간략하게 설명합니다. 이 정보는 모든 작업 유형에 적용됩니다.


STARTING
RUNNING
SUCCEDED
FAILED
WAITING
STOPPING
STOPPED
ERROR
TIMEOUT

AWS Glue 스트리밍

AWS Glue의 구성 요소인 AWS Glue 스트리밍을 통해 거의 실시간으로 스트리밍 데이터를 효율적으로 처리하여 데이터 모으기, 처리, 기계 학습과 같은 중요한 작업을 수행할 수 있습니다. AWS Glue Streaming은 Apache Spark Streaming 프레임워크를 사용하여 스트리밍 데이터를 대규모로 처리할 수 있는 서버리스 서비스를 제공합니다. AWS Glue는 Apache Spark를 기반으로 서버리스 인프라, 자동 크기 조정, 시각적 작업 개발, 스트리밍 작업을 위한 인스턴트 온 노트북 및 기타 성능 개선과 같은 다양한 최적화를 제공합니다.

스트리밍 사용 사례

다음은 AWS Glue 스트리밍의 일반적인 사용 사례입니다.

실시간에 가까운 데이터 처리: AWS Glue 스트리밍을 통해 조직은 스트리밍 데이터를 거의 실시간으로 처리하여 최신 정보를 기반으로 인사이트를 얻고 시기적절한 결정을 내릴 수 있습니다.

사기 탐지: AWS Glue 스트리밍 데이터의 실시간 분석을 위해 스트리밍을 활용하면 신용 카드 사기, 네트워크 침입 또는 온라인 사기와 같은 사기 행위를 탐지하는 데 유용합니다. 수신 데이터를 지속적으로 처리하고 분석하여 의심스러운 패턴이나 이상 징후를 신속하게 식별할 수 있습니다.

소셜 미디어 분석: AWS Glue 스트리밍을 통해 트윗, 게시물, 댓글 등의 실시간 소셜 미디어 데이터를 처리할 수 있으므로 조직은 추세를 모니터링하고, 감정 분석을 하고, 브랜드 평판을 실시간으로 관리할 수 있습니다.

사물 인터넷(IoT) 분석: AWS Glue 스트리밍은 IoT 디바이스, 센서 및 연결된 기계에서 생성되는 고속 데이터 스트림을 처리하고 분석하는 데 적합합니다. 이는 실시간 모니터링, 이상 탐지, 예측 유지 보수 및 기타 IoT 분석 사용 사례를 지원합니다.

클릭스트림 분석: AWS Glue 스트리밍은 웹 사이트 또는 모바일 애플리케이션의 실시간 클릭스트림 데이터를 처리하고 분석할 수 있습니다. 이를 통해 기업은 사용자 행동에 대한 인사이트를 얻고, 사용자 경험을 개인화하고, 실시간 클릭스트림 데이터를 기반으로 마케팅 캠페인을 최적화할 수 있습니다.

로그 모니터링 및 분석: AWS Glue 스트리밍은 서버, 애플리케이션 또는 네트워크 디바이스의 로그 데이터를 실시간으로 지속적으로 처리하고 분석할 수 있습니다. 이는 이상을 탐지하고, 문제를 해결하고, 시스템 상태와 성능을 모니터링하는 데 도움이 됩니다.

추천 시스템: AWS Glue 스트리밍은 사용자 활동 데이터를 실시간으로 처리하고 추천 모델을 동적으로 업데이트할 수 있습니다. 이를 통해 사용자 행동 및 기본 설정을 기반으로 개인화된 실시간 추천이 가능합니다.

다음은 AWS Glue 스트리밍을 적용할 수 있는 다양한 사용 사례의 몇 가지 예제입니다. AWS 에코시스템 및 관리형 서비스와의 통합으로 클라우드에서 실시간 스트림 처리 및 분석을 위한 편리한 선택이 됩니다.

AWS Glue 스트리밍 사용의 이점은 무엇인가요?

다음은 AWS Glue 스트리밍 사용의 이점입니다.

- 서버리스: AWS Glue 스트리밍은 서버리스이므로 인프라를 관리할 필요가 없습니다. 이를 통해 운영 오버헤드가 줄어들고 사용자는 인프라 관리보다는 데이터 처리 및 분석 작업에 집중할 수 있습니다.
- 자동 크기 조정: AWS Glue 스트리밍은 워크로드에 따라 처리 용량을 동적으로 조정하는 자동 크기 조정 기능을 제공합니다. 데이터 볼륨의 변동을 처리하기 위해 자동으로 스케일 아웃 또는 스케일 인하여 최적의 성능과 리소스 활용도를 보장합니다.
- 시각적 개발: 스트리밍 작업 개발은 복잡할 수 있습니다. AWS Glue 스트리밍은 시각적 저작 도구인 AWS Glue Studio를 제공하여 이러한 문제를 해결합니다. AWS Glue Studio는 스트리밍 워크플로 생성 프로세스를 간소화하고 개발자가 스트리밍 애플리케이션을 시각적으로 설계 및 관리할 수 있도록 하여 학습 곡선을 줄이고 생산성을 높입니다.
- 비용 효율성: AWS Glue 스트리밍은 서버리스 서비스로서 인프라를 프로비저닝하고 유지 보수할 필요가 없으므로 비용 효율성을 제공합니다. 스트리밍 작업을 실행하는 동안 소비된 리소스를 기준으로 사용자에게 요금이 청구되므로 실제 사용량에 따라 비용을 최적화하고 조정할 수 있습니다.
- 복잡한 워크로드 처리: AWS Glue 스트리밍은 복잡한 스트리밍 워크로드를 처리하도록 설계되었습니다. 대량의 실시간 데이터를 처리 및 분석하고, 고급 변환을 지원하고, 다른 AWS 서비스와 통합하여 정교한 스트리밍 데이터 파이프라인과 분석 워크플로를 지원할 수 있습니다.
- 종속 없음: AWS Glue 스트리밍은 유연성을 제공하고 벤더 종속을 방지합니다. 사용자는 AWS Glue 스트리밍을 보다 광범위한 AWS 에코시스템의 일부로 활용하여 다른 AWS 서비스와 원활하게 통합할 수 있습니다. 이를 통해 특정 기술이나 플랫폼에 얽매이지 않고도 기존 데이터 소스, 애플리케이션 및 서비스와 쉽게 통합할 수 있습니다.

AWS Glue 스트리밍은 언제 사용하나요?

스트리밍 사용 사례에는 여러 가지 옵션이 있습니다. 다음 시나리오에서는 AWS Glue 스트리밍을 권장합니다.

1. 배치 처리에 이미 AWS Glue 또는 Spark를 사용하고 있다면 AWS Glue 스트리밍이 이상적인 선택입니다. 새로운 언어나 프레임워크를 배울 필요 없이 스트리밍 작업 구축으로 원활하게 전환할 수

있습니다. AWS Glue 스트리밍은 기존 지식과 인프라를 활용하여 작업 개발 프로세스를 간소화하고 데이터 처리 기능을 실시간 스트리밍 시나리오로 쉽게 확장할 수 있도록 합니다.

2. 배치, 스트리밍, 이벤트 기반 워크로드를 처리하기 위한 통합 서비스나 제품이 필요하다면 AWS Glue 스트리밍이 적합한 솔루션입니다. AWS Glue 스트리밍을 사용하면 데이터 처리 요구 사항을 단일 프레임워크로 통합하여 여러 시스템을 관리하는 복잡성을 없앨 수 있습니다. 이를 통해 다양한 데이터 워크플로를 효율적으로 개발하고 유지 보수하는 동시에 다양한 워크로드 유형에서 일관성과 호환성을 보장할 수 있습니다.
3. AWS Glue 스트리밍은 매우 큰 스트리밍 데이터 볼륨과 스트림 또는 관계형 데이터베이스 간 조인과 같은 복잡한 변환과 관련된 시나리오에 적합합니다. 대량의 데이터 스트림을 효율적으로 처리하고 분석할 수 있으므로 까다로운 워크로드를 쉽게 처리할 수 있습니다. 고속 데이터 모으기든 복잡한 데이터 조작이든 AWS Glue 스트리밍의 확장성과 고급 처리 기능은 최적의 성능과 정확한 결과를 보장합니다.
4. 스트리밍 작업을 구축하는 데 시각적 접근 방식을 선호하는 경우 AWS Glue는 스트리밍 애플리케이션을 시각적으로 설계하고 관리할 수 있는 AWS Glue Studio를 제공하여 개발 프로세스를 간소화합니다. 이 직관적인 인터페이스를 통해 개발자는 시각적 인터페이스를 사용하여 스트리밍 워크플로를 생성, 구성 및 모니터링할 수 있으므로 학습 곡선을 줄이고 생산성을 높일 수 있습니다.
5. AWS Glue 스트리밍은 10초 이상의 엄격한 서비스 수준 계약(SLA)이 있는 실시간에 가까운 사용 사례에 탁월한 선택입니다.
6. Apache Iceberg, Apache Hudi 또는 Delta Lake를 사용하여 트랜잭션 데이터 레이크를 구축하는 경우 AWS Glue 스트리밍은 이러한 오픈 테이블 형식을 기본적으로 지원합니다. 이러한 원활한 통합을 통해 이러한 트랜잭션 데이터 레이크에서 직접 스트리밍 데이터를 처리하여 데이터 일관성, 무결성 및 호환성을 보장할 수 있습니다.
7. 다양한 데이터 대상에 대한 스트리밍 데이터를 모아야 하는 경우: AWS Glue 스트리밍은 Amazon Redshift, Amazon RDS, Amazon Aurora, Oracle, SQL Server 및 기타 대상과 같은 다양한 데이터 대상에 대한 기본 대상을 제공합니다.

지원되는 데이터 원본

AWS Glue 스트리밍에서 지원되는 데이터 소스는 다음과 같습니다.

- Amazon Kinesis
- Amazon MSK(Managed Streaming for Apache Kafka)
- 자체 관리형 Apache Kafka

지원되는 데이터 대상

AWS Glue 스트리밍은 다음과 같은 다양한 데이터 대상을 지원합니다.

- AWS Glue 데이터 카탈로그에서 지원하는 데이터 대상
- Amazon S3
- Amazon Redshift
- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- Snowflake
- JDBC를 사용하여 연결할 수 있는 모든 데이터베이스
- Apache Iceberg, Delta 및 Apache Hudi
- AWS Glue Marketplace 커넥터

자습서: AWS Glue Studio를 사용하여 첫 번째 스트리밍 워크로드 구축

이 자습서에서는 AWS Glue Studio를 사용하여 스트리밍 작업을 생성하는 방법을 알아봅니다. AWS Glue Studio는 AWS Glue 작업을 생성하는 시각적 인터페이스입니다.

지속적으로 실행되고 Amazon Kinesis Data Streams, Apache Kafka 및 Amazon Managed Streaming for Apache Kafka(Amazon MSK)에서 스트리밍 소스의 데이터를 소비하는 스트리밍 추출, 전환, 적재(ETL) 작업을 생성할 수 있습니다.

사전 조건

이 자습서를 따르려면 AWS Glue, Amazon Kinesis, Amazon S3, Amazon Athena, AWS CloudFormation, AWS Lambda 및 Amazon Cognito를 사용할 수 있는 AWS 콘솔 권한이 있는 사용자가 필요합니다.

Amazon Kinesis의 스트리밍 데이터 소비

주제

- Kinesis 데이터 생성기로 모의 데이터 생성
- AWS Glue Studio로 AWS Glue 스트리밍 작업 생성
- 변환 수행 및 Amazon S3에 변환된 결과 저장

Kinesis 데이터 생성기로 모의 데이터 생성

Kinesis 데이터 생성기(KDG)를 사용하여 JSON 형식의 샘플 데이터를 종합적으로 생성할 수 있습니다. 도구 설명서에서 전체 지침과 세부 정보를 찾아볼 수 있습니다.

1. 시작하려면

   Launch Stack 을 클릭하여 AWS 환경에서 AWS CloudFormation 템플릿을 실행합니다.

   > **Note**
   >
   > Kinesis 데이터 생성기용 Amazon Cognito 사용자와 같은 일부 리소스가 AWS 계정에 이미 존재하기 때문에 CloudFormation 템플릿 오류가 발생할 수 있습니다. 다른 자습서나 블로그에서 이미 설정했기 때문일 수 있습니다. 이 문제를 해결하려면 새 AWS 계정에서 템플릿을 사용해 새로 시작하거나 다른 AWS 리전을 탐색해 봅니다. 이러한 옵션을 사용하면 기존 리소스와 충돌하지 않고 자습서를 실행할 수 있습니다.

   템플릿은 Kinesis 데이터 스트림과 Kinesis Data Generator 계정을 프로비저닝합니다. 또한 데이터를 보관할 Amazon S3 버킷과 이 자습서에 필요한 권한이 있는 Glue 서비스 역할을 생성합니다.
2. KDG가 인증에 사용할 사용자 이름과 암호를 입력합니다. 나중에 사용할 수 있도록 사용자 이름과 암호를 기록해 둡니다.
3. 마지막 단계까지 다음을 선택합니다. IAM 리소스 생성을 확인합니다. 화면 상단에 최소 요구 사항을 충족하지 않는 암호와 같은 오류가 있는지 확인하고 템플릿을 배포합니다.
4. 스택의 출력 탭으로 이동합니다. 템플릿이 배포되면 생성된 속성 KinesisDataGeneratorUrl이 표시됩니다. 해당 URL을 클릭합니다.
5. 기록해 둔 사용자 이름과 암호를 입력합니다.
6. 사용 중인 리전을 선택하고 Kinesis 스트림 `GlueStreamTest-{AWS::AccountId}`를 선택합니다.
7. 다음 템플릿을 입력합니다.

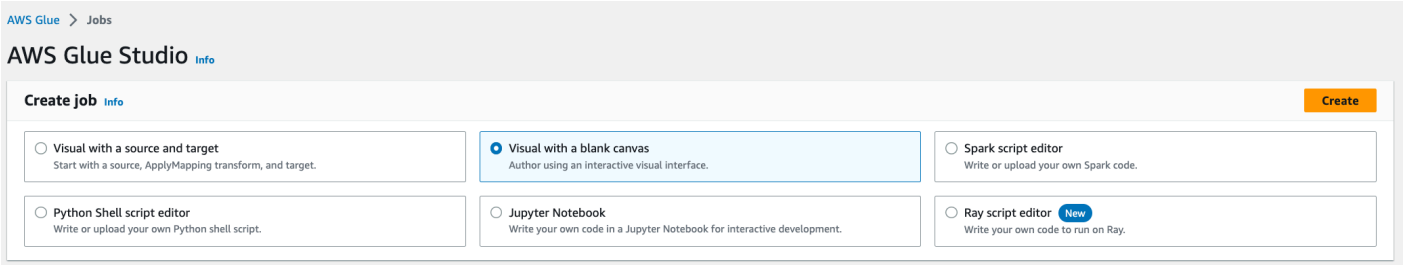
```
{
    "ventilatorid": {{random.number(100)}},
    "eventtime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "serialnumber": "{{random.uuid}}",
    "pressurecontrol": {{random.number(
        {
            "min":5,
            "max":30
        }
    )}},
    "o2stats": {{random.number(
        {
            "min":92,
            "max":98
        }
    )}},
    "minutevolume": {{random.number(
        {
            "min":5,
            "max":8
        }
    )}},
    "manufacturer": "{{random.arrayElement(
        ["3M", "GE","Vyaire", "Getinge"]
    )}}"
}
```

이제 테스트 템플릿을 사용하여 모의 데이터를 보고 데이터 전송을 사용하여 모의 데이터를 Kinesis로 모을 수 있습니다.

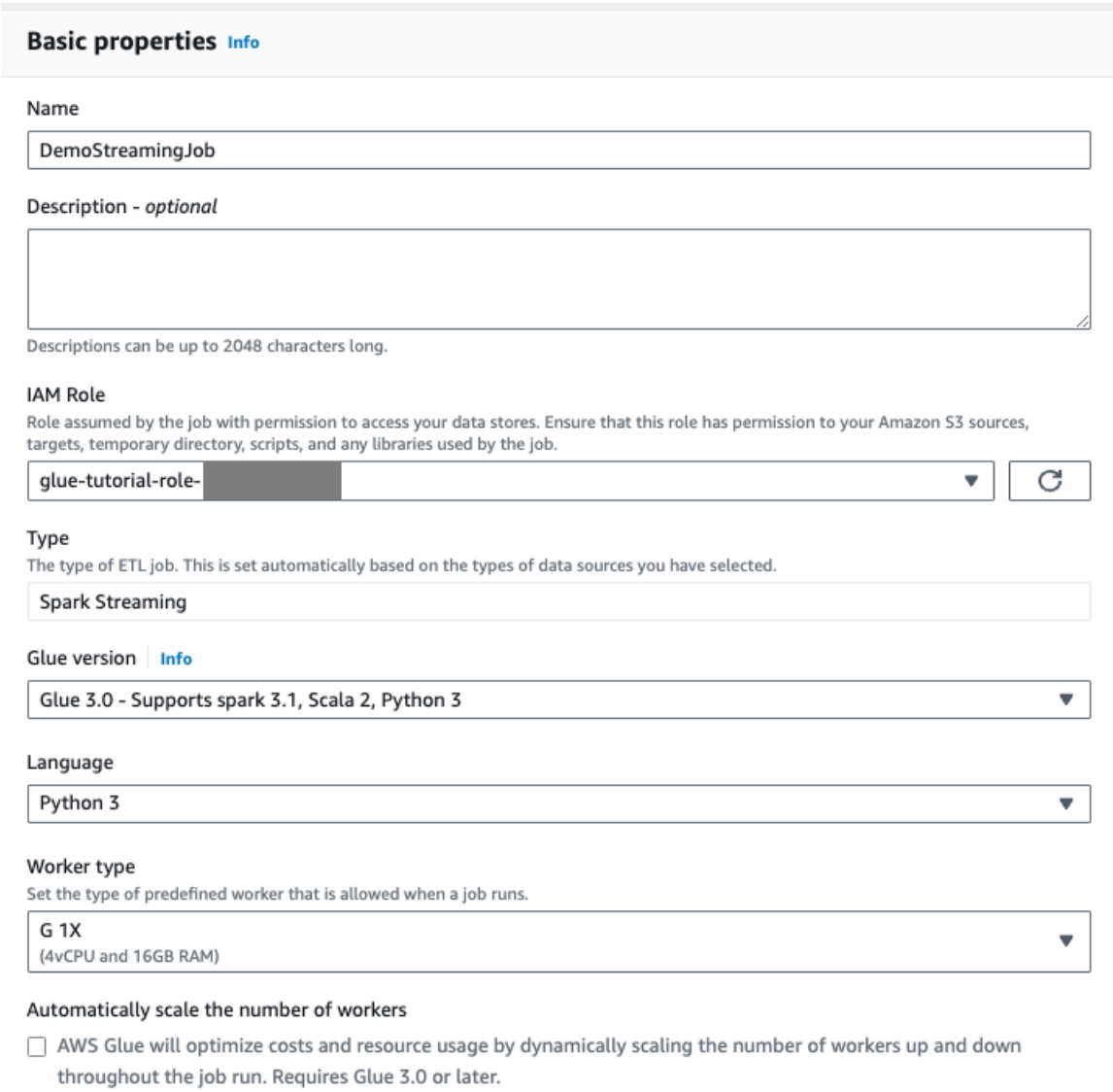
8. 데이터 전송을 클릭하고 Kinesis에 5~10,000개의 레코드를 생성합니다.

AWS Glue Studio로 AWS Glue 스트리밍 작업 생성

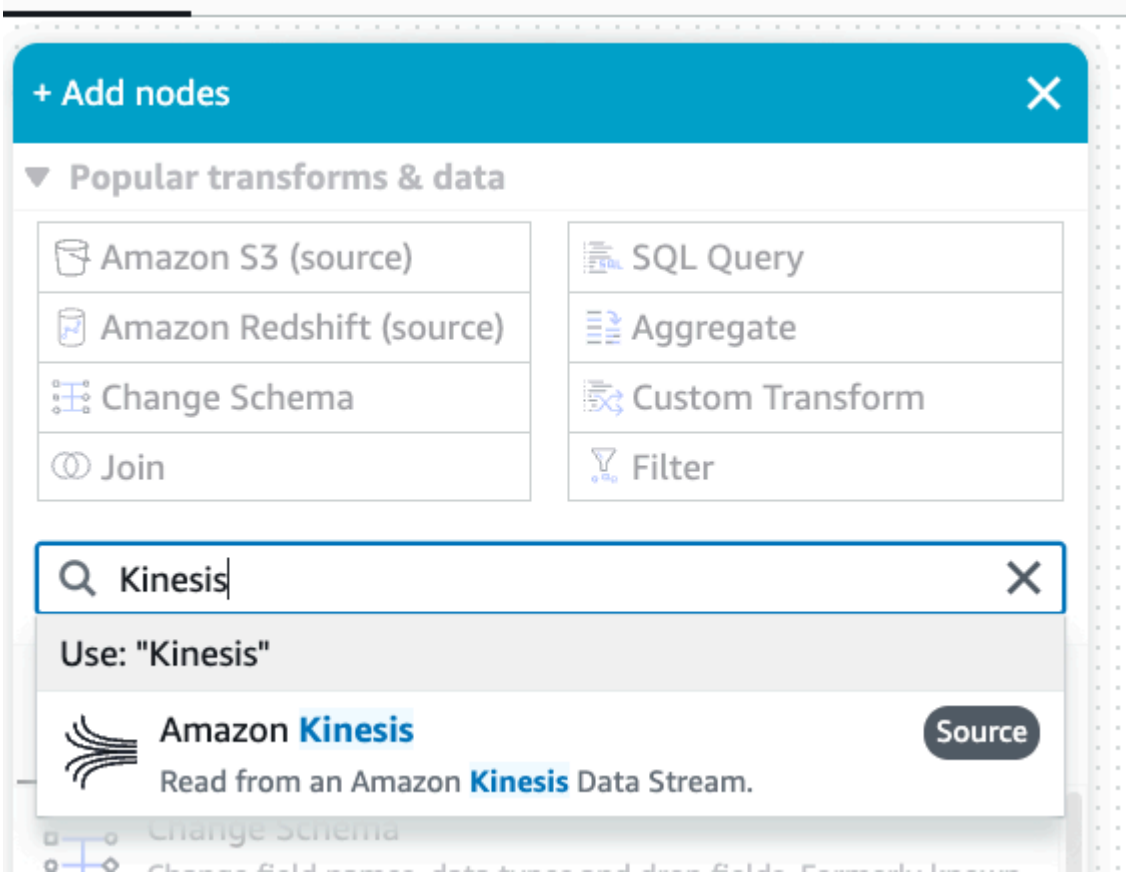
1. 동일한 리전의 콘솔에서 AWS Glue로 이동합니다.
2. 데이터 통합 및 ETL 아래의 왼쪽 탐색 표시줄에서 ETL 작업을 선택합니다.
3. 빈 캔버스가 있는 시각적 객체를 통해 AWS Glue 작업을 생성합니다.



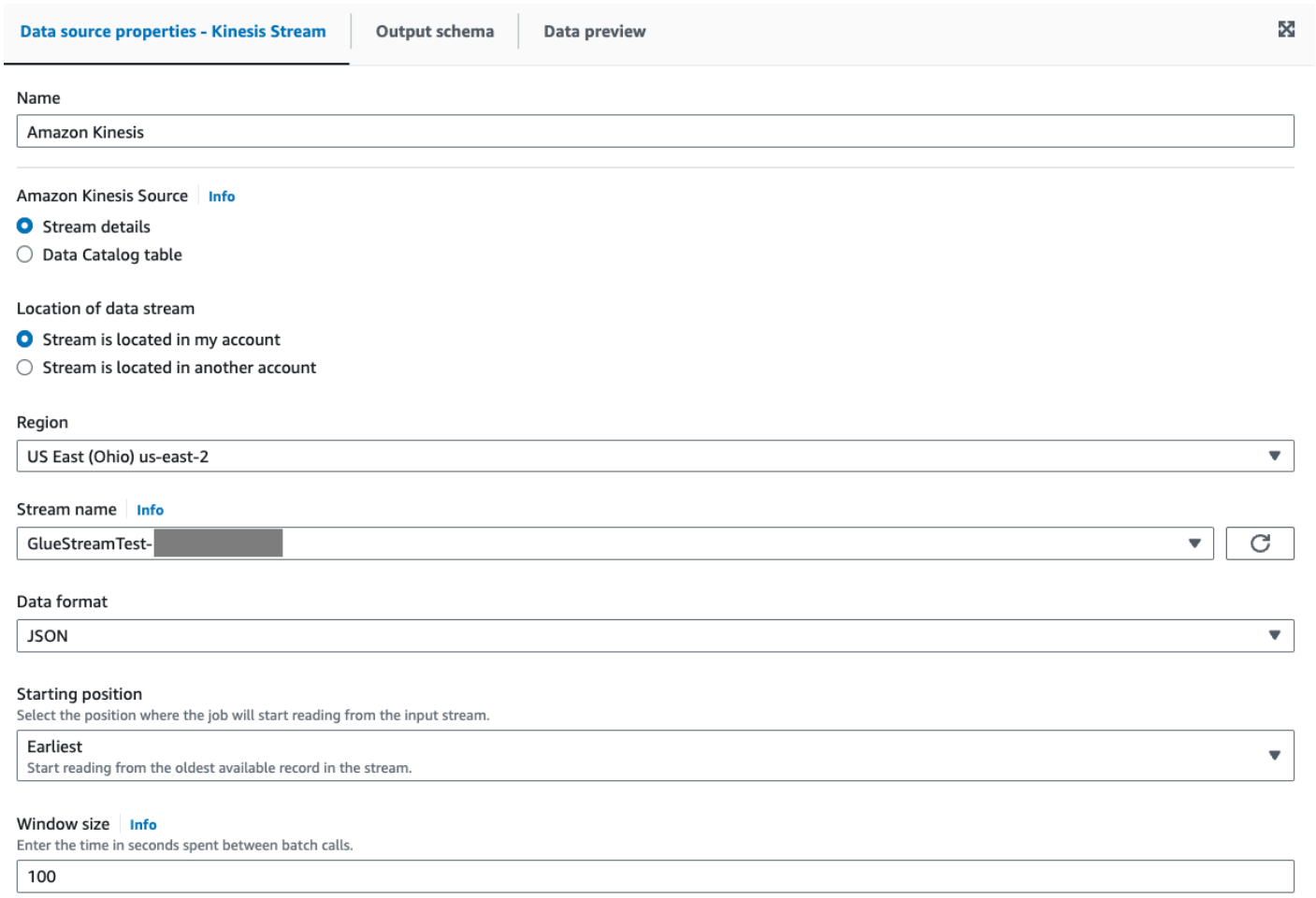
4. 작업 세부 정보 탭으로 이동합니다.
5. AWS Glue 작업 이름으로 `DemoStreamingJob`을 입력합니다.
6. IAM 역할로 CloudFormation 템플릿에서 프로비저닝한 역할인 `glue-tutorial-role-${AWS::AccountId}`를 선택합니다.
7. Glue 버전으로 Glue 3.0을 선택합니다. 다른 모든 옵션은 기본값으로 둡니다.



8. 시각적 객체 탭으로 이동합니다.
9. 더하기 아이콘을 클릭합니다. 검색 창에 Kinesis를 입력합니다. Amazon Kinesis 데이터 소스를 선택합니다.



10. 데이터 소스 속성 - Kinesis Stream 탭에서 Amazon Kinesis 소스에 대한 스트림 세부 정보를 선택합니다.
11. 데이터 스트림의 위치에서 스트림이 내 계정에 있음을 선택합니다.
12. 사용 중인 리전을 선택합니다.
13. `GlueStreamTest-{AWS::AccountId}` 스트림을 선택합니다.
14. 다른 모든 설정은 기본값으로 유지합니다.

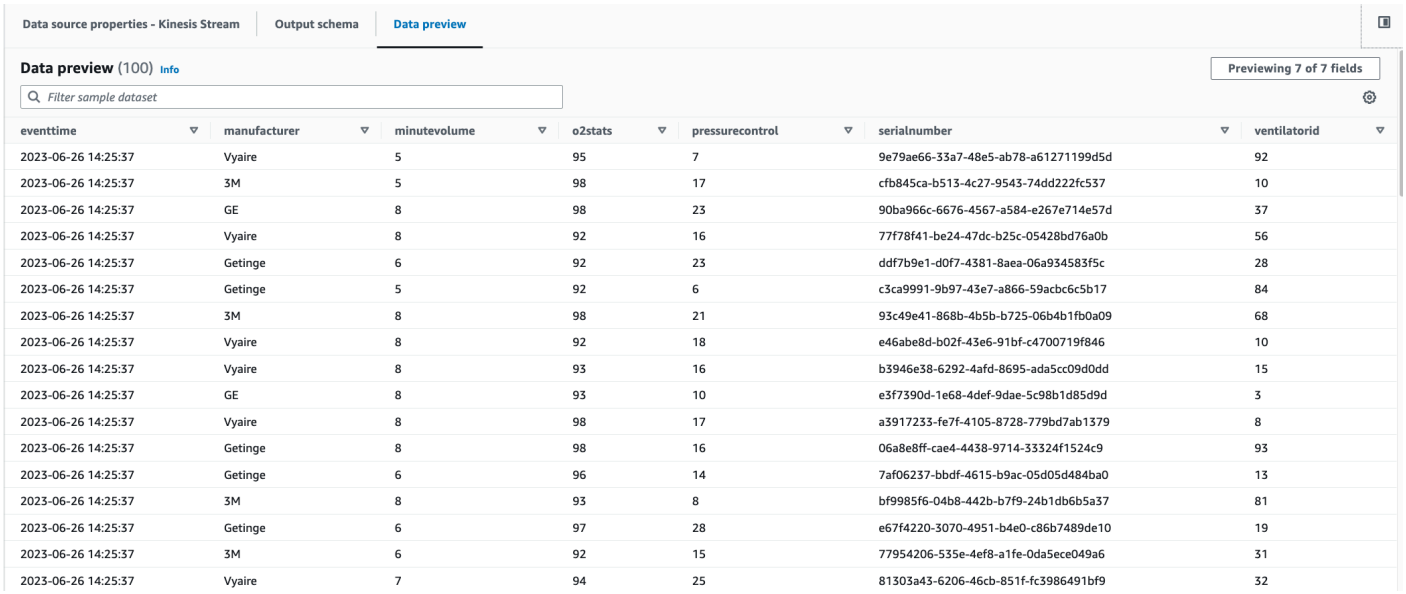


15.데이터 미리 보기 탭으로 이동합니다.

16.KDG에서 생성된 모의 데이터를 미리 보는 데이터 미리 보기 세션 시작을 클릭합니다. AWS Glue 스트리밍 작업에 대해 이전에 생성한 Glue 서비스 역할을 선택합니다.

미리 보기 데이터가 표시되는 데 30~60초 걸립니다. 표시할 데이터 없음이 표시되면 기어 아이콘을 클릭하고 샘플링할 행 수를 100으로 변경합니다.

다음과 같이 샘플 데이터를 확인할 수 있습니다.



Data source properties - Kinesis Stream | Output schema | Data preview

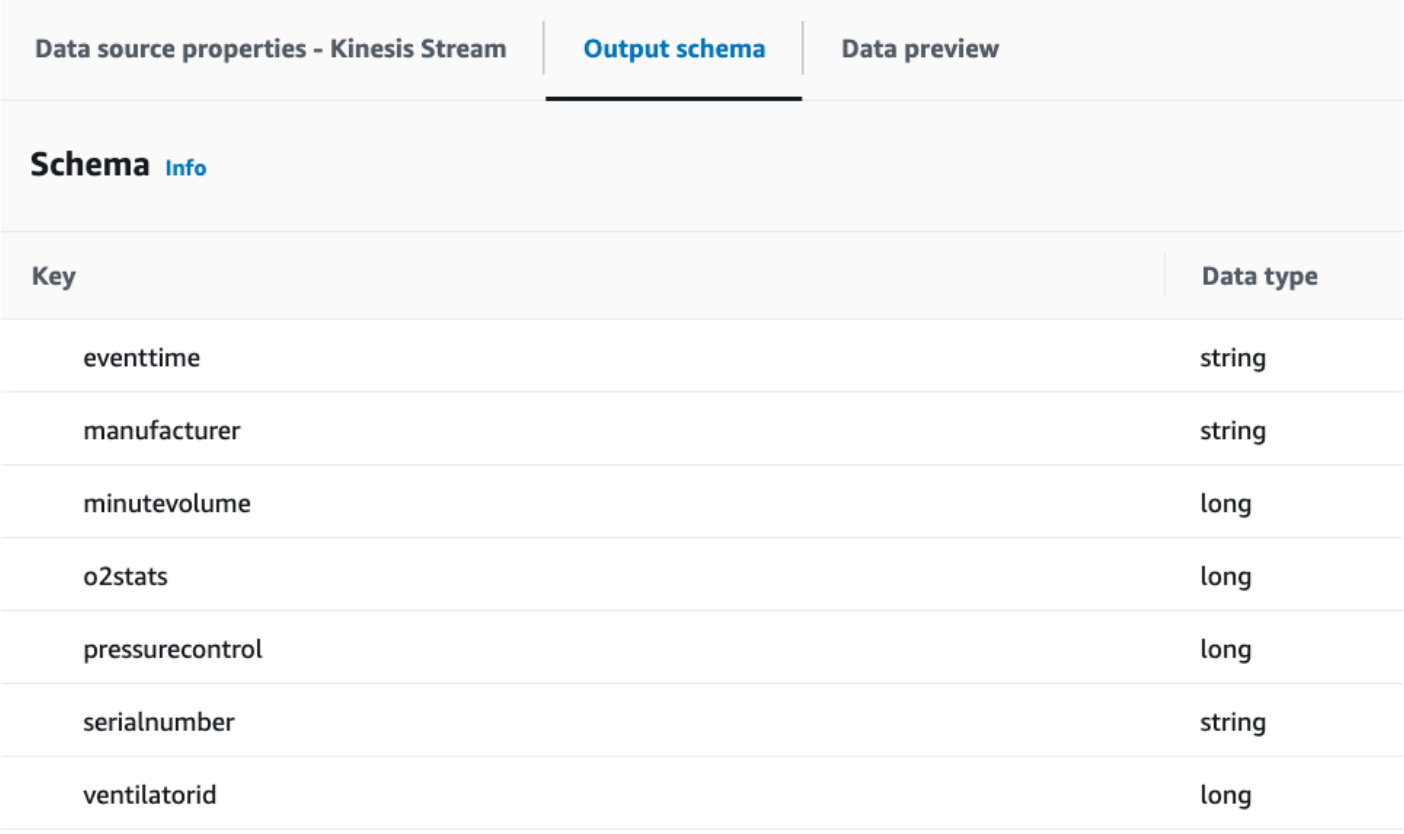
Data preview (100) Info

Filter sample dataset

Previewing 7 of 7 fields

eventtime	manufacturer	minutevolume	o2stats	pressurecontrol	serialnumber	ventilatorid
2023-06-26 14:25:37	Vyaire	5	95	7	9e79ae66-33a7-48e5-ab78-a61271199d5d	92
2023-06-26 14:25:37	3M	5	98	17	cfb845ca-b513-4c27-9543-74dd222fc537	10
2023-06-26 14:25:37	GE	8	98	23	90ba966c-6676-4567-a584-e267e714e57d	37
2023-06-26 14:25:37	Vyaire	8	92	16	77f78f41-be24-47dc-b25c-05428bd76a0b	56
2023-06-26 14:25:37	Getinge	6	92	23	ddf7b9e1-d0f7-4381-8aea-06a934583f5c	28
2023-06-26 14:25:37	Getinge	5	92	6	c3ca9991-9b97-43e7-a866-59acbc6c5b17	84
2023-06-26 14:25:37	3M	8	98	21	93c49e41-868b-4b5b-b725-06b4b1fb0a09	68
2023-06-26 14:25:37	Vyaire	8	92	18	e46abe8d-b02f-43e6-91bf-c4700719f846	10
2023-06-26 14:25:37	Vyaire	8	93	16	b3946e38-6292-4afd-8695-ada5cc09d0dd	15
2023-06-26 14:25:37	GE	8	93	10	e3f7390d-1e68-4def-9dae-5c98b1d85d9d	3
2023-06-26 14:25:37	Vyaire	8	98	17	a3917233-fe7f-4105-8728-779bd7ab1379	8
2023-06-26 14:25:37	Getinge	8	98	16	06a8e8ff-cae4-4438-9714-33324f1524c9	93
2023-06-26 14:25:37	Getinge	6	96	14	7af06237-bbdf-4615-b9ac-05d05d484ba0	13
2023-06-26 14:25:37	3M	8	93	8	bf9985f6-04b8-442b-b7f9-24b1db6b5a37	81
2023-06-26 14:25:37	Getinge	6	97	28	e67f4220-3070-4951-b4e0-c86b7489de10	19
2023-06-26 14:25:37	3M	6	92	15	77954206-535e-4ef8-a1fe-0da5ece049a6	31
2023-06-26 14:25:37	Vyaire	7	94	25	81303a43-6206-46cb-851f-fc3986491bf9	32

출력 스키마 탭에서 추론된 스키마를 확인할 수도 있습니다.



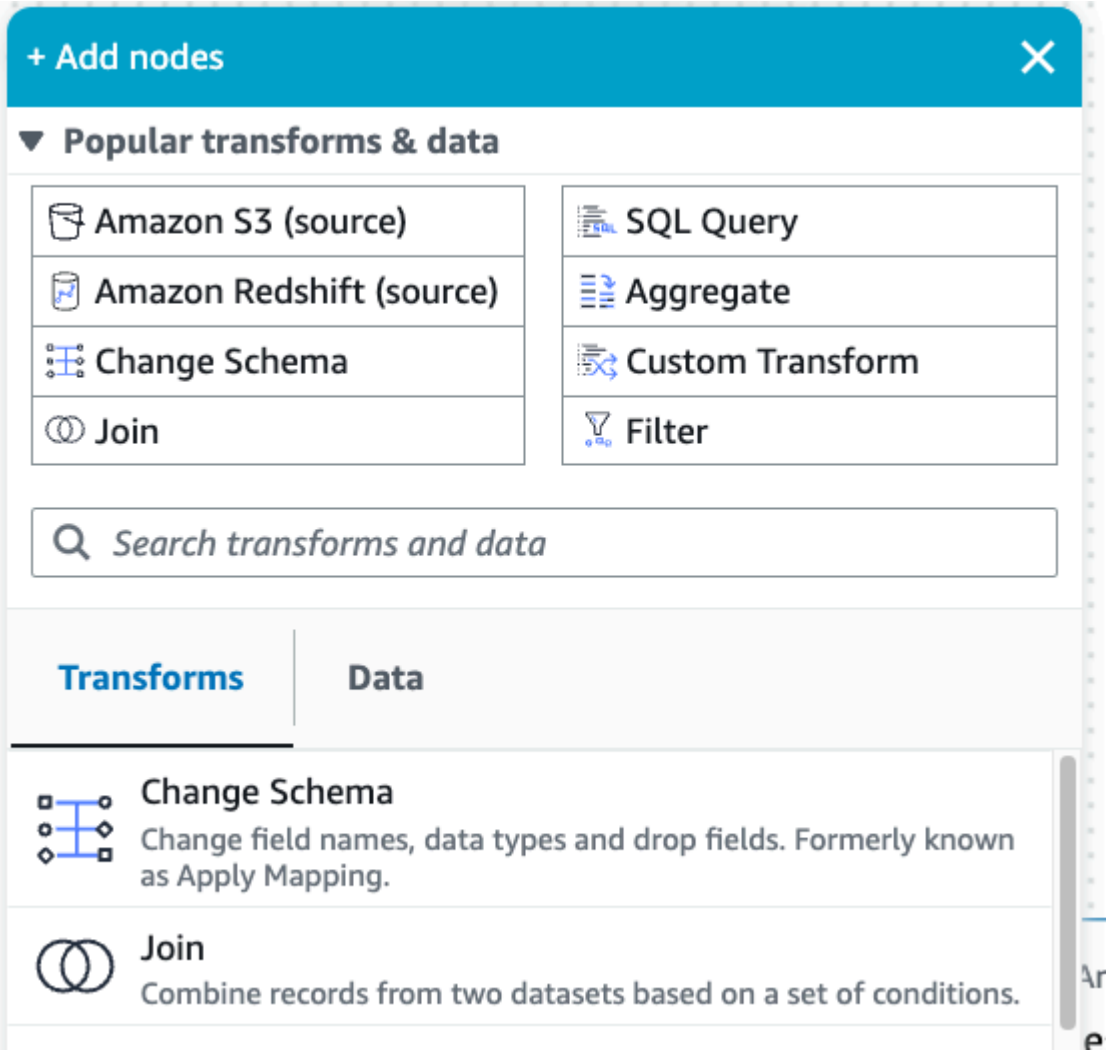
Data source properties - Kinesis Stream | Output schema | Data preview

Schema Info

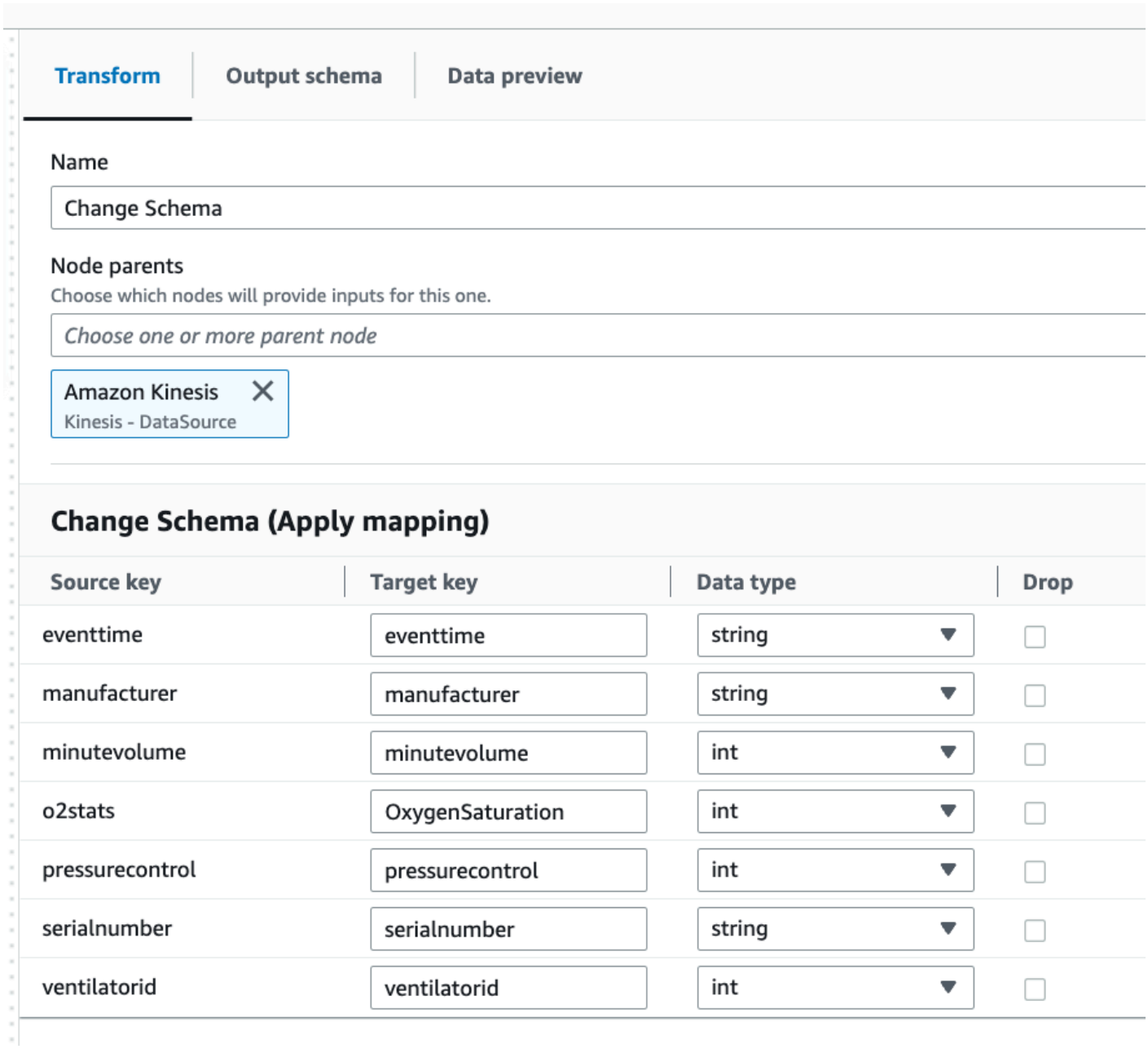
Key	Data type
eventtime	string
manufacturer	string
minutevolume	long
o2stats	long
pressurecontrol	long
serialnumber	string
ventilatorid	long

변환 수행 및 Amazon S3에 변환된 결과 저장

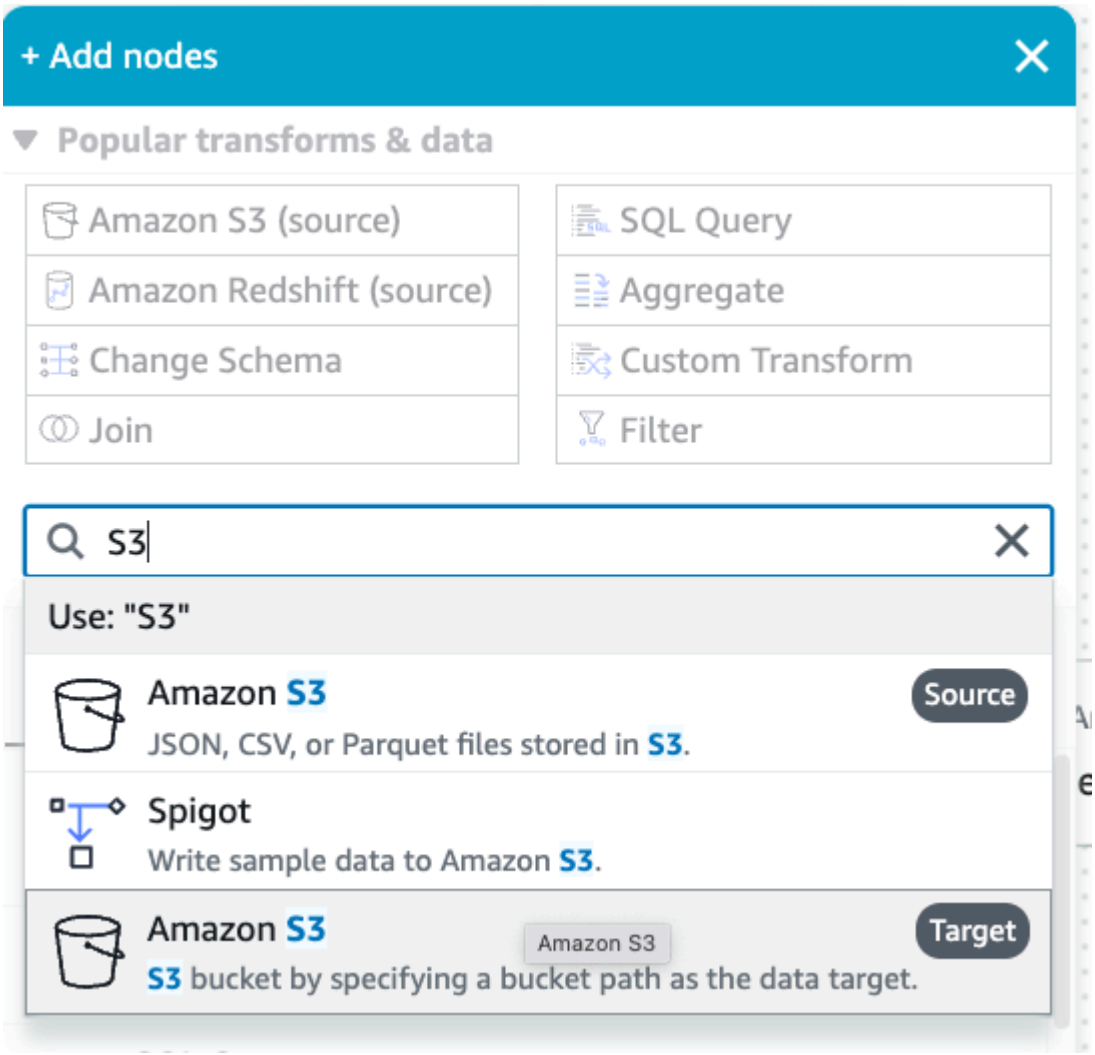
1. 소스 노드를 선택한 상태에서 왼쪽 상단의 더하기 아이콘을 클릭하여 변환 단계를 추가합니다.
2. 스키마 변경 단계를 선택합니다.



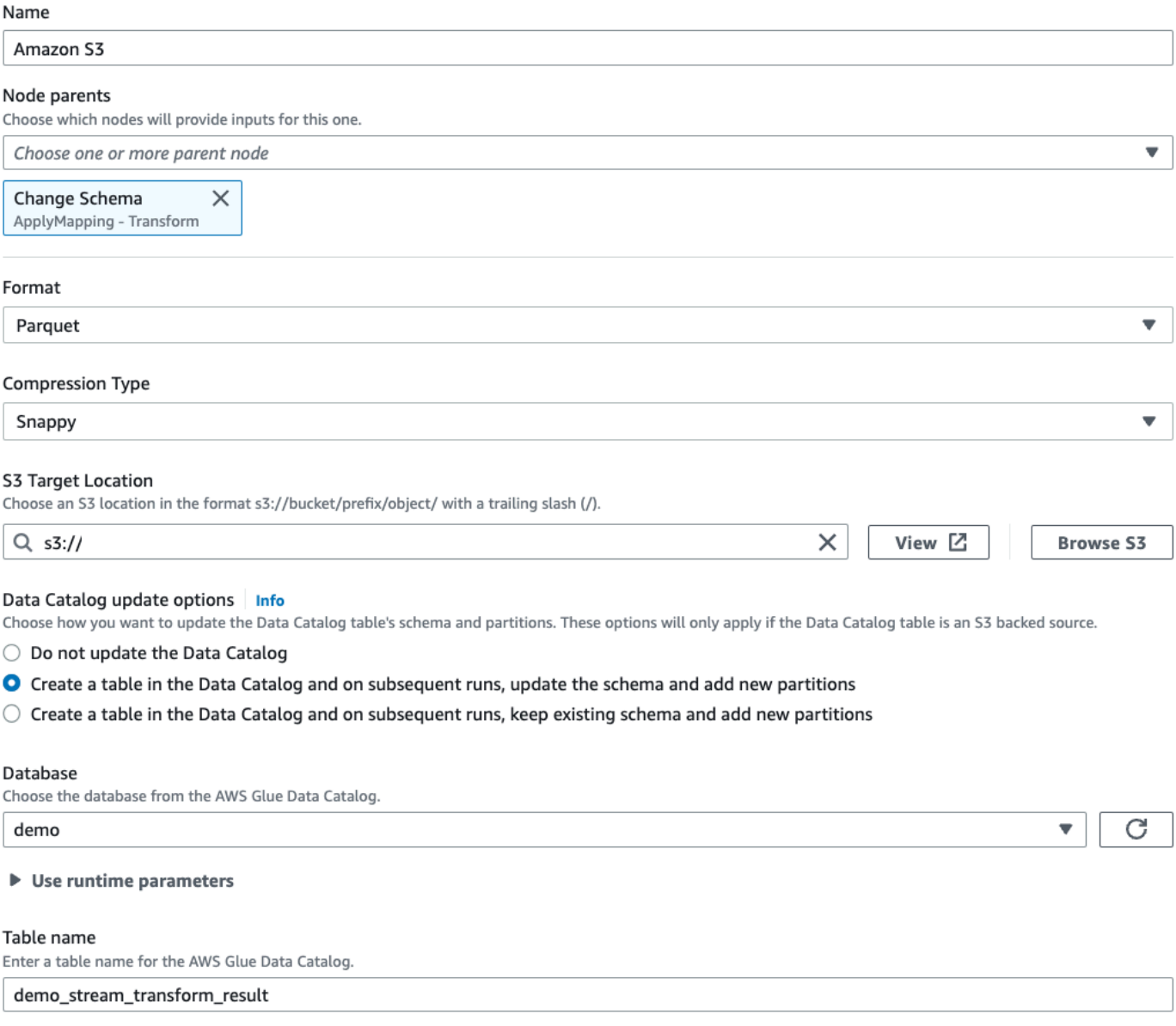
3. 이 단계에서 필드 이름을 바꾸고 필드의 데이터 형식을 변환할 수 있습니다. `o2stats` 열의 이름을 `OxygenSaturation`으로 바꾸고 모든 `long` 데이터 형식을 `int`로 변환합니다.



4. 더하기 아이콘을 클릭하여 Amazon S3 대상을 추가합니다. 검색 상자에 S3를 입력하고 Amazon S3 - 대상 변환 단계를 선택합니다.



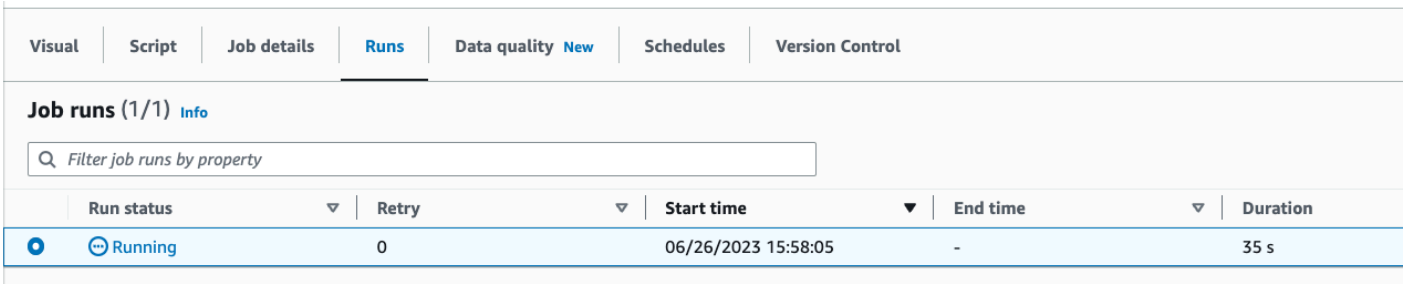
5. 대상 파일 형식으로 Parquet을 선택합니다.
6. 압축 유형으로 Snappy를 선택합니다.
7. CloudFormation 템플릿 `streaming-tutorial-s3-target-{AWS::AccountId}`로 생성된 S3 대상 위치를 입력합니다.
8. 데이터 카탈로그에 테이블 생성, 후속 실행 시 기존 스키마 유지 및 새 파티션 추가를 선택합니다.
9. Amazon S3 대상 테이블의 스키마를 저장할 대상 데이터베이스와 테이블 이름을 입력합니다.



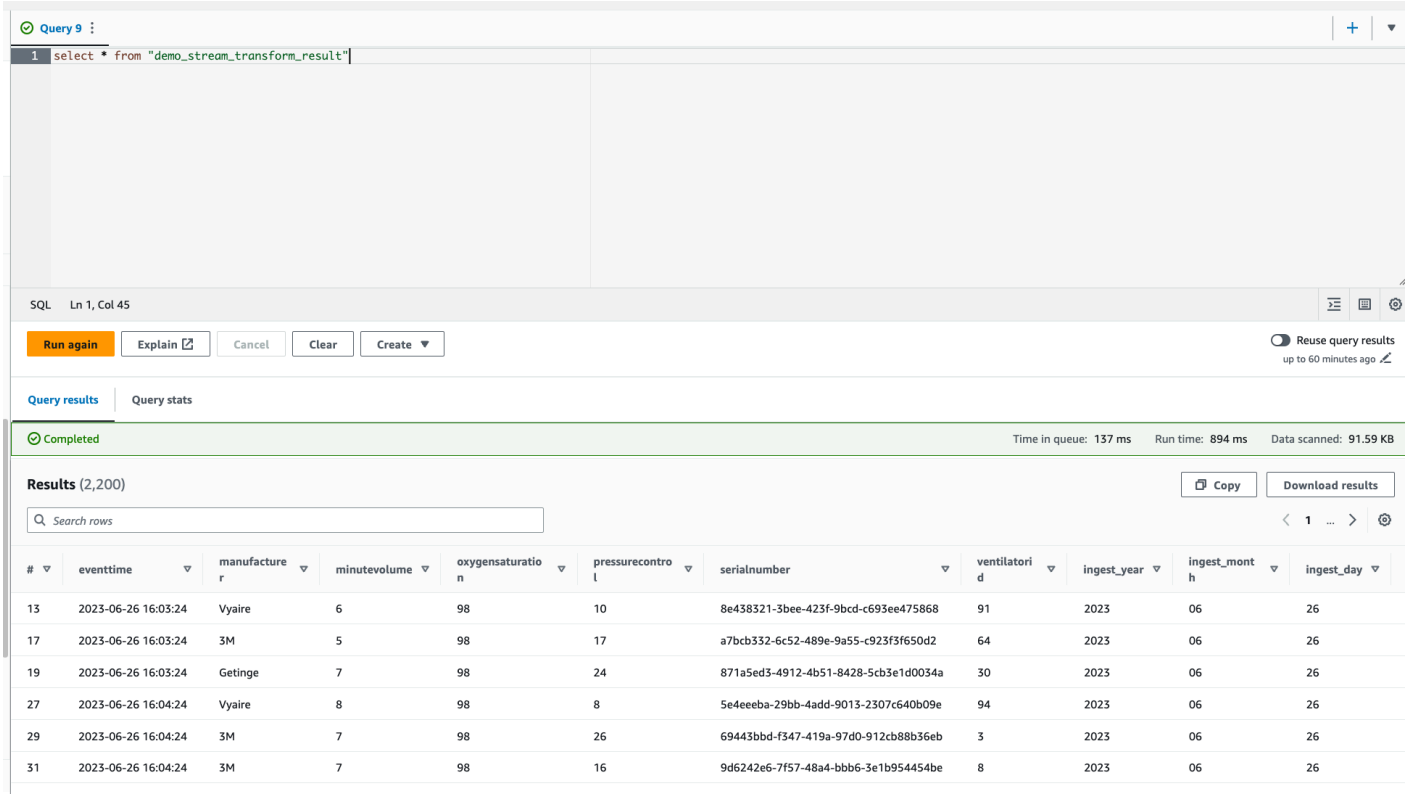
10스크립트 탭을 클릭하여 생성된 코드를 봅니다.

11오른쪽 상단의 저장을 클릭하여 ETL 코드를 저장한 다음, 실행을 클릭하여 AWS Glue 스트리밍 작업을 시작합니다.

실행 탭에서 실행 상태를 확인할 수 있습니다. 작업을 3~5분간 실행한 후 중지합니다.



12Amazon Athena에서 생성된 새 테이블을 확인합니다.



자습서: AWS Glue Studio 노트북을 사용하여 첫 번째 스트리밍 워크로드 구축

이 자습서에서는 AWS Glue Studio 노트북을 활용하여 실시간에 가까운 데이터 처리를 위해 ETL 작업을 대화형으로 구축하고 개선하는 방법을 살펴보겠습니다. AWS Glue를 처음 사용하거나 기술 세트를 향상시키고자 할 때 이 가이드는 AWS Glue 대화형 세션 노트북의 잠재력을 최대한 활용할 수 있도록 프로세스를 안내합니다.

AWS Glue 스트리밍을 사용하면 지속적으로 실행되고 Amazon Kinesis Data Streams, Apache Kafka, Amazon Managed Streaming for Apache Kafka(Amazon MSK) 등의 스트리밍 소스의 데이터를 소비하는 스트리밍 추출, 전환, 적재(ETL) 작업을 생성할 수 있습니다.

사전 조건

이 자습서를 따르려면 AWS Glue, Amazon Kinesis, Amazon S3, Amazon Athena, AWS CloudFormation, AWS Lambda 및 Amazon Cognito를 사용할 수 있는 AWS 콘솔 권한이 있는 사용자가 필요합니다.

Amazon Kinesis의 스트리밍 데이터 소비

주제

- Kinesis 데이터 생성기로 모의 데이터 생성
- AWS Glue Studio로 AWS Glue 스트리밍 작업 생성
- 정리
- 결론

Kinesis 데이터 생성기로 모의 데이터 생성

> **Note**
>
> 이전 자습서: AWS Glue Studio를 사용하여 첫 번째 스트리밍 워크로드 구축을 완료한 경우 계정에 Kinesis Data Generator가 이미 설치되어 있으므로 아래의 1~8단계를 건너뛰고 AWS Glue Studio로 AWS Glue 스트리밍 작업 생성 섹션으로 이동할 수 있습니다.

Kinesis 데이터 생성기(KDG)를 사용하여 JSON 형식의 샘플 데이터를 종합적으로 생성할 수 있습니다. 도구 설명서에서 전체 지침과 세부 정보를 찾아볼 수 있습니다.

1. 시작하려면

   Launch Stack 을 클릭하여 AWS 환경에서 AWS CloudFormation 템플릿을 실행합니다.

> **Note**
>
> Kinesis 데이터 생성기용 Amazon Cognito 사용자와 같은 일부 리소스가 AWS 계정에 이미 존재하기 때문에 CloudFormation 템플릿 오류가 발생할 수 있습니다. 다른 자습서나 블로그에서 이미 설정했기 때문일 수 있습니다. 이 문제를 해결하려면 새 AWS 계정에서 템플릿을 사용해 새로 시작하거나 다른 AWS 리전을 탐색해 봅니다. 이러한 옵션을 사용하면 기존 리소스와 충돌하지 않고 자습서를 실행할 수 있습니다.

템플릿은 Kinesis 데이터 스트림과 Kinesis Data Generator 계정을 프로비저닝합니다.

2. KDG가 인증에 사용할 사용자 이름과 암호를 입력합니다. 나중에 사용할 수 있도록 사용자 이름과 암호를 기록해 둡니다.
3. 마지막 단계까지 다음을 선택합니다. IAM 리소스 생성을 확인합니다. 화면 상단에 최소 요구 사항을 충족하지 않는 암호와 같은 오류가 있는지 확인하고 템플릿을 배포합니다.
4. 스택의 출력 탭으로 이동합니다. 템플릿이 배포되면 생성된 속성 KinesisDataGeneratorUrl이 표시됩니다. 해당 URL을 클릭합니다.
5. 기록해 둔 사용자 이름과 암호를 입력합니다.
6. 사용 중인 리전을 선택하고 Kinesis 스트림 `GlueStreamTest-{AWS::AccountId}`를 선택합니다.
7. 다음 템플릿을 입력합니다.

```
{
    "ventilatorid": {{random.number(100)}},
    "eventtime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "serialnumber": "{{random.uuid}}",
    "pressurecontrol": {{random.number(
        {
            "min":5,
            "max":30
        }
    )}},
    "o2stats": {{random.number(
        {
            "min":92,
            "max":98
        }
    )}},
```

```
    "minutevolume": {{random.number(
        {
            "min":5,
            "max":8
        }
    )}},
    "manufacturer": "{{random.arrayElement(
        ["3M", "GE","Vyaire", "Getinge"]
    )}}"
}
```

이제 테스트 템플릿을 사용하여 모의 데이터를 보고 데이터 전송을 사용하여 모의 데이터를 Kinesis로 모을 수 있습니다.

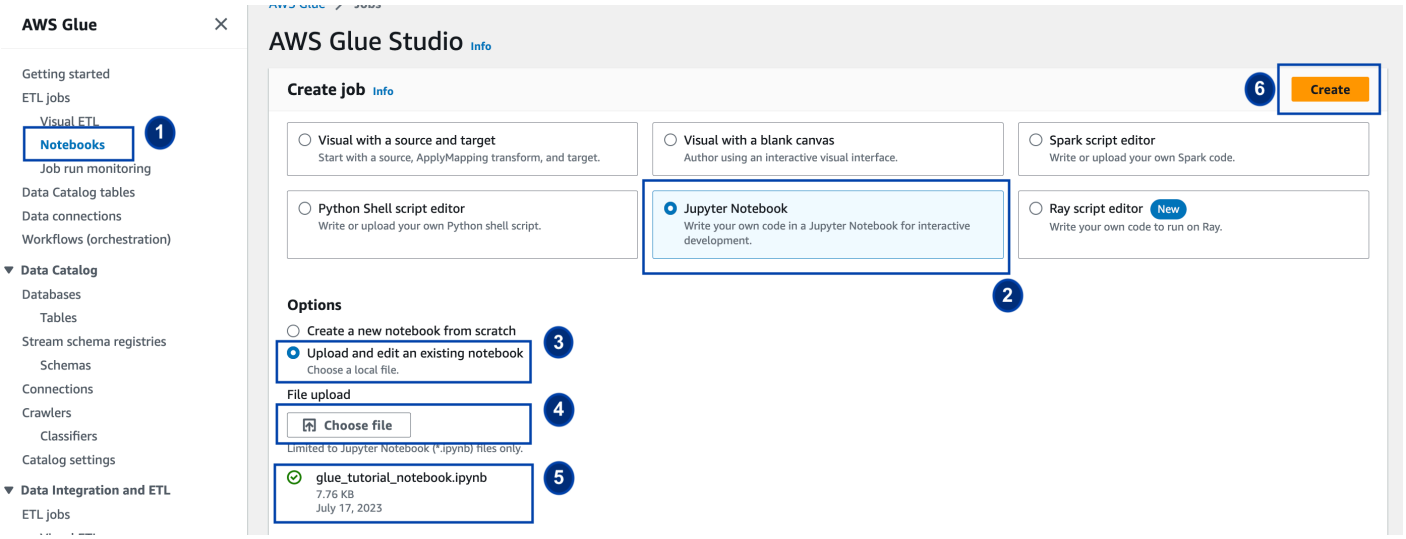
8. 데이터 전송을 클릭하고 Kinesis에 5~10,000개의 레코드를 생성합니다.

AWS Glue Studio로 AWS Glue 스트리밍 작업 생성

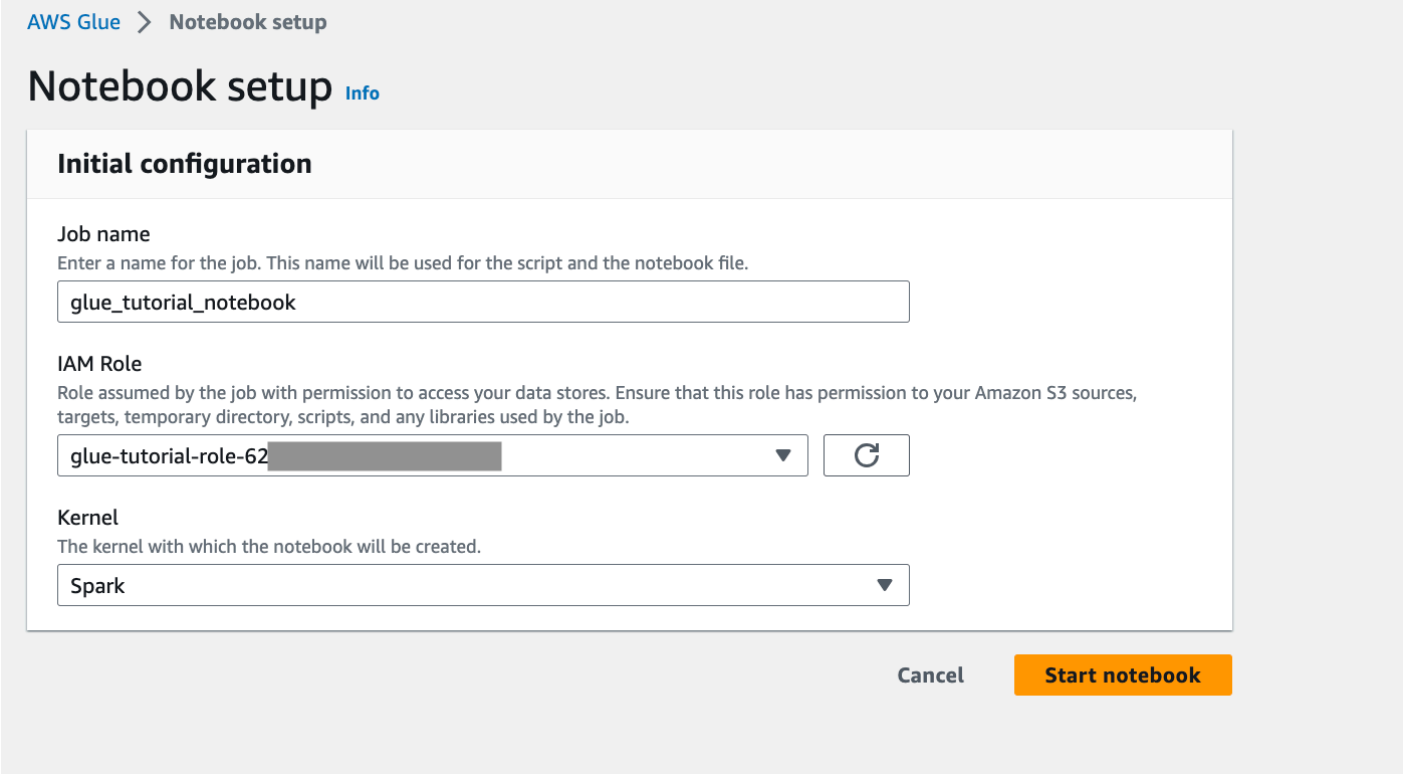
AWS Glue Studio는 데이터 통합 파이프라인의 설계, 오케스트레이션 및 모니터링 프로세스를 간소화하는 시각적 인터페이스입니다. 이를 통해 사용자는 광범위한 코드를 작성하지 않고도 데이터 변환 파이프라인을 구축할 수 있습니다. AWS Glue Studio에는 시각적 작업 작성 환경 외에도 이 자습서의 나머지 부분에서 사용할 AWS Glue 대화형 세션에서 지원하는 Jupyter Notebook도 포함되어 있습니다.

AWS Glue 스트리밍 대화형 세션 작업 설정

1. 제공된 노트북 파일을 다운로드하여 로컬 디렉토리에 저장합니다.
2. AWS Glue 콘솔을 열고 왼쪽 창에서 노트북 > Jupyter Notebook > 기존 노트북 업로드 및 편집을 클릭합니다. 이전 단계의 노트북을 업로드하고 생성을 클릭합니다.



3. 작업 이름, 역할을 제공하고 기본 Spark 커널을 선택합니다. 그런 다음, 노트북 시작을 클릭합니다. IAM 역할로 CloudFormation 템플릿에서 프로비저닝한 역할을 선택합니다. CloudFormation의 출력 탭에서 이를 확인할 수 있습니다.



노트북에는 자습서를 계속하는 데 필요한 모든 지침이 있습니다. 노트북의 지침을 실행하거나 이 자습서를 따라 작업 개발을 계속할 수 있습니다.

노트북 셀을 실행합니다.

1. (선택 사항) 첫 번째 코드 셀인 `%help`는 사용 가능한 모든 노트북 매직을 나열합니다. 지금은 이 셀을 건너뛰어도 되지만 자유롭게 탐색해 봅니다.
2. 다음 코드 블록 `%streaming`으로 시작합니다. 이 매직은 작업 유형을 스트리밍으로 설정하여 AWS Glue 스트리밍 ETL 작업을 개발, 디버깅 및 배포할 수 있게 합니다.
3. 다음 셀을 실행하여 AWS Glue 대화형 세션을 생성합니다. 출력 셀에는 세션 생성을 확인하는 메시지가 있습니다.

Run this cell to set up and start your interactive session.

```
[1]: %glue_version 3.0

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
from awsglue import DynamicFrame
from datetime import datetime
from pyspark.sql.types import StructType, StructField, StringType, LongType
from pyspark.sql.functions import lit,col,from_json
import boto3

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
```

```
Setting Glue version to: 3.0
Authenticating with environment variables and user-defined glue_role_arn: arn:aws:iam::62          0:role/glue-tutorial-role
Trying to create a Glue session for the kernel.
Worker Type: G.1X
Number of Workers: 5
Session ID:                                af
Job Type: gluestreaming
Applying the following default arguments:
--glue_kernel_version 0.37.3
--enable-glue-datacatalog true
Waiting for session 4                                  to get into ready status...
Session 48                                 has been created.
```

4. 다음 셀은 변수를 정의합니다. 값을 작업에 적합한 값으로 바꾸고 셀을 실행합니다. 예:

```
output_database_name="default"
output_table_name="test_stream_001"

account_id = boto3.client("sts").get_caller_identity()["Account"]
region_name=boto3.client('s3').meta.region_name
stream_arn_name = "arn:aws:kinesis:{}:{}:stream/GlueStreamTest-{}".format(region_name,account_id,account_id)
s3_bucket_name = "streaming-tutorial-s3-target-{}".format(account_id)

output_location = "s3://{}/streaming_output/".format(s3_bucket_name)
checkpoint_location = "s3://{}/checkpoint_location/".format(s3_bucket_name)
```

5. 데이터가 이미 Kinesis Data Streams로 스트리밍되고 있으므로 다음 셀은 스트림의 결과를 소비합니다. 다음 셀을 실행합니다. 인쇄 문이 없으므로 이 셀에서 예상되는 출력이 없습니다.
6. 다음 셀에서는 샘플 세트를 가져와서 수신 스트림을 탐색하고 해당 스키마와 실제 데이터를 인쇄합니다. 예:

Sample and print the incoming records

the sampling is for debugging purpose. You may comment off the entire code cell below, before deploying the actual code

[4]:
```
options = {
    "pollingTimeInMs": "20000",
    "windowSize": "5 seconds"
}
sampled_dynamic_frame = glueContext.getSampleStreamingDynamicFrame(data_frame, options, None)

count_of_sampled_records = sampled_dynamic_frame.count()

print(count_of_sampled_records)

sampled_dynamic_frame.printSchema()

sampled_dynamic_frame.toDF().show(10,False)
```

```
100
root
|-- eventtime: string
|-- manufacturer: string
|-- minutevolume: long
|-- o2stats: long
|-- pressurecontrol: long
|-- serialnumber: string
|-- ventilatorid: long

+-------------------+------------+------------+-------+---------------+------------------------------------+------------+
|eventtime          |manufacturer|minutevolume|o2stats|pressurecontrol|serialnumber                        |ventilatorid|
+-------------------+------------+------------+-------+---------------+------------------------------------+------------+
|2023-07-18 10:20:11|3M          |6           |92     |24             |a3e860ba-24b9-41c4-bc10-91c6b35e1406|6           |
|2023-07-18 10:20:11|Vyaire      |6           |95     |6              |96101dca-3e88-457f-b390-e3291df48a81|26          |
|2023-07-18 10:20:12|Getinge     |8           |96     |24             |18f3d448-1dee-4c80-835b-1a0daa818915|22          |
|2023-07-18 10:20:12|Getinge     |7           |98     |30             |25f425cd-b978-4953-9a03-4d607a639364|91          |
|2023-07-18 10:20:12|GE          |5           |93     |25             |2cd7cdc2-f5f5-4ff2-ae32-45e5a8922d53|93          |
```

7. 다음으로 실제 데이터 변환 로직을 정의합니다. 셀은 모든 마이크로 배치 중 트리거되는 `processBatch` 메서드로 구성됩니다. 셀을 실행합니다. 높은 수준에서 수신 스트림에 대해 다음을 수행합니다.

   a. 입력 열의 하위 집합을 선택합니다.

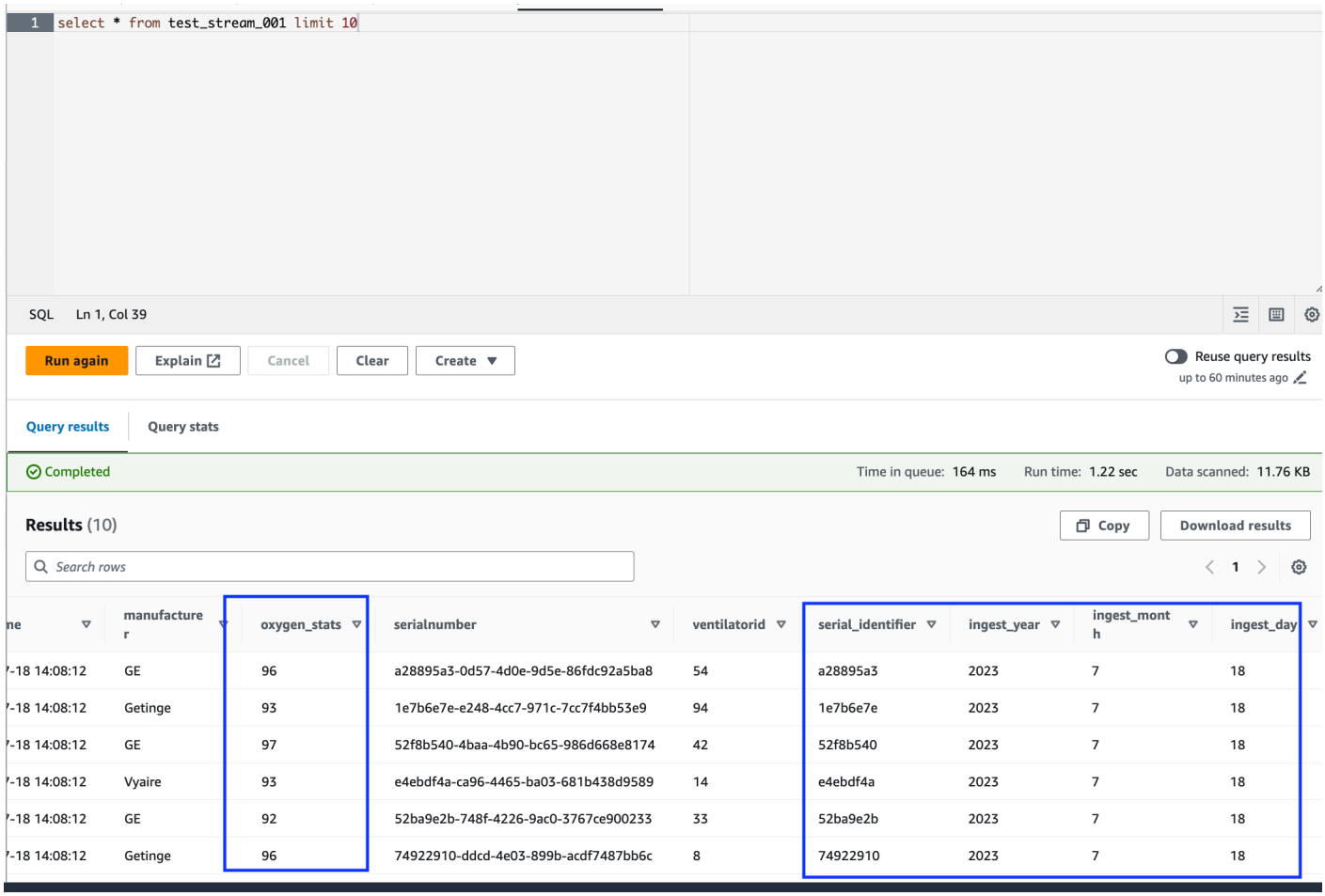
   b. 열 이름을 o2stats에서 oxygen_stats로 바꿉니다.

   c. 새 열(serial_identifier, ingest_year, ingest_month 및 ingest_day)을 파생합니다.

   d. 결과를 Amazon S3 버킷에 저장하고 분할된 AWS Glue 카탈로그 테이블도 생성합니다.

8. 마지막 셀에서는 10초마다 프로세스 배치를 트리거합니다. 셀을 실행하고 Amazon S3 버킷과 AWS Glue 카탈로그 테이블이 채워질 때까지 약 30초간 기다립니다.

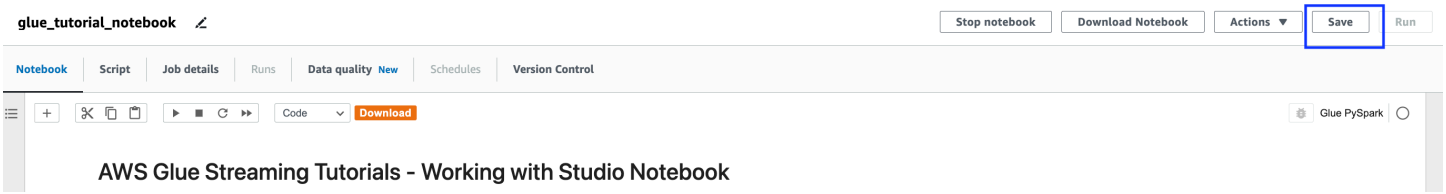
9. 마지막으로 Amazon Athena 쿼리 편집기를 사용하여 저장된 데이터를 찾아봅니다. 이름이 바뀐 열과 새 파티션도 볼 수 있습니다.



노트북에는 자습서를 계속하는 데 필요한 모든 지침이 있습니다. 노트북의 지침을 실행하거나 이 자습서를 따라 작업 개발을 계속할 수 있습니다.

AWS Glue 작업을 저장하고 실행합니다.

대화형 세션 노트북을 사용하여 애플리케이션 개발 및 테스트가 완료되면 노트북 인터페이스 상단에 있는 저장을 클릭합니다. 저장한 후에는 애플리케이션을 작업으로 실행할 수도 있습니다.



정리

계정에 추가 요금이 발생하지 않도록 하려면 지침의 일부로 시작한 스트리밍 작업을 중지합니다. 노트북을 중지하면 됩니다. 그러면 세션이 종료됩니다. Amazon S3 버킷을 비우고 이전에 프로비저닝한 AWS CloudFormation 스택을 삭제합니다.

결론

이 자습서에서는 AWS Glue Studio Notebook을 사용하여 다음을 수행하는 방법을 시연했습니다.

- 노트북을 사용하여 스트리밍 ETL 작업 작성
- 수신 데이터 스트림 미리 보기
- AWS Glue 작업을 게시할 필요 없이 코딩 및 문제 수정
- 엔드 투 엔드 작업 코드를 검토하고, 디버깅을 제거하고, 노트북에서 명령문이나 셀을 인쇄합니다.
- 코드를 AWS Glue 작업으로 게시합니다.

이 자습서의 목표는 AWS Glue 스트리밍 및 대화형 세션을 사용한 실습 경험을 제공하는 것입니다. 이를 개별 AWS Glue 스트리밍 사용 사례에 대한 참조로 사용하는 것이 좋습니다. 자세한 내용은 AWS Glue 대화형 세션 시작하기 단원을 참조하십시오.

AWS Glue 스트리밍 개념

다음 섹션에서는 AWS Glue 스트리밍의 개념에 대한 정보를 제공합니다.

주제

- AWS Glue 스트리밍 작업의 구조

AWS Glue 스트리밍 작업의 구조

AWS Glue 스트리밍 작업은 Spark 스트리밍 패러다임에서 작동하며 Spark 프레임워크의 구조화된 스트리밍을 활용합니다. 스트리밍 작업은 특정 시간 간격으로 스트리밍 데이터 소스를 지속적으로 폴링하여 레코드를 마이크로 배치로 가져옵니다. 다음 섹션에서는 AWS Glue 스트리밍 작업의 여러 부분을 살펴봅니다.

forEachBatch

`forEachBatch` 메서드는 AWS Glue 스트리밍 작업 실행의 진입점입니다. AWS Glue 스트리밍 작업은 `forEachBatch` 메서드를 사용하여 스트리밍 작업의 수명 주기 동안 활성 상태로 유지되는 반복자처럼 작동하는 데이터를 폴링하고 정기적으로 새 데이터에 대한 스트리밍 소스를 폴링하고 최신 데이터를 마이크로 배치로 처리합니다.

```
glueContext.forEachBatch(
    frame=dataFrame_AmazonKinesis_node1696872487972,
    batch_function=processBatch,
    options={
        "windowSize": "100 seconds",
        "checkpointLocation": args["TempDir"] + "/" + args["JOB_NAME"] + "/
checkpoint/",
    },
)
```

forEachBatch의 frame 속성을 구성하여 스트리밍 소스를 지정합니다. 이 예제에서는 작업을 생성하는 동안 빈 캔버스에서 생성한 소스 노드가 작업의 기본 DataFrame으로 채워집니다. 각 마이크로 배치 작업에 대해 간접적으로 호출하기로 결정한 function으로 batch_function 속성을 설정합니다. 수신 데이터에 대한 배치 변환을 처리할 함수를 정의해야 합니다.

소스

processBatch 함수의 첫 번째 단계에서 프로그램은 forEachBatch의 프레임 속성으로 정의한 DataFrame의 레코드 수를 확인합니다. 프로그램은 비어 있지 않은 DataFrame에 모으기 타임스탬프를 추가합니다. data_frame.count()>0 절은 최신 마이크로 배치가 비어 있지 않고 추가 처리를 위한 준비가 되었는지 여부를 결정합니다.

```
def processBatch(data_frame, batchId):
    if data_frame.count() >0:
        AmazonKinesis_node1696872487972 = DynamicFrame.fromDF(
            glueContext.add_ingestion_time_columns(data_frame, "hour"),
            glueContext,
            "from_data_frame",
        )
```

Mapping

프로그램의 다음 섹션은 매핑을 적용하는 것입니다. Spark DataFrame의 Mapping.apply 메서드를 사용하면 데이터 요소에 대한 변환 규칙을 정의할 수 있습니다. 일반적으로 이름을 바꾸거나, 데이터 형식을 변경하거나, 소스 데이터 열에 사용자 지정 함수를 적용하고 이를 대상 열에 매핑할 수 있습니다.

```
    #Script generated for node ChangeSchema
    ChangeSchema_node16986872679326 = ApplyMapping.apply(
        frame =  AmazonKinesis_node1696872487972,
        mappings = [
            ("eventtime", "string", "eventtime", "string"),
            ("manufacturer", "string", "manufacturer", "string"),
            ("minutevolume", "long", "minutevolume", "int"),
            ("o2stats", "long", "OxygenSaturation", "int"),
            ("pressurecontrol", "long", "pressurecontrol", "int"),
            ("serialnumber", "string", "serialnumber", "string"),
            ("ventilatorid", "long", "ventilatorid", "long"),
            ("ingest_year", "string", "ingest_year", "string"),
```

```
            ("ingest_month", "string", "ingest_month", "string"),
            ("ingest_day", "string", "ingest_day", "string"),
            ("ingest_hour", "string", "ingest_hour", "string"),
        ],
        transformation_ctx="ChangeSchema_node16986872679326",
    )
        )
```

Sink

이 섹션에서는 스트리밍 소스에서 수신 데이터 세트가 대상 위치에 저장됩니다. 이 예제에서는 Amazon S3 위치에 데이터를 씁니다. 캔버스에서 작업을 생성할 때 사용한 설정에 따라 `AmazonS3_node_path` 속성 세부 정보가 미리 채워집니다. 사용 사례에 따라 `updateBehavior`를 설정할 수 있습니다. 그런 다음, 데이터 카탈로그 테이블을 업데이트하지 않거나, 데이터 카탈로그를 생성하고 후속 실행 시 데이터 카탈로그 스키마를 업데이트하거나 카탈로그 테이블을 생성하고 후속 실행 시 스키마 정의를 업데이트하지 않도록 결정할 수 있습니다.

`partitionKeys` 속성은 스토리지 파티션 옵션을 정의합니다. 기본 동작은 소스 섹션에서 사용할 수 있게 된 `ingestion_time_columns` 단위로 데이터를 분할하는 것입니다. `compression` 속성을 사용하면 대상 쓰기 중 적용할 압축 알고리즘을 설정할 수 있습니다. Snappy, LZO 또는 GZIP을 압축 기법으로 설정할 수 있는 옵션이 있습니다. `enableUpdateCatalog` 속성은 AWS Glue 카탈로그 테이블을 업데이트해야 하는지 여부를 제어합니다. 이 속성에 사용할 수 있는 옵션은 `True` 또는 `False`입니다.

```
    #Script generated for node Amazon S3
    AmazonS3_node1696872743449 = glueContext.getSink(
        path =  AmazonS3_node1696872743449_path,
        connection_type = "s3",
        updateBehavior = "UPDATE_IN_DATABASE",
        partitionKeys = ["ingest_year", "ingest_month", "ingest_day", "ingest_hour"],
        compression = "snappy",
        enableUpdateCatalog = True,
        transformation_ctx = "AmazonS3_node1696872743449",
    )
```

AWS Glue 카탈로그 싱크

이 작업 섹션은 AWS Glue 카탈로그 테이블 업데이트 동작을 제어합니다. AWS Glue 카탈로그 데이터베이스 이름과 설계 중인 AWS Glue 작업과 연결된 테이블 이름에 따라 `catalogDatabase` 및 `catalogTableName` 속성을 설정합니다. `setFormat` 속성을 통해 대상 데이터의 파일 형식을 정의할 수 있습니다. 이 예제에서는 데이터를 Parquet 형식으로 저장합니다.

이 자습서를 참조하여 AWS Glue 스트리밍 작업을 설정하고 실행하면 Amazon Kinesis Data Streams에서 생성된 스트리밍 데이터가 빠른 압축을 통해 Parquet 형식으로 Amazon S3 위치에 저장됩니다. 스트리밍 작업이 성공적으로 실행되면 Amazon Athena를 통해 데이터를 쿼리할 수 있습니다.

```
AmazonS3_node1696872743449 = setCatalogInfo(
    catalogDatabase =  "demo", catalogTableName = "demo_stream_transform_result"
)
AmazonS3_node1696872743449.setFormat("glueparquet")
AmazonS3_node1696872743449.writeFormat("ChangeSchema_node16986872679326")
)
```

AWS Glue 스트리밍 연결

다음 섹션에서는 AWS Glue 스트리밍에서 연결을 사용하는 방법에 대한 정보를 제공합니다.

주제

- Kafka 연결 관련 작업
- Kinesis 연결 관련 작업

Kafka 연결 관련 작업

Data Catalog 테이블에 저장된 정보를 사용하거나 데이터 스트림에 직접 액세스할 수 있는 정보를 제공하여 Kafka 연결을 통해 Kafka Data Streams에서 읽고 쓸 수 있습니다. 연결에서는 Kafka 클러스터 또는 Amazon Managed Streaming for Apache Kafka 클러스터를 지원합니다. Kafka의 정보를 Spark DataFrame으로 읽은 다음 AWS Glue DynamicFrame으로 변환할 수 있습니다. DynamicFrame을 JSON 형식으로 Kafka에 쓸 수 있습니다. 데이터 스트림에 직접 액세스하는 경우 이러한 옵션을 사용하여 데이터 스트림에 액세스하는 방법에 대한 정보를 제공합니다.

getCatalogSource 또는 create_data_frame_from_catalog를 사용하여 Kafka 스트리밍 소스에서 레코드를 소비하거나 getCatalogSink 또는 write_dynamic_frame_from_catalog를 사용하여 Kafka에 레코드를 쓰는 경우 작업에는 데이터 카탈로그 데이터베이스와 테이블 이름 정보가 있으며 이를 사용하여 Kafka 스트리밍 소스에서 읽기 위한 몇 가지 기본 파라미터를 얻을 수 있습니다. getSource, getCatalogSink, getSourceWithFormat, getSinkWithFormat, createDataFrameFromOptions, create_data_frame_from_options 또는 write_dynamic_frame_from_catalog를 사용하는 경우 여기에 설명된 연결 옵션을 통해 이러한 기본 파라미터를 지정해야 합니다.

GlueContext 클래스에 지정된 메서드에 대해 다음 인수를 사용하여 Kafka에 대한 연결 옵션을 지정할 수 있습니다.

- Scala
  - connectionOptions: getSource, createDataFrameFromOptions, getSink와(과) 함께 사용
  - additionalOptions: getCatalogSource, getCatalogSink와 함께 사용
  - options: getSourceWithFormat, getSinkWithFormat와 함께 사용
- Python
  - connection_options: create_data_frame_from_options, write_dynamic_frame_from_options와 함께 사용
  - additional_options: create_data_frame_from_catalog, write_dynamic_frame_from_catalog와 함께 사용
  - options: getSource, getSink와 함께 사용

스트리밍 ETL 작업에 대한 참고 및 제한 사항은 the section called “스트리밍 ETL 참고 사항 및 제한 사항” 섹션을 참조하세요.

Kafka 구성

인터넷을 통해 사용할 수 있는 Kafka 스트림에 연결하기 위한 AWS의 필수 조건은 없습니다.

AWS Glue Kafka 연결을 생성하여 연결 보안 인증을 관리할 수 있습니다. 자세한 내용은 the section called “Kafka 데이터 스트림에 대한 연결 생성” 단원을 참조하십시오. AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공하고 메서드 직접 호출에서 connectionName 파라미터로 *ConnectionName*을 제공합니다.

경우에 따라 추가 필수 조건을 구성해야 합니다.

- IAM 인증과 함께 Amazon Managed Streaming for Apache Kafka를 사용하는 경우 적절한 IAM 구성이 필요합니다.
- Amazon VPC 내에서 Amazon Managed Streaming for Apache Kafka를 사용하는 경우 적절한 Amazon VPC 구성이 필요합니다. Amazon VPC 연결 정보를 제공하는 AWS Glue 연결을 생성해야 합니다. AWS Glue 연결을 추가 네트워크 연결로 포함하려면 작업 구성이 필요합니다.

스트리밍 ETL 작업 필수 조건에 대한 자세한 내용은 the section called “스트리밍 ETL 작업” 섹션을 참조하세요.

예제: Kafka 스트림에서 읽기

the section called “forEachBatch”과(와) 함께 사용합니다.

Kafka 스트리밍 소스의 예:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

예: Kafka 스트림에 쓰기

Kafka에 쓰기의 예제:

`getSink` 메서드를 사용한 예제:

```
data_frame_datasource0 =
glueContext.getSink(
 connectionType="kafka",
 connectionOptions={
  JsonOptions("""{
   "connectionName": "ConfluentKafka",
   "classification": "json",
   "topic": "kafka-auth-topic",
   "typeOfData": "kafka"}
```

```
  """)},
 transformationContext="dataframe_ApacheKafka_node1711729173428")
 .getDataFrame()
```

write_dynamic_frame.from_options 메서드를 사용한 예제:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.write_dynamic_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Kafka 연결 옵션 참조

읽을 때는 "connectionType": "kafka"를 사용한 다음 연결 옵션을 사용합니다.

- "bootstrap.servers" (필수) 부트스트랩 서버 URL의 목록입니다(예: b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094). 이 옵션은 API 호출에 지정하거나 데이터 카탈로그의 테이블 메타데이터에 정의해야 합니다.
- "security.protocol" (필수) 브로커와 통신하는 데 사용되는 프로토콜입니다. 가능한 값은 "SSL" 또는 "PLAINTEXT"입니다.
- "topicName": (필수) 쉼표로 구분된 구독할 주제의 목록입니다. "topicName", "assign" 또는 "subscribePattern" 중에서 하나만 지정해야 합니다.
- "assign": (필수) 사용할 특정 TopicPartitions을 지정하는 JSON 문자열입니다. "topicName", "assign" 또는 "subscribePattern" 중에서 하나만 지정해야 합니다.

  예: '{"topicA":[0,1],"topicB":[2,4]}'
- "subscribePattern": (필수) 구독할 주제 목록을 식별하는 Java 정규식 문자열입니다. "topicName", "assign" 또는 "subscribePattern" 중에서 하나만 지정해야 합니다.

  예: 'topic.*'
- "classification"(필수) 레코드의 데이터에서 사용하는 파일 형식입니다. 데이터 카탈로그를 통해 제공되지 않는 한 필수입니다.
- "delimiter"(선택 사항) classification이 CSV일 때 사용되는 값 구분 기호입니다. 기본값은 ','입니다.

- "startingOffsets": (선택 사항) 데이터를 읽을 Kafka 주제의 시작 위치입니다. 가능한 값은 "earliest" 또는 "latest"입니다. 기본값은 "latest"입니다.
- "startingTimestamp": (선택 사항, AWS Glue 버전 4.0 이상에서만 지원됨) 데이터를 읽을 Kafka 주제에 있는 레코드의 타임스탬프입니다. 가능한 값은 yyyy-mm-ddTHH:MM:SSZ 패턴에서 UTC 형식의 타임스탬프 문자열입니다(여기서, Z는 UTC 시간대 오프셋(+/-)임, 예: '2023-04-04T08:00:00-04:00').

  참고: AWS Glue 스트리밍 스크립트의 연결 옵션 목록에는 'startingOffsets' 또는 'startingTimestamp' 중 하나만 존재할 수 있습니다. 이 두 속성을 모두 포함하면 작업에 실패합니다.
- "endingOffsets": (선택 사항) 일괄 쿼리가 종료되는 엔드포인트입니다. 가능한 값은 "latest" 또는 각 TopicPartition의 끝 오프셋을 지정하는 JSON 문자열입니다.

  JSON 문자열의 경우 포맷은 {"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}입니다. 오프셋으로 값 -1은 "latest"를 나타냅니다.
- "pollTimeoutMs": (선택 사항) Spark 작업 실행기의 Kafka에서 데이터를 폴링하는 시간 제한(밀리초)입니다. 기본값은 600000입니다.
- "numRetries": (선택 사항) Kafka 오프셋 가져오기에 실패하기 전에 재시도할 횟수입니다. 기본값은 3입니다.
- "retryIntervalMs": (선택 사항) Kafka 오프셋을 가져오기를 다시 시도하기 전에 대기하는 시간(밀리초)입니다. 기본값은 10입니다.
- "maxOffsetsPerTrigger": (선택 사항) 트리거 간격당 처리되는 최대 오프셋 수에 대한 속도 제한입니다. 지정된 총 오프셋 수는 서로 다른 볼륨의 topicPartitions에 비례하여 분할됩니다. 기본값은 null입니다. 즉, 소비자가 알려진 최신 오프셋까지 모든 오프셋을 읽습니다.
- "minPartitions": (선택 사항) Kafka에서 읽을 원하는 최소 파티션 수입니다. 기본값은 null이며 이는 Spark 파티션의 수가 Kafka 파티션의 수와 동일함을 의미합니다.
- "includeHeaders": (선택 사항) Kafka 헤더를 포함할지 여부입니다. 옵션이 "true"로 설정되면 데이터 출력에는 유형이 Array[Struct(key: String, value: String)]인 "glue_streaming_kafka_headers"라는 추가 열이 포함됩니다. 기본값은 "false"입니다. 이 옵션은 AWS Glue 버전 3.0 이상에서만 사용할 수 있습니다.
- "schema": (InferSchema가 false로 설정된 경우 필수) 페이로드를 처리하는 데 사용할 스키마입니다. 분류가 avro인 경우 제공된 스키마는 Avro 스키마 형식이어야 합니다. 분류가 avro가 아닌 경우 제공된 스키마는 DDL 스키마 형식이어야 합니다.

  다음은 스키마의 예입니다.

Example in DDL schema format

```
'column1' INT, 'column2' STRING , 'column3' FLOAT
```

Example in Avro schema format

```
{
"type":"array",
"items":
{
"type":"record",
"name":"test",
"fields":
[
  {
    "name":"_id",
    "type":"string"
  },
  {
    "name":"index",
    "type":
    [
      "int",
      "string",
      "float"
    ]
  }
]
}
}
```

- "inferSchema": (선택 사항) 기본값은 'false'입니다. 'true'로 설정하면 스키마가 런타임 시 foreachbatch 내의 페이로드에서 감지됩니다.
- "avroSchema": (더 이상 사용되지 않음) Avro 형식을 사용할 때 Avro 데이터의 스키마를 지정하는 데 사용되는 파라미터입니다. 이 파라미터는 이제 사용 중단되었습니다. schema 파라미터를 사용합니다.
- "addRecordTimestamp": (선택 사항) 이 옵션이 'true'로 설정되면 데이터 출력에는 이름이 '__src_timestamp'라는 추가 열이 포함됩니다. 이 열은 주제에서 해당 레코드를 수신한 시간을 나타냅니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.

- "emitConsumerLagMetrics": (선택 사항) 이 옵션을 'true'로 설정하면 각 배치에 대해 주제에서 수신한 가장 오래된 레코드와 AWS Glue에 도착한 시간 사이의 지표를 CloudWatch로 내보냅니다. 지표의 이름은 'glue.driver.streaming.maxConsumerLagInMs'입니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.

쓸 때는 "connectionType": "kafka"를 사용한 다음 연결 옵션을 사용합니다.

- "connectionName"(필수) Kafka 클러스터에 연결하는 데 사용되는 AWS Glue 연결의 이름(Kafka 소스와 유사).
- "topic"(필수) 주제 열이 존재하는 경우 주제 구성 옵션이 설정되지 않은 한 Kafka에 지정된 행을 쓸 때 해당 값이 주제로 사용됩니다. 즉, topic 구성 옵션이 주제 열을 재정의합니다.
- "partition"(선택 사항) 유효한 파티션 번호가 지정되면 레코드를 보낼 때 해당 partition이 사용됩니다.

  파티션을 지정하지 않았지만 key가 있는 경우 키의 해시를 사용하여 파티션이 선택됩니다.

  key와 partition이 모두 존재하지 않으면 파티션에 최소 batch.size 바이트가 생성될 때 변경 사항을 고정 파티셔닝하여 파티션이 선택됩니다.
- "key"(선택 사항) partition이 null인 경우 파티셔닝에 사용됩니다.
- "classification"(선택 사항) 레코드의 데이터에서 사용하는 파일 형식입니다. JSON, CSV, Avro만 지원합니다.

  Avro 형식을 사용하면 직렬화할 사용자 지정 avroSchema를 제공할 수 있지만 역직렬화를 위해서는 소스에서도 이를 제공해야 한다는 점에 유의하세요. 그렇지 않으면 기본적으로 직렬화를 위해 Apache AvroSchema를 사용합니다.

또한 필요에 따라 Kafka 프로듀서 구성 파라미터를 업데이트하여 Kafka 싱크를 미세 튜닝할 수 있습니다. 연결 옵션에는 허용 목록이 없으며 모든 키-값 쌍은 싱크에 그대로 유지됩니다.

그러나 적용되지 않는 일부 거부 옵션의 목록이 있습니다. 자세한 내용은 Kafka 특정 구성을 참조하세요.

Kinesis 연결 관련 작업

Data Catalog 테이블에 저장된 정보를 사용하거나 데이터 스트림에 직접 액세스할 수 있는 정보를 제공하여 Kinesis 연결을 통해 Amazon Kinesis 데이터 스트림에서 읽고 쓸 수 있습니다. Kinesis의 정보를 Spark DataFrame으로 읽은 다음 AWS Glue DynamicFrame으로 변환할 수 있습니다.

DynamicFrame을 JSON 형식으로 Kinesis에 쓸 수 있습니다. 데이터 스트림에 직접 액세스하는 경우 이러한 옵션을 사용하여 데이터 스트림에 액세스하는 방법에 대한 정보를 제공합니다.

`getCatalogSource` 또는 `create_data_frame_from_catalog`를 통해 Kinesis 스트리밍 소스의 레코드를 사용하는 경우 작업에 데이터 카탈로그 데이터베이스 및 테이블 이름 정보가 있으며, 해당 정보를 사용하여 Kinesis 스트리밍 소스에서 읽기 위한 몇 가지 기본 파라미터를 얻을 수 있습니다. `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` 또는 `create_data_frame_from_options`를 사용하는 경우 여기에 설명된 연결 옵션을 통해 이러한 기본 파라미터를 지정해야 합니다.

`GlueContext` 클래스에 지정된 메서드에 대해 다음 인수를 사용하여 Kinesis에 대한 연결 옵션을 지정할 수 있습니다.

- Scala
  - `connectionOptions`: `getSource`, `createDataFrameFromOptions`, `getSink`와(과) 함께 사용
  - `additionalOptions`: `getCatalogSource`, `getCatalogSink`와 함께 사용
  - `options`: `getSourceWithFormat`, `getSinkWithFormat`와 함께 사용
- Python
  - `connection_options`: `create_data_frame_from_options`, `write_dynamic_frame_from_options`와 함께 사용
  - `additional_options`: `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`와 함께 사용
  - `options`: `getSource`, `getSink`와 함께 사용

스트리밍 ETL 작업에 대한 참고 및 제한 사항은 the section called “스트리밍 ETL 참고 사항 및 제한 사항” 섹션을 참조하세요.

Kinesis 구성

AWS Glue Spark 작업에서 Kinesis 데이터 스트림을 연결하려면 몇 가지 필수 조건이 필요합니다.

- 읽는 경우 AWS Glue 작업에는 Kinesis 데이터 스트림에 대한 읽기 액세스 수준의 IAM 권한이 있어야 합니다.
- 쓰는 경우 AWS Glue 작업에는 Kinesis 데이터 스트림에 대한 쓰기 액세스 수준의 IAM 권한이 있어야 합니다.

경우에 따라 추가 필수 조건을 구성해야 합니다.

- AWS Glue 작업이 추가 네트워크 연결(일반적으로 다른 데이터 세트에 연결)로 구성되었고 이러한 연결 중 하나에서 Amazon VPC 네트워크 옵션을 제공하는 경우 작업에 Amazon VPC를 통해 통신하도록 지시합니다. 이 경우 Amazon VPC를 통해 통신하도록 Kinesis 데이터 스트림도 구성해야 합니다. Amazon VPC와 Kinesis 데이터 스트림 사이에서 인터페이스 VPC 엔드포인트를 생성하면 됩니다. 자세한 내용은 Using Kinesis Data Streams with Interface VPC Endpoints를 참조하세요.
- 다른 계정에서 Amazon Kinesis Data Streams를 지정할 때 크로스 계정 액세스를 허용하도록 역할과 정책을 설정해야 합니다. 자세한 내용은 예: 다른 계정의 Kinesis 스트림에서 읽기를 참조하세요.

스트리밍 ETL 작업 필수 조건에 대한 자세한 내용은 the section called “스트리밍 ETL 작업” 섹션을 참조하세요.

Kinesis에서 읽기

예제: Kinesis 스트림에서 읽기

the section called “forEachBatch”과(와) 함께 사용합니다.

Amazon Kinesis 스트리밍 소스의 예:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Kinesis에 쓰기

예: Kinesis 스트림에 쓰기

the section called “forEachBatch”과(와) 함께 사용합니다. DynamicFrame은 JSON 형식으로 스트림에 작성됩니다. 작업을 여러 번 시도해도 쓸 수 없다면 작업이 실패합니다. 기본적으로 각 DynamicFrame 레코드는 Kinesis 스트림으로 개별적으로 전송됩니다. `aggregationEnabled` 및 관련 파라미터를 사용하여 이 동작을 구성할 수 있습니다.

스트리밍 작업에서 Amazon Kinesis로 작성하는 예:

Python

```
glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite
    connection_type="kinesis",
    connection_options={
        "partitionKey": "part1",
        "streamARN": "arn:aws:kinesis:us-east-1:111122223333:stream/streamName",
    }
)
```

Scala

```
glueContext.getSinkWithFormat(
                connectionType="kinesis",
                options=JsonOptions("""{
                    "streamARN": "arn:aws:kinesis:us-
east-1:111122223333:stream/streamName",
                    "partitionKey": "part1"
                }"""),
            )
            .writeDynamicFrame(frameToWrite)
```

Kinesis 연결 파라미터

Amazon Kinesis Data Streams에 대한 연결 옵션을 지정합니다.

Kinesis 스트리밍 데이터 원본에 대한 다음 연결 옵션을 사용합니다.

- `"streamARN"` (필수) 읽기/쓰기에 사용됩니다. Kinesis 데이터 스트림의 ARN.
- `"classification"` (읽기 필수) 읽기에 사용됩니다. 레코드의 데이터에서 사용하는 파일 형식. 데이터 카탈로그를 통해 제공되지 않는 한 필수입니다.
- `"streamName"` – (선택 사항) 읽기에 사용됩니다. 읽을 Kinesis 데이터 스트림의 이름. `endpointUrl`와(과) 함께 사용됩니다.
- `"endpointUrl"` – (선택 사항) 읽기에 사용됩니다. 기본값: "https://kinesis.us-east-1.amazonaws.com". Kinesis 스트림의 AWS 엔드포인트. 특정 지역에 연결하는 경우가 아니면 변경하지 않아도 됩니다.

- "partitionKey" – (선택 사항) 쓰기에 사용됩니다. 레코드를 생성할 때 사용되는 Kinesis 파티션 키.
- "delimiter" (선택 사항) 읽기에 사용됩니다. classification이(가) CSV일 때 사용되는 값 구분 기호입니다. 기본값은 ','입니다.
- "startingPosition": (선택 사항) 읽기에 사용됩니다. 데이터를 읽을 Kinesis 데이터 스트림의 시작 위치입니다. 가능한 값은 yyyy-mm-ddTHH:MM:SSZ 패턴에서 UTC 형식의 타임스탬프 문자열이나 "latest", "trim_horizon" 또는 "earliest"입니다(여기서, Z는 UTC 시간대 오프셋(+/-)임, 예: '2023-04-04T08:00:00-04:00'). 기본값은 "latest"입니다. 참고: "startingPosition"에 대한 UTC 형식의 타임스탬프 문자열은 AWS Glue 버전 4.0 이상에서만 지원됩니다.
- "failOnDataLoss": (선택 사항) 활성 샤드가 누락되거나 만료된 경우 작업이 실패합니다. 기본값은 "false"입니다.
- "awsSTSRoleARN": (선택 사항) 읽기/쓰기에 사용됩니다. AWS Security Token Service(AWS STS)을(를) 사용하여 맡을 역할의 Amazon 리소스 이름(ARN). 이 역할에는 Kinesis 데이터 스트림에 대한 레코드 작업을 설명하거나 읽을 수 있는 권한이 있어야 합니다. 다른 계정의 데이터 스트림에 액세스할 때 이 파라미터를 사용해야 합니다. "awsSTSSessionName"과(와) 함께 사용합니다.
- "awsSTSSessionName": (선택 사항) 읽기/쓰기에 사용됩니다. AWS STS을(를) 사용하여 역할을 맡는 세션의 식별자입니다. 다른 계정의 데이터 스트림에 액세스할 때 이 파라미터를 사용해야 합니다. "awsSTSRoleARN"과(와) 함께 사용합니다.
- "awsSTSEndpoint": (선택 사항) 위임 받은 역할을 사용하여 Kinesis에 연결할 때 사용할 AWS STS 엔드포인트. 이를 통해 VPC에서 리전의 AWS STS 엔드포인트를 사용할 수 있지만, 기본 글로벌 엔드포인트로는 불가능합니다.
- "maxFetchTimeInMs": (선택 사항) 읽기에 사용됩니다. 작업 실행기가 Kinesis 데이터 스트림에서 현재 배치에 대한 레코드를 읽는 데 걸리는 최대 시간(밀리초(ms) 단위로 지정)입니다. 이 시간 내에 여러 개의 GetRecords API 호출을 할 수 있습니다. 기본값은 1000입니다.
- "maxFetchRecordsPerShard": (선택 사항) 읽기에 사용됩니다. 마이크로 배치에 따라 Kinesis 데이터 스트림에서 샤드당 가져올 최대 레코드 수입니다. 참고: 스트리밍 작업이 이미 Kinesis의 동일한 get-records 호출에서 추가 레코드를 읽은 경우 클라이언트가 이 제한을 초과할 수 있습니다. maxFetchRecordsPerShard가 엄격해야 한다면 maxRecordPerRead의 배수여야 합니다. 기본값은 100000입니다.
- "maxRecordPerRead": (선택 사항) 읽기에 사용됩니다. 각 getRecords 작업에서 Kinesis 데이터 스트림에서 가져올 최대 레코드 수. 기본값은 10000입니다.
- "addIdleTimeBetweenReads": (선택 사항) 읽기에 사용됩니다. 연속 두 getRecords 작업 사이에 시간 지연을 추가합니다. 기본값은 "False"입니다. 이 옵션은 Glue 버전 2.0 이상에서만 구성할 수 있습니다.

- `"idleTimeBetweenReadsInMs"`: (선택 사항) 읽기에 사용됩니다. 연속 두 `getRecords` 작업 사이의 최소 시간 지연으로, ms 단위로 지정됩니다. 기본값은 `1000`입니다. 이 옵션은 Glue 버전 2.0 이상에서만 구성할 수 있습니다.
- `"describeShardInterval"`: (선택 사항) 읽기에 사용됩니다. 스크립트가 리샤딩을 고려하기 위한 두 `ListShards` API 호출 사이의 최소 시간 간격. 자세한 내용은 Amazon Kinesis Data Streams Developer Guide의 Strategies for Resharding을 참조하세요. 기본값은 `1s`입니다.
- `"numRetries"`: (선택 사항) 읽기에 사용됩니다. Kinesis Data Streams API 요청의 최대 재시도 횟수입니다. 기본값은 `3`입니다.
- `"retryIntervalMs"`: (선택 사항) 읽기에 사용됩니다. Kinesis Data Streams API 호출을 재시도하기 전의 휴지 기간(ms 단위로 지정)입니다. 기본값은 `1000`입니다.
- `"maxRetryIntervalMs"`: (선택 사항) 읽기에 사용됩니다. Kinesis Data Streams API 호출을 두 번 재시도하는 사이의 최대 휴지 시간(ms 단위로 지정)입니다. 기본값은 `10000`입니다.
- `"avoidEmptyBatches"`: (선택 사항) 읽기에 사용됩니다. 배치가 시작되기 전에 Kinesis 데이터 스트림에서 읽지 않은 데이터를 확인하여 빈 마이크로 배치 작업 생성을 방지합니다. 기본값은 `"False"`입니다.
- `"schema"`: (inferSchema가 거짓으로 설정된 경우 필수 사항) 읽기에 사용됩니다. 페이로드를 처리하는 데 사용하는 스키마. 분류가 `avro`인 경우 제공된 스키마는 Avro 스키마 형식이어야 합니다. 분류가 `avro`가 아닌 경우 제공된 스키마는 DDL 스키마 형식이어야 합니다.

  다음은 스키마의 예입니다.

  Example in DDL schema format

  ```
  `column1` INT, `column2` STRING , `column3` FLOAT
  ```

  Example in Avro schema format

  ```
  {
    "type":"array",
    "items":
    {
      "type":"record",
      "name":"test",
      "fields":
      [
        {
          "name":"_id",
          "type":"string"
  ```

```
      },
      {
        "name":"index",
        "type":
        [
          "int",
          "string",
          "float"
        ]
      }
    ]
  }
}
```

- "inferSchema": (선택 사항) 읽기에 사용됩니다. 기본값은 'false'입니다. 'true'로 설정하면 스키마가 런타임 시 foreachbatch 내의 페이로드에서 감지됩니다.
- "avroSchema": (더 이상 사용되지 않음) 읽기에 사용됩니다. Avro 형식을 사용할 때 Avro 데이터의 스키마를 지정하는 데 사용되는 파라미터입니다. 이 파라미터는 이제 사용 중단되었습니다. schema 파라미터를 사용합니다.
- "addRecordTimestamp": (선택 사항) 읽기에 사용됩니다. 이 옵션이 'true'로 설정되면 데이터 출력에는 이름이 '__src_timestamp'라는 추가 열이 포함됩니다. 이 열은 스트림에서 해당 레코드를 수신한 시간을 나타냅니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.
- "emitConsumerLagMetrics": (선택 사항) 읽기에 사용됩니다. 옵션을 '참'으로 설정하면 각 배치에 대해 스트림에서 수신한 가장 오래된 레코드와 AWS Glue에 도착한 시간 사이의 지표를 CloudWatch로 내보냅니다. 지표의 이름은 'glue.driver.streaming.maxConsumerLagInMs'입니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.
- "fanoutConsumerARN": (선택 사항) 읽기에 사용됩니다. streamARN에서 지정된 스트림에 대한 Kinesis 스트림 소비자의 ARN. Kinesis 연결에 대해 향상된 팬아웃 모드를 활성화하는 데 사용됩니다. 향상된 팬아웃과 함께 Kinesis 스트림을 사용하는 방법에 대한 자세한 내용은 the section called “Kinesis 스트리밍 작업에서 향상된 팬아웃 사용” 섹션을 참조하세요.
- "recordMaxBufferedTime" – (선택 사항) 쓰기에 사용됩니다. 기본값: 1000(ms). 레코드 쓰기를 대기하는 중 레코드가 버퍼링되는 최대 시간.
- "aggregationEnabled" – (선택 사항) 쓰기에 사용됩니다. 기본값: true. Kinesis로 보내기 전에 레코드를 집계해야 하는지 여부를 지정합니다.

- `"aggregationMaxSize"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 51200(바이트). 레코드가 이 한도보다 크면 애그리게이터를 우회합니다. 참고 Kinesis는 레코드 크기를 50KB로 제한합니다. 이 값이 50KB를 초과하도록 설정하면 Kinesis에서 크기가 초과된 레코드를 거부합니다.
- `"aggregationMaxCount"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 4294967295. 집계된 레코드에 포함할 최대 항목 수.
- `"producerRateLimit"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 150(%). 단일 생성자(예: 작업)에서 보내는 샤드당 처리량을 백엔드 한도의 백분율로 제한합니다.
- `"collectionMaxCount"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 500. PutRecords 요청에 포함할 최대 항목 수.
- `"collectionMaxSize"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 5242880(바이트). PutRecords 요청으로 전송할 수 있는 데이터 최대량.

AWS Glue 스트리밍 자동 크기 조정

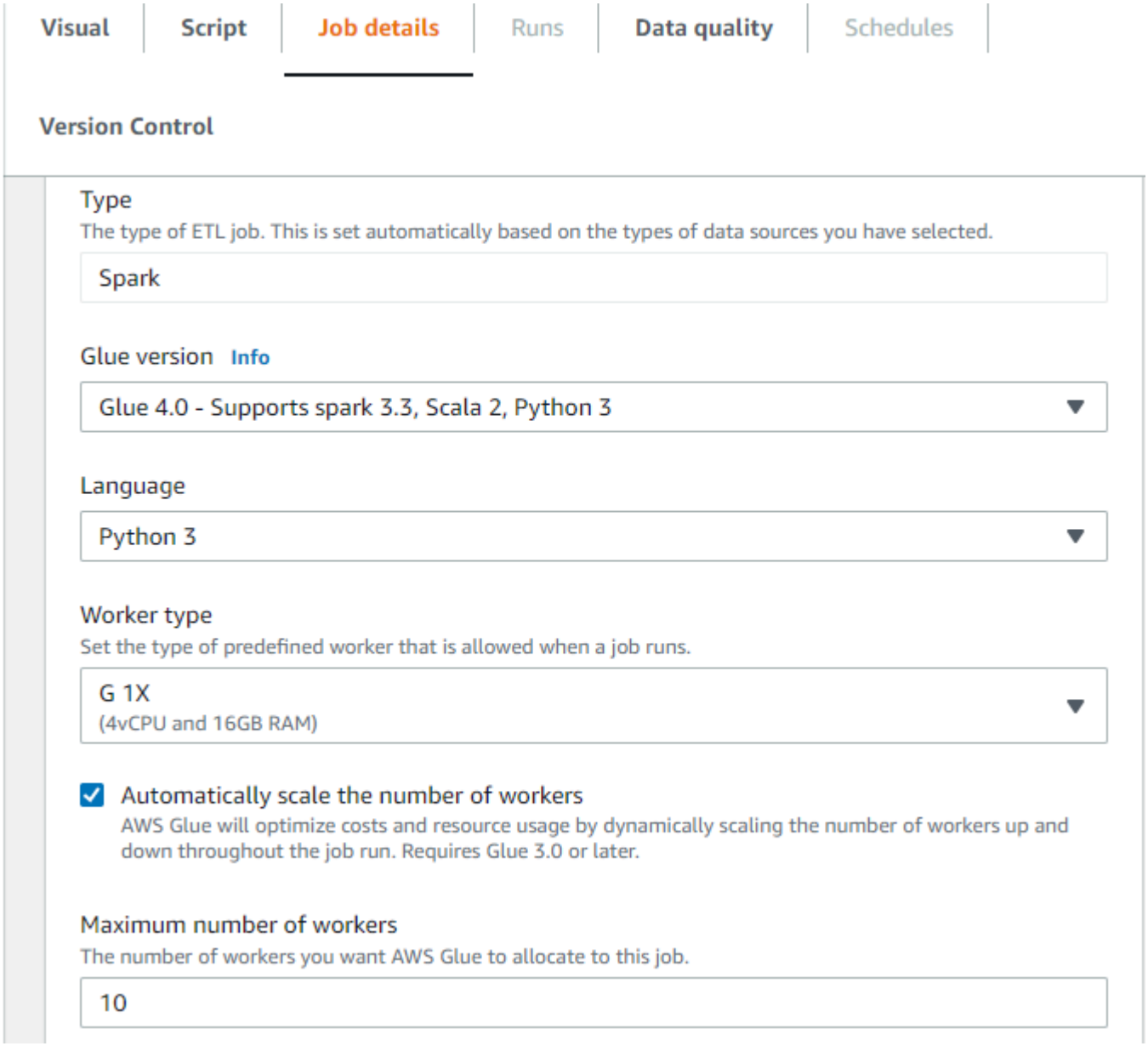
AWS Glue 스트리밍 ETL 작업은 스트리밍 소스의 데이터를 지속적으로 소비하고, 전송 중인 데이터를 정리 및 변환하여 분석에 사용할 수 있도록 합니다. AWS Glue 오토 스케일링은 작업 실행의 각 단계를 모니터링하여 유휴 상태일 때 작업자를 끄거나 추가 병렬 처리가 가능한 경우 작업자를 추가할 수 있습니다.

다음 섹션에서는 AWS Glue 스트리밍 자동 크기 조정에 대한 정보를 제공합니다.

AWS Glue Studio에서 Auto Scaling 사용

AWS Glue Studio의 Job details(작업 세부 정보) 탭에서 유형을 Spark 또는 Spark Streaming으로, Glue version(Glue 버전)을 **`Glue 3.0`** 또는 **`Glue 4.0`**으로 선택합니다. 그러면 작업자 유형(Worker type) 아래에 확인란이 표시됩니다.

- 작업자 수 자동 크기 조정(Automatically scale the number of workers) 옵션을 선택합니다.
- 최대 작업자 수(Maximum number of workers)를 설정하여 작업 실행에 판매할 수 있는 최대 작업자 수를 정의합니다.



AWS CLI 또는 SDK를 사용하여 Auto Scaling 사용 설정

AWS CLI에서 Auto Scaling을 사용 설정하여 작업을 실행하려면, 다음 구성으로 `start-job-run`을 실행합니다.

```
{
    "JobName": "<your job name>",
    "Arguments": {
        "--enable-auto-scaling": "true"
    },
    "WorkerType": "G.2X", // G.1X and G.2X are allowed for Auto Scaling Jobs
```

```
    "NumberOfWorkers": 20, // represents Maximum number of workers
    ...other job run configurations...
}
```

ETL 작업 실행이 완료되면 get-job-run을 호출하여 DPU 초 단위로 작업 실행의 실제 리소스 사용량을 확인할 수도 있습니다. 참고: 새 필드 DPUSeconds는 Auto Scaling이 사용 설정된 AWS Glue 3.0 이상의 배치 작업에만 표시됩니다. 이 필드는 스트리밍 작업에는 지원되지 않습니다.

```
$ aws glue get-job-run --job-name your-job-name --run-id jr_xx --endpoint https://
glue.us-east-1.amazonaws.com --region us-east-1
{
    "JobRun": {
        ...
        "GlueVersion": "3.0",
        "DPUSeconds": 386.0
    }
}
```

동일한 구성의 AWS Glue SDK를 사용하여 Auto Scaling으로 작업 실행을 구성할 수도 있습니다.

작동 방식

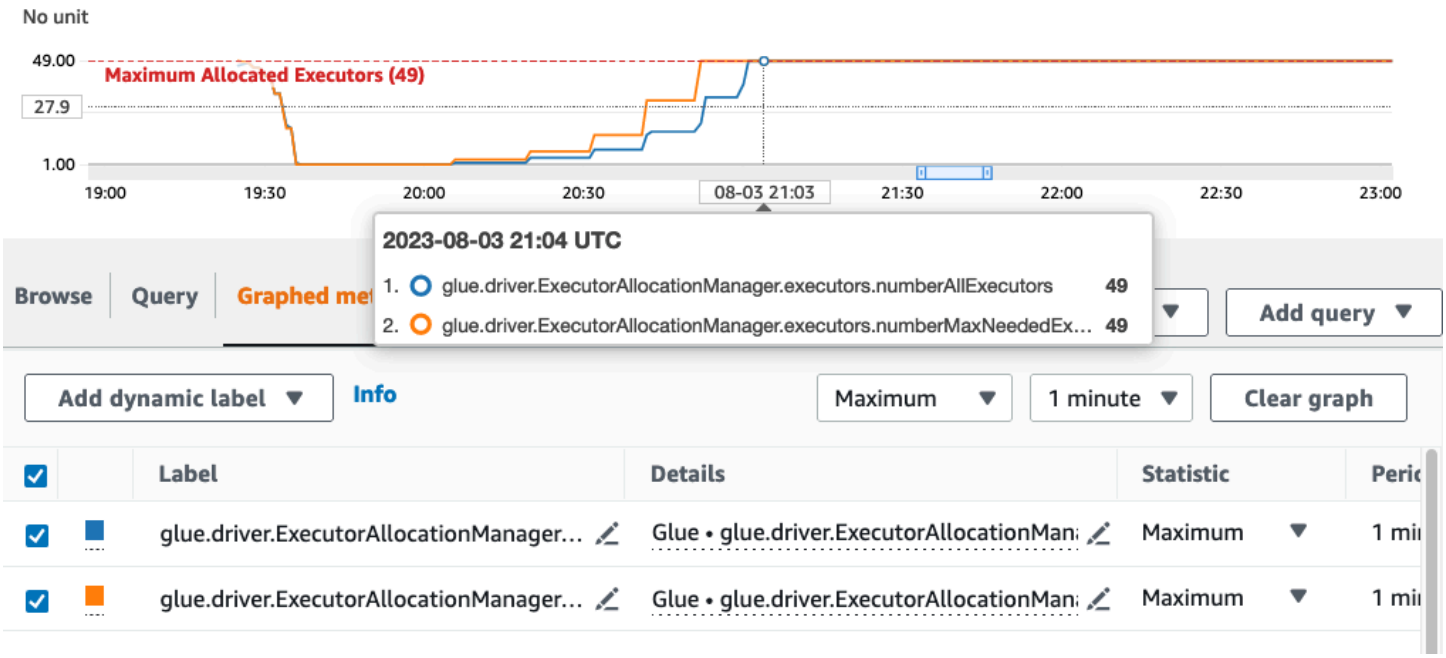
마이크로 배치 전반에 걸쳐 크기 조정

다음 예제는 자동 크기 조정의 작동 방식을 설명하는 데 사용됩니다.

- 50개의 DPU로 시작하는 AWS Glue 작업이 있습니다.
- 자동 크기 조정이 활성화되었습니다.

이 예제에서 AWS Glue는 몇 개의 마이크로 배치에 대한 'batchProcessingTimeInMs' 지표를 살펴보고 사용자가 설정한 기간 내에서 작업이 완료되는지 확인합니다. 작업이 더 빨리 완료되는 경우 작업이 얼마나 빨리 완료되는지에 따라 AWS Glue가 스케일 다운될 수 있습니다. 'numberAllExecutors'로 그린 이 지표를 Amazon CloudWatch에서 모니터링하여 자동 크기 조정이 어떻게 작동하는지 확인할 수 있습니다.

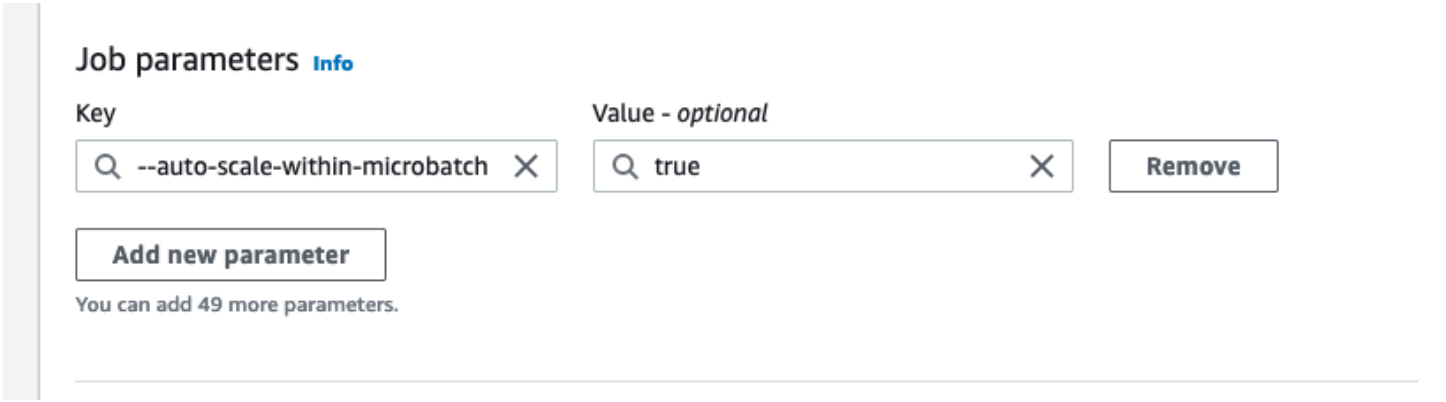
실행기 수는 각 마이크로 배치가 완료된 후에만 기하급수적으로 스케일 업 또는 다운됩니다. Amazon CloudWatch 모니터링 로그에서 볼 수 있듯이, AWS Glue는 필요한 실행기 수(주황색 선)를 살펴보고 그에 맞춰 자동으로 실행기 크기(파란색 선)를 조정합니다.



AWS Glue가 실행기 수를 스케일 다운하고 데이터 볼륨 증가에 따른 마이크로 배치 처리 시간 증가가 관찰되면 AWS Glue는 지정된 상한인 50 DPU까지 스케일 업합니다.

마이크로 배치 내에서 크기 조정

위 예제에서 시스템은 완료된 마이크로 배치 몇 개를 모니터링하여 스케일 업할지 스케일 다운할지 결정합니다. 기간이 길어질수록 몇 개의 마이크로 배치를 기다리는 대신 마이크로 배치 내에서 더 빠르게 응답하기 위해 Auto Scaling이 필요합니다. 이러한 경우 추가 구성 `--auto-scaIe-within-microbatch`를 `true`로 사용할 수 있습니다. 아래와 같이 AWS Glue Studio의 AWS Glue 작업 속성에 이를 추가할 수 있습니다.



AWS Glue 스트리밍을 위한 유지 관리 기간

AWS Glue에서는 정기적으로 유지 관리 활동을 수행합니다. 이러한 유지 관리 기간 동안에는 AWS Glue에서 스트리밍 작업을 다시 시작해야 합니다. 유지 관리 기간을 지정하여 작업이 다시 시작되는 시기를 제어할 수 있습니다. 이 섹션에서는 유지 관리 기간과 고려해야 할 특정 동작을 설정할 수 있는 위치를 간략하게 설명합니다.

주제

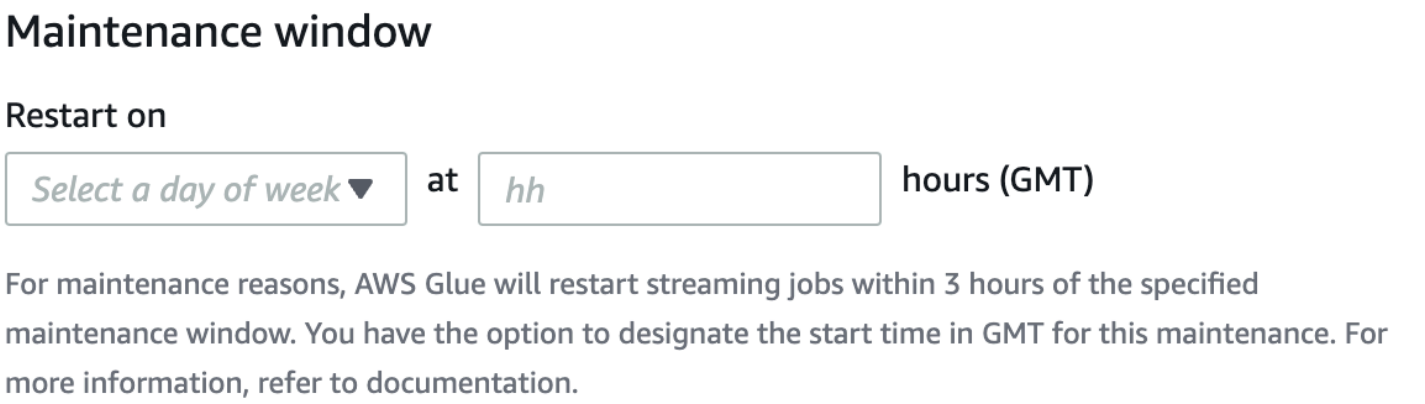
- 유지 관리 기간 설정
- 유지 관리 기간 동작
- 작업 모니터링
- 데이터 손실 처리

유지 관리 기간 설정

AWS Glue Studio 또는 API를 사용하여 유지 관리 기간을 설정할 수 있습니다.

AWS Glue Studio에서 유지 관리 기간 설정

AWS Glue 스트리밍 작업의 작업 세부 정보 페이지에서 유지 관리 기간을 지정할 수 있습니다. GMT로 날짜 및 시간을 지정할 수 있습니다. AWS Glue는 지정된 기간에 작업을 다시 시작합니다.



API에서 유지 관리 기간 설정

또는 작업 생성 API에서 유지 관리 기간을 설정할 수 있습니다. 다음은 API를 통해 유지 관리 기간을 구성하는 예제입니다.

```
aws glue create-job –name jobName –role roleArnForTheJob –command
 Name=gluestreaming,ScriptLocation=s3-path-to-the-script --maintenance-window="Sun:10"
```

다음은 예제 명령입니다.

```
aws glue create-job –name testMaintenance –role arn:aws:iam::012345678901:role/
Glue_DefaultRole –command Name=gluestreaming,ScriptLocation=s3://glue-example-test/
example.py –maintenance-window="Sun:10
```

유지 관리 기간 동작

AWS Glue는 작업을 다시 시작할 시기를 결정하기 위해 일련의 단계를 수행합니다.

1. 새 스트리밍 작업이 시작되면 AWS Glue는 먼저 작업 실행과 관련된 제한 시간이 있는지 확인합니다. 제한 시간을 통해 작업 종료 시간을 구성할 수 있습니다. 제한 시간이 7일 미만인 경우 작업이 다시 시작되지 않습니다.
2. 제한 시간이 7일을 초과하면 AWS Glue는 작업에 대해 유지 관리 기간이 구성되어 있는지 확인합니다. 그러면 해당 기간이 선택되고 해당 기간이 작업 실행에 할당됩니다. AWS Glue는 지정된 유지 관리 기간으로부터 3시간 이내에 작업을 다시 시작합니다. 예를 들어, 유지 관리 기간을 GMT 기준 월요일 오전 10시로 설정하면 작업이 GMT 기준 오전 10시에서 오후 1시 사이에 다시 시작됩니다.
3. 유지 관리 기간이 구성되지 않은 경우 AWS Glue는 다시 시작 시간을 작업 실행 시작 시간으로부터 7일 후로 자동 설정합니다. 예를 들어, 2024년 7월 1일 오전 12:00 GMT에 작업을 시작한 후 유지 관리 기간을 지정하지 않은 경우 2024년 7월 8일 오전 12:00 GMT에 작업이 다시 시작되도록 설정됩니다.

   > **Note**
   >
   > 스트리밍 작업을 이미 실행 중인 경우 이 변경 사항은 2024년 7월 1일부터 적용됩니다. 6월 30일까지 유지 관리 기간을 구성할 수 있습니다. 7월 1일 이후에는 이 설명서에 따라 시작하는 모든 스트리밍 작업이 다시 시작됩니다. 추가 지원이 필요한 경우 AWS Support에 문의할 수 있습니다.

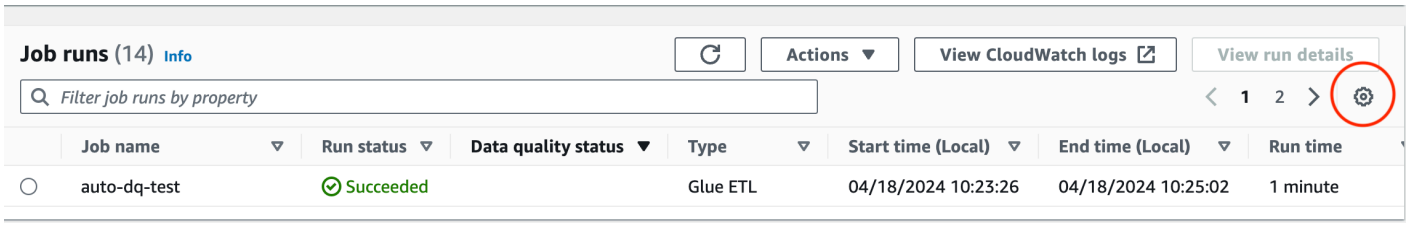
4. 때때로, 특히 진행 중인 마이크로 배치가 처리되지 않을 때 AWS Glue에서 작업을 다시 시작하지 못할 수 있습니다. 이 경우 작업이 중단되지 않습니다. 이 경우 AWS Glue는 14일 후에 작업을 다시 시작하며, 이때 유지 관리 기간은 적용되지 않습니다.

작업 모니터링

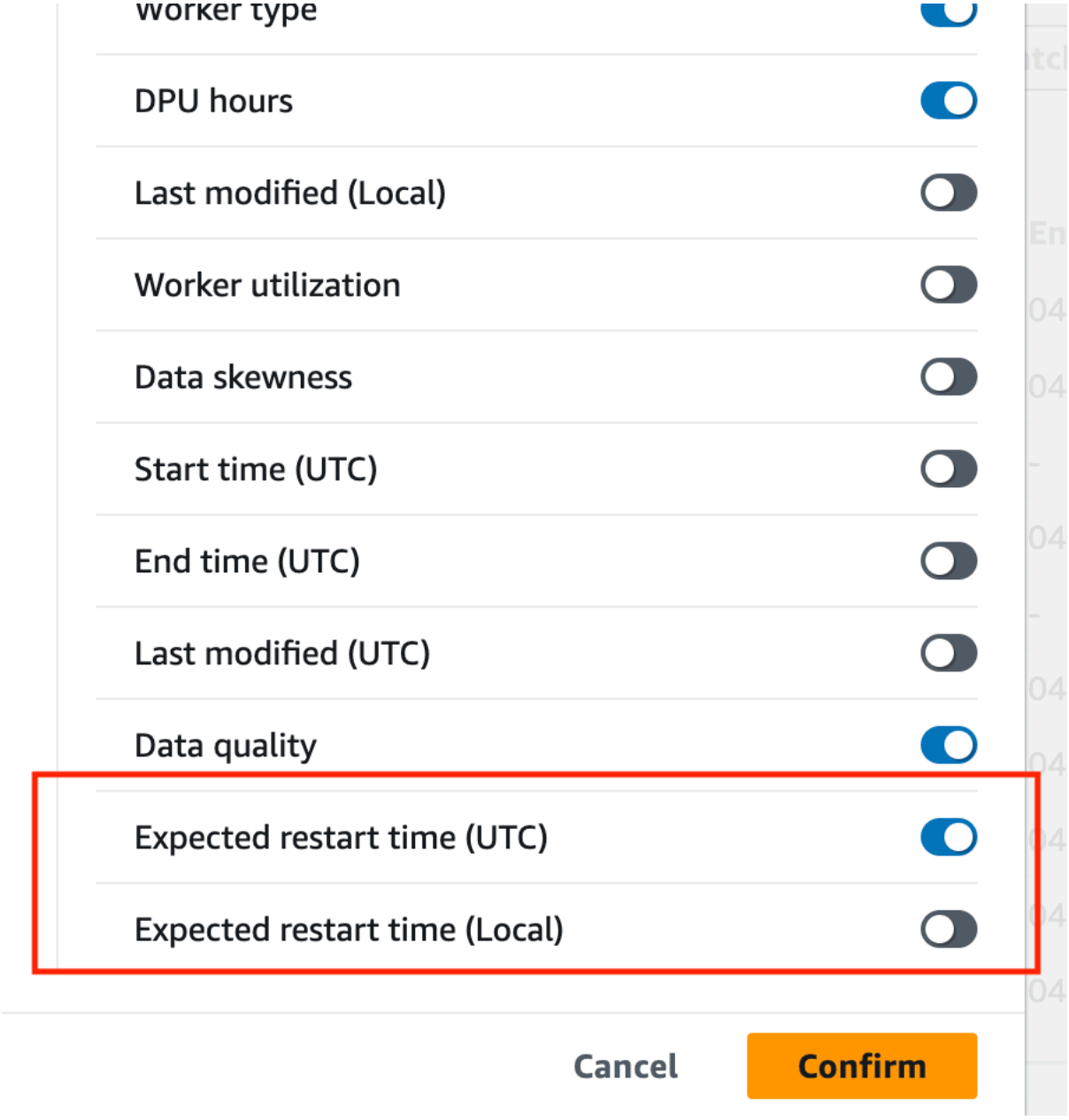
AWS Glue Studio 모니터링 페이지에서 작업을 모니터링할 수 있습니다.

스트리밍 작업의 예상 다음 다시 시작 시간을 보려면 모니터링 페이지의 작업 실행 테이블에 열을 표시합니다.

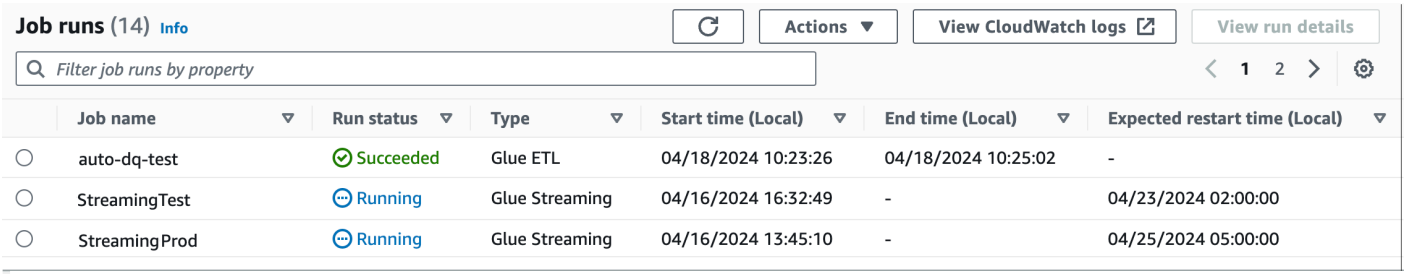
1. 테이블 오른쪽 상단에서 톱니 아이콘을 클릭합니다.



2. 아래로 스크롤하여 예상 다시 시작 시간 열을 켭니다. UTC 및 로컬 시간 옵션을 모두 사용할 수 있습니다.



3. 그런 다음, 테이블의 열을 볼 수 있습니다.



원래 작업의 상태는 '만료됨'이고 새 작업 인스턴스의 상태는 '실행 중'입니다. 다시 시작된 새 작업 실행의 작업 실행 ID는 초기 작업 실행 ID와 다시 시작 횟수를 나타내는 접두사 'restart_'의 연결로 구성됩니다. 예를 들어 초기 작업 실행 ID가 `jr_1234`인 경우 다시 시작된 작업 실행의 첫 번째 다시 시작 시 ID는 `jr1234_restart_1`입니다. 두 번째 다시 시작 시 `jr1234_restart_2`와 같은 식입니다.

다시 시작이므로 재시도는 영향을 받지 않습니다. 자동 재시도로 인해 실행이 실패하고 새 실행이 시작되면 다시 시작 카운터는 1부터 다시 시작됩니다. 예를 들어 `jr_1234_attempt_3_restart_5`에서 실행이 실패하면 자동 재시도 시 `jr_id1_attempt_4`의 ID로 새 실행이 시작되고 7일 후에 이 시도가 다시 시작되면 새 실행 ID는 `jr_id1_attempt_4_restart_1`입니다.

데이터 손실 처리

유지 관리가 다시 시작되는 동안 AWS Glue 스트리밍은 이전 작업 실행과 다시 시작된 작업 실행 간 데이터 무결성과 일관성을 보장하는 프로세스를 따릅니다. AWS Glue는 작업 다시 시작 사이에서 데이터 무결성과 일관성을 보장하지 않으며, 스트리밍 작업 내에서 중복된 데이터를 처리하기 위해 아키텍처 고려 사항을 사용하는 것이 좋습니다.

1. 유지 관리 다시 시작 조건 감지: AWS Glue 스트리밍은 7일 후 유지 관리 기간에 도달하거나 14일 후 하드 다시 시작이 필요한 경우와 같이 유지 관리 다시 시작을 트리거해야 하는 시기를 나타내는 조건을 모니터링합니다.
2. 정상적인 종료 간접 호출: 유지 관리 다시 시작 조건이 충족되면 AWS Glue 스트리밍은 현재 실행 중인 작업에 대해 정상적인 종료 프로세스를 시작합니다. 이 프로세스는 다음 단계를 포함합니다.
   a. 새 데이터 수집 중지: 스트리밍 작업은 입력 소스(예: Kafka 주제, Kinesis 스트림 또는 파일)에서 새 데이터 소비를 중지합니다.
   b. 보류 중인 데이터 처리: 작업은 내부 버퍼 또는 대기열에 이미 있는 모든 데이터를 계속 처리합니다.
   c. 오프셋 및 체크포인트 커밋: 작업이 외부 시스템(예: Kafka, Kinesis 또는 Amazon S3)에 최신 오프셋 또는 체크포인트를 커밋하여 다시 시작된 작업이 이전 작업이 중단된 위치에서 선택할 수 있도록 합니다.
3. 작업 다시 시작: 정상적인 종료 프로세스가 완료되면 AWS Glue 스트리밍은 보존된 상태 및 체크포인트를 사용하여 작업을 다시 시작합니다. 다시 시작된 작업은 마지막으로 커밋된 오프셋 또는 체크포인트에서 처리를 선택하여 데이터가 손실되거나 복제되지 않도록 합니다.
4. 데이터 처리 재개: 다시 시작된 작업은 이전 작업이 중단된 시점부터 데이터 처리를 재개합니다. 마지막으로 커밋된 오프셋 또는 체크포인트부터 시작하여 입력 소스에서 새 데이터를 계속 수집하며 정의된 ETL 로직에 따라 데이터를 처리합니다.

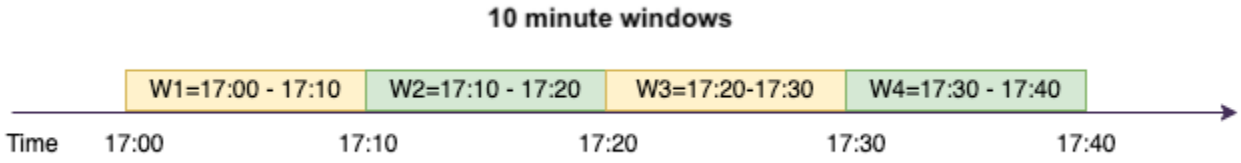
고급 AWS Glue 스트리밍 개념

현대의 데이터 기반 애플리케이션에서 데이터의 중요성은 시간 경과에 따라 감소하고 그 가치는 예측에서 반응으로 전환됩니다. 따라서 고객은 더 빠른 의사 결정을 내리기 위해 실시간으로 데이터를 처리하기를 원합니다. IoT 센서와 같은 실시간 데이터 피드를 처리할 때 데이터가 정렬되지 않은 상태로 도착하거나 네트워크 지연 시간과 수집 중 기타 소스 관련 오류로 인해 처리가 지연될 수 있습니다. AWS Glue 플랫폼의 일부인 AWS Glue 스트리밍은 이러한 기능을 기반으로 Apache Spark 정형 스트리밍으로 구동되는 확장 가능한 서버리스 스트리밍 ETL을 제공하여 사용자에게 실시간 데이터 처리 기능을 제공합니다.

이 주제에서는 AWS Glue 스트리밍의 고급 스트리밍 개념과 기능을 살펴보겠습니다.

스트림 처리 시 시간 고려 사항

스트림을 처리할 때 시간에 대한 네 가지 개념이 있습니다.



- 이벤트 타임 – 이벤트가 발생한 시간입니다. 대부분의 경우 이 필드는 소스의 이벤트 데이터 자체에 포함됩니다.
- 이벤트 기간 - 두 이벤트 시간 사이의 기간입니다. 위 다이어그램에 표시된 것처럼 W1은 17:00부터 17:10까지의 이벤트 기간입니다. 각 이벤트 기간은 여러 이벤트의 그룹입니다.
- 트리거 시간 - 트리거 시간은 데이터 처리 및 결과 업데이트 빈도를 제어합니다. 마이크로 배치 처리가 시작된 시간입니다.
- 모으기 시간 - 스트림 데이터가 스트리밍 서비스에 수집된 시간입니다. 이벤트 시간이 이벤트 자체에 포함되지 않은 경우 경우에 따라 이 시간을 기간 설정에 사용할 수 있습니다.

기간 설정

기간 설정은 이벤트 기간을 기준으로 여러 이벤트를 그룹화하고 집계하는 기법입니다. 다음 예제에서는 기간 설정의 이점과 이를 사용하는 경우를 살펴보겠습니다.

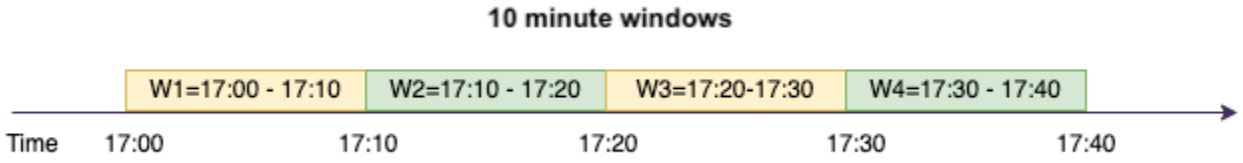
비즈니스 사용 사례에 따라 Spark에서 지원하는 세 가지 유형의 기간이 있습니다.

- 연속 기간 - 집계하는 일련의 겹치지 않는 고정된 크기의 이벤트 기간입니다.

- 슬라이딩 기간 – '고정된 크기'라는 점에서 연속 기간과 유사하지만, 슬라이드 지속 시간이 기간 자체의 지속 시간보다 짧으면 기간이 겹치거나 지나갈 수 있습니다.
- 세션 기간 - 입력 데이터 이벤트로 시작하여 공백 또는 비활성 기간 내에 입력을 수신하는 한 자체적으로 계속 확장됩니다. 세션 기간은 입력에 따라 기간 길이의 크기가 정적이거나 동적일 수 있습니다.

텀블링 윈도우

연속 기간은 집계하는 일련의 겹치지 않는 고정된 크기의 이벤트 기간입니다. 실제 사례를 들어 설명해 보겠습니다.



ABC Auto는 새로운 브랜드의 스포츠카에 대한 마케팅 캠페인을 진행하고자 합니다. 이를 위해 스포츠카 팬이 가장 많은 도시를 선택하려고 합니다. 이 목표를 달성하기 위해 이 회사는 웹 사이트에 자동차를 소개하는 15초짜리 짧은 광고를 선보입니다. 모든 '클릭'과 해당 '도시'가 기록되어 Amazon Kinesis Data Streams로 스트리밍됩니다. 10분 동안의 클릭 수를 세고 이를 도시별로 그룹화하여 수요가 가장 높은 도시를 확인하려고 합니다. 다음은 집계의 출력입니다.

window_start_time	window_end_time	구/군/시	total_clicks
2023-07-10 17:00:00	2023-07-10 17:10:00	댈러스	75
2023-07-10 17:00:00	2023-07-10 17:10:00	시카고	10
2023-07-10 17:20:00	2023-07-10 17:30:00	댈러스	20
2023-07-10 17:20:00	2023-07-10 17:30:00	시카고	50

위에서 설명한 것처럼 이러한 이벤트 기간은 트리거 시간 간격과 다릅니다. 예를 들어, 트리거 시간이 1분마다인 경우에도 출력 결과에는 겹치지 않는 집계 기간이 10분만 표시됩니다. 최적화를 위해서는 트리거 간격을 이벤트 기간에 맞추는 것이 좋습니다.

위 표에서 17:00~17:10 기간 동안 Dallas는 75번의 클릭이 있었고 Chicago는 10번의 클릭이 있었습니다. 또한 17:10~17:20 기간은 어느 도시의 데이터도 없으므로 생략됩니다.

이제 다운스트림 분석 애플리케이션에서 이 데이터에 대한 추가 분석을 실행하여 마케팅 캠페인을 실행할 가장 독점적인 도시를 결정할 수 있습니다.

AWS Glue에서 연속 기간 사용

1. Amazon Kinesis Data Streams DataFrame을 생성하고 여기에서 읽습니다. 예제:

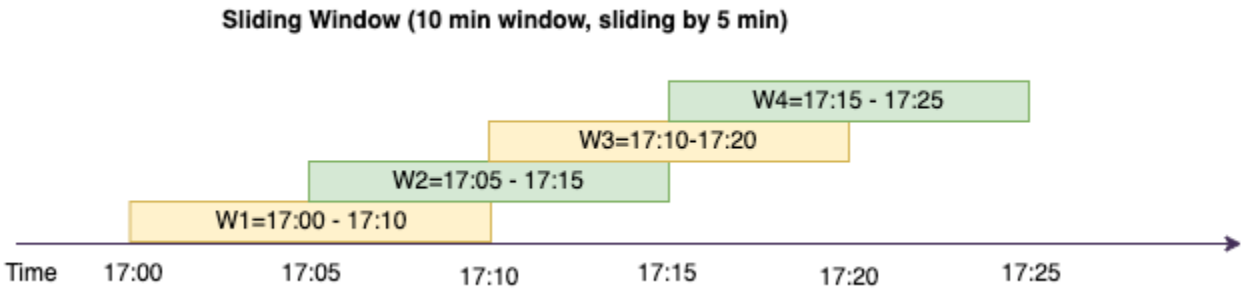
```
parsed_df = kinesis_raw_df \
                .selectExpr('CAST(data AS STRING)') \
                .select(from_json("data", ticker_schema).alias("data")) \
                .select('data.event_time','data.ticker','data.trade','data.volume',
 'data.price')
```

2. 연속 기간에서 데이터를 처리합니다. 아래 예제에서 데이터는 10분 연속 기간의 입력 필드 'event_time'을 기준으로 그룹화되고 출력을 Amazon S3 데이터 레이크에 씁니다.

```
grouped_df = parsed_df \
                .groupBy(window("event_time", "10 minutes"), "city") \
                .agg(sum("clicks").alias("total_clicks"))

summary_df = grouped_df \
                .withColumn("window_start_time", col("window.start")) \
                .withColumn("window_end_time", col("window.end")) \
                .withColumn("year", year("window_start_time")) \
                .withColumn("month", month("window_start_time")) \
                .withColumn("day", dayofmonth("window_start_time")) \
                .withColumn("hour", hour("window_start_time")) \
                .withColumn("minute", minute("window_start_time")) \
                .drop("window")

write_result = summary_df \
                .writeStream \
                .format("parquet") \
                .trigger(processingTime="10 seconds") \
                .option("checkpointLocation", "s3a://bucket-stock-stream/stock-
stream-catalog-job/checkpoint/") \
                .option("path", "s3a://bucket-stock-stream/stock-stream-catalog-
job/summary_output/") \
                .partitionBy("year", "month", "day") \
                .start()
```

슬라이딩 윈도우

슬라이딩 기간은 '고정된 크기'라는 점에서 연속 기간과 유사하지만, 슬라이드 지속 시간이 기간 자체의 지속 시간보다 짧으면 기간이 겹치거나 지나갈 수 있습니다. 슬라이딩의 특성으로 인해 입력이 여러 기간에 바인딩될 수 있습니다.



이해를 돕기 위해 신용 카드 사기 가능성을 탐지하려는 은행의 예를 들어 보겠습니다. 스트리밍 애플리케이션은 신용 카드 거래의 지속적인 스트림을 모니터링할 수 있습니다. 이러한 트랜잭션은 10분 기간으로 집계될 수 있으며 5분마다 기간이 앞으로 이동하여 가장 오래된 5분의 데이터를 제거하고 최신 5분의 새 데이터를 추가합니다. 각 기간 내에서 미국에서의 트랜잭션 직후 호주에서의 트랜잭션과 같은 의심스러운 패턴을 확인하면서 국가별로 트랜잭션을 그룹화할 수 있습니다. 간소화를 위해 총 트랜잭션 금액이 100 USD를 초과하는 경우 이러한 트랜잭션을 사기로 분류하겠습니다. 이러한 패턴이 탐지되면 사기 가능성이 있다는 신호이며 카드가 동결될 수 있습니다.

신용 카드 처리 시스템은 국가와 함께 각 카드 ID에 대한 일련의 거래 이벤트를 Kinesis로 전송하고 있습니다. AWS Glue 작업은 분석을 실행하고 다음과 같은 집계 결과를 생성합니다.

window_start_time	window_end_time	card_last_four	country	total_amount
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	미국	85
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	호주	10
2023-07-10 17:05:45	2023-07-10 17:15:45	6544	미국	50
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	미국	50

window_start_time	window_end_time	card_last_four	country	total_amount
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	호주	150

위 집계를 기준으로 5분마다 이동하는 10분 기간을 트랜잭션 금액별로 합산하여 볼 수 있습니다. 이상치(호주에서 150 USD 트랜잭션)가 있는 17:10~17:20 기간에서 이상이 탐지되었습니다. AWS Glue는 이러한 이상을 탐지하고 boto3를 사용하여 잘못된 키가 포함된 경보 이벤트를 SNS 주제에 푸시할 수 있습니다. 또한 Lambda 함수가 이 주제를 구독하고 조치를 취할 수 있습니다.

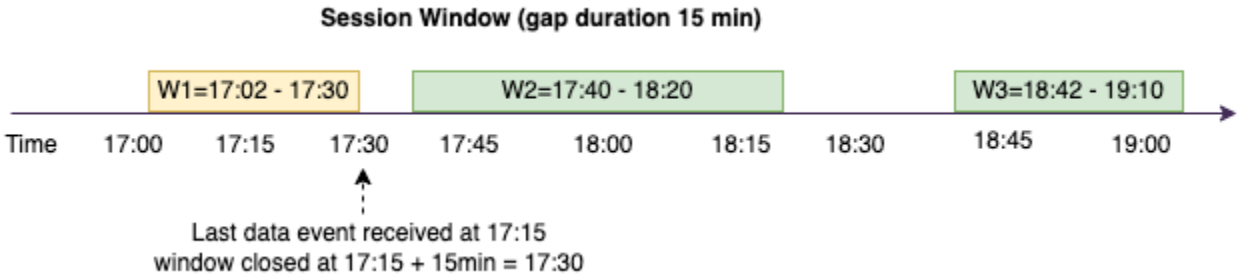
연속 기간에서 데이터 처리

`group-by` 절과 기간 함수는 아래와 같이 슬라이딩 기간을 구현하는 데 사용됩니다.

```
grouped_df = parsed_df \
            .groupBy(window(col("event_time"), "10 minute", "5 min"), "country",
 "card_last_four") \
            .agg(sum("tx_amount").alias("total_amount"))
```

세션 기간

크기가 고정된 위의 두 기간과 달리 세션 기간은 입력에 따라 기간 길이의 크기가 정적이거나 동적일 수 있습니다. 세션 기간은 입력 데이터 이벤트로 시작하여 공백 또는 비활성 기간 내에 입력을 수신하는 한 자체적으로 계속 확장됩니다.



예를 들어 보겠습니다. ABC 호텔은 일주일 중 가장 바쁜 시간이 언제인지 알아보고 고객에게 더 나은 가격을 제공하고자 합니다. 손님이 체크인하는 즉시 세션 기간이 시작되고 Spark는 해당 이벤트 기간에 대한 집계로 상태를 유지합니다. 손님이 체크인할 때마다 이벤트가 생성되어 Amazon Kinesis Data

Streams로 전송됩니다. 호텔은 15분 동안 체크인이 없을 경우 이벤트 기간을 닫을 수 있다고 판단합니다. 새 체크인이 있을 때 다음 이벤트 기간이 다시 시작됩니다. 출력은 다음과 같습니다.

window_start_time	window_end_time	구/군/시	total_checkins
2023-07-10 17:02:00	2023-07-10 17:30:00	댈러스	50
2023-07-10 17:02:00	2023-07-10 17:30:00	시카고	25
2023-07-10 17:40:00	2023-07-10 18:20:00	댈러스	75
2023-07-10 18:50:45	2023-07-10 19:15:45	댈러스	20

첫 번째 체크인은 event_time=17:02에 발생했습니다. 집계 이벤트 기간은 17:02에 시작됩니다. 이 집계는 15분 이내에 이벤트를 수신하는 한 계속됩니다. 위 예제에서 수신한 마지막 이벤트는 17:15였고 이후 15분 동안 이벤트가 없었습니다. 그 결과 Spark는 17:15+15min = 17:30에 이벤트 기간을 닫고 17:02~17:30으로 설정했습니다. 새 체크인 데이터 이벤트를 수신한 17:47에 새 이벤트 기간을 시작했습니다.

세션 기간에서 데이터 처리

`group-by` 절과 기간 함수는 슬라이딩 기간을 구현하는 데 사용됩니다.

```
grouped_df = parsed_df \
            .groupBy(session_window(col("event_time"), "10 minute"), "city") \
            .agg(count("check_in").alias("total_checkins"))
```

출력 모드

출력 모드는 무제한 테이블의 결과를 외부 싱크에 쓰는 모드입니다. 세 가지 모드를 사용할 수 있습니다. 다음 예제에서는 각 마이크로 배치에서 데이터 라인이 스트리밍되고 처리될 때 단어 발생 횟수를 계산합니다.

- 전체 모드 - 현재 이벤트 기간에서 단어 수가 업데이트되지 않았더라도 모든 마이크로 배치 처리가 끝나면 전체 결과 테이블을 싱크에 씁니다.
- 추가 모드 - 이 모드는 기본 모드로, 마지막 트리거 이후 결과 테이블에 추가된 새 단어 및/또는 행만 싱크에 씁니다. 이 모드는 맵, flatMap, 필터와 같은 쿼리의 상태 비저장 스트리밍에 적합합니다.

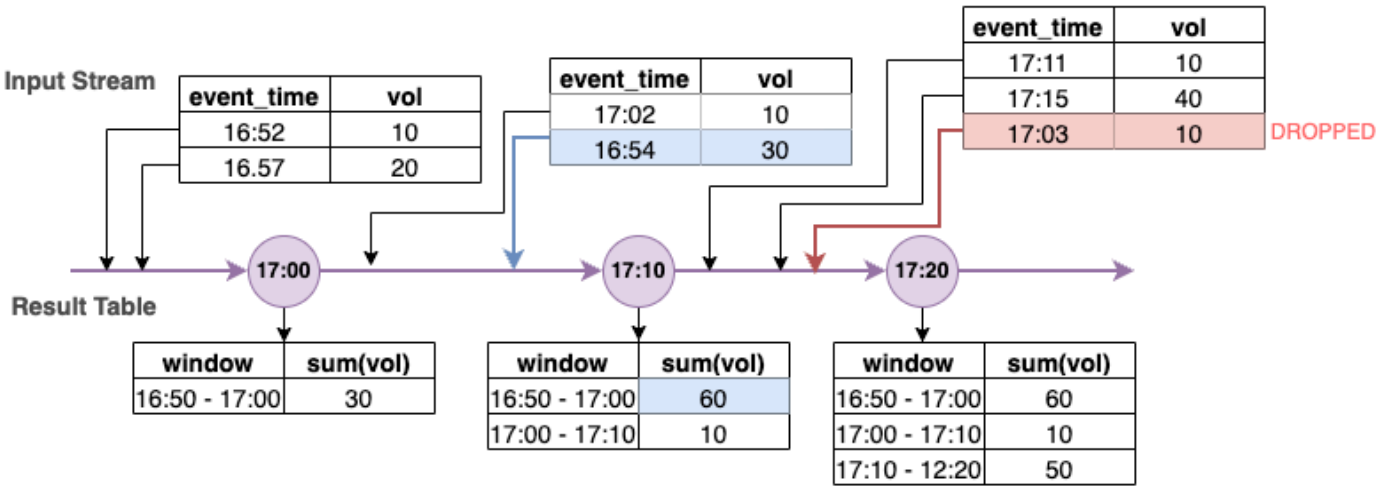
- 업데이트 모드 - 마지막 트리거 이후 업데이트되거나 추가된 결과 테이블의 단어 및/또는 행만 싱크에 씁니다.

> **Note**
>
> 출력 모드 = '업데이트'는 세션 기간에서 지원되지 않습니다.

최신 데이터 및 워터마크 처리

실시간 데이터로 작업할 때 네트워크 지연 및 업스트림 오류로 인해 데이터 도착이 지연될 수 있으며 놓친 이벤트 기간에 다시 집계를 수행하는 메커니즘이 필요합니다. 하지만 이렇게 하려면 상태를 유지해야 합니다. 이와 동시에 상태 크기를 제한하려면 오래된 데이터를 정리해야 합니다. Spark 버전 2.1에는 상태를 유지하고 사용자가 지연 데이터에 대한 임곗값을 지정할 수 있는 워터마킹이라는 기능에 대한 지원이 추가되었습니다.

위의 주식 시세 표시기 예제를 참조하여 지연 데이터에 대한 허용 임곗값을 10분 이내로 가정해 보겠습니다. 간단하게 하기 위해 연속 기간을 가정하겠습니다(티커는 AMZ, 거래는 BUY).



위 다이어그램에서는 10분의 연속 기간 동안 총 볼륨을 계산하고 있습니다. 17:00, 17:10, 17:20에 트리거가 있습니다. 타임라인 화살표 위에는 입력 데이터 스트림이 있고 아래에는 무제한 결과 테이블이 있습니다.

첫 번째 10분의 연속 기간에서는 event_time을 기준으로 집계했고, total_volume은 30으로 계산되었습니다. 두 번째 이벤트 기간에서 Spark는 event_time=17:02의 첫 번째 데이터 이벤트를 받았습니다. 이는 지금까지 Spark에서 본 최대 event_time이므로 워터마크 임곗값은 10분 뒤로 설정됩니다(즉, watermark_event_time=16:52). 16:52 이후의 event_time이 있는 모든 데이터 이벤트는 시간 제

한 집계로 간주되며 그 이전의 모든 데이터 이벤트는 삭제됩니다. 이를 통해 Spark는 추가 10분 동안 중간 상태를 유지하여 지연 데이터를 수용할 수 있습니다. 17:08 즈음에 Spark는 임곗값 내에 있는 event_time=16:54의 이벤트를 수신했습니다. 따라서 Spark는'16:50~17:00' 이벤트 기간을 다시 계산했고 총 볼륨은 30에서 60으로 업데이트되었습니다.

그러나 트리거 시간 17:20에 Spark가 event_time=17:15의 이벤트를 수신하면 watermark_event_time=17:05로 설정됩니다. 따라서 event_time=17:03의 지연 데이터 이벤트는 '너무 늦음'으로 간주되어 무시되었습니다.

```
Watermark Boundary = Max(Event Time) - Watermark Threshold
```

AWS Glue에서 워터마크 사용

Spark는 워터마크 경계를 통과할 때까지 외부 싱크에 데이터를 내보내거나 쓰지 않습니다. AWS Glue에서 워터마크를 구현하려면 아래 예제를 참조하세요.

```
grouped_df = parsed_df \
                .withWatermark("event_time", "10 minutes") \
                .groupBy(window("event_time", "5 minutes"), "ticker") \
                .agg(sum("volume").alias("total_volume"))
```

AWS Glue 스트리밍 작업 모니터링 정보

스트리밍 작업을 모니터링하는 것은 ETL 파이프라인 구축의 중요한 부분입니다. Spark UI 외에도 Amazon CloudWatch를 사용하여 지표를 모니터링할 수도 있습니다. 다음은 AWS Glue 프레임워크에서 내보내는 스트리밍 지표 목록입니다. 모든 AWS Glue 지표의 전체 목록은 Amazon CloudWatch 지표를 사용한 AWS Glue 모니터링을 참조하세요.

AWS Glue는 구조화된 스트리밍 프레임워크를 사용하여 입력 이벤트를 처리합니다. 코드에서 Spark API를 직접 사용하거나 이러한 지표를 게시하는 `GlueContext`에서 제공하는 `ForEachBatch`를 활용할 수 있습니다. 이러한 지표를 이해하려면 먼저 `windowSize`를 파악해야 합니다.

windowSize: `windowSize`는 사용자가 제공하는 마이크로 배치 간격입니다. 기간을 60초로 지정하면 AWS Glue 스트리밍 작업이 60초 또는 그 이상(이전 배치가 완료되지 않은 경우) 대기한 후 스트리밍 소스에서 배치의 데이터를 읽고 `ForEachBatch`에 제공된 변환을 적용합니다. 이를 트리거 간격이라고도 합니다.

상태 및 성능 특성을 이해하기 위해 지표를 더 자세히 검토해 보겠습니다.

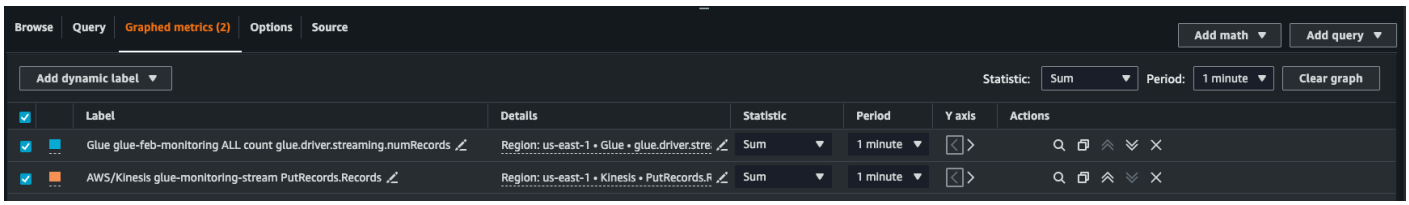
> **Note**
>
> 지표는 30초마다 내보내집니다. windowSize가 30초 미만인 경우 보고된 지표는 집계입니다. 예를 들어, windowSize가 10초이고 마이크로 배치당 20개의 레코드를 꾸준히 처리하고 있다고 가정해 보겠습니다. 이 시나리오에서 numRecords에 대해 내보낸 지표 값은 60입니다. 사용 가능한 데이터가 없으면 지표가 내보내지지 않습니다. 또한 소비자 지연 지표의 경우 해당 지표를 가져오는 기능을 활성화해야 합니다.

AWS Glue 스트리밍 지표 시각화

시각적 지표 도표화

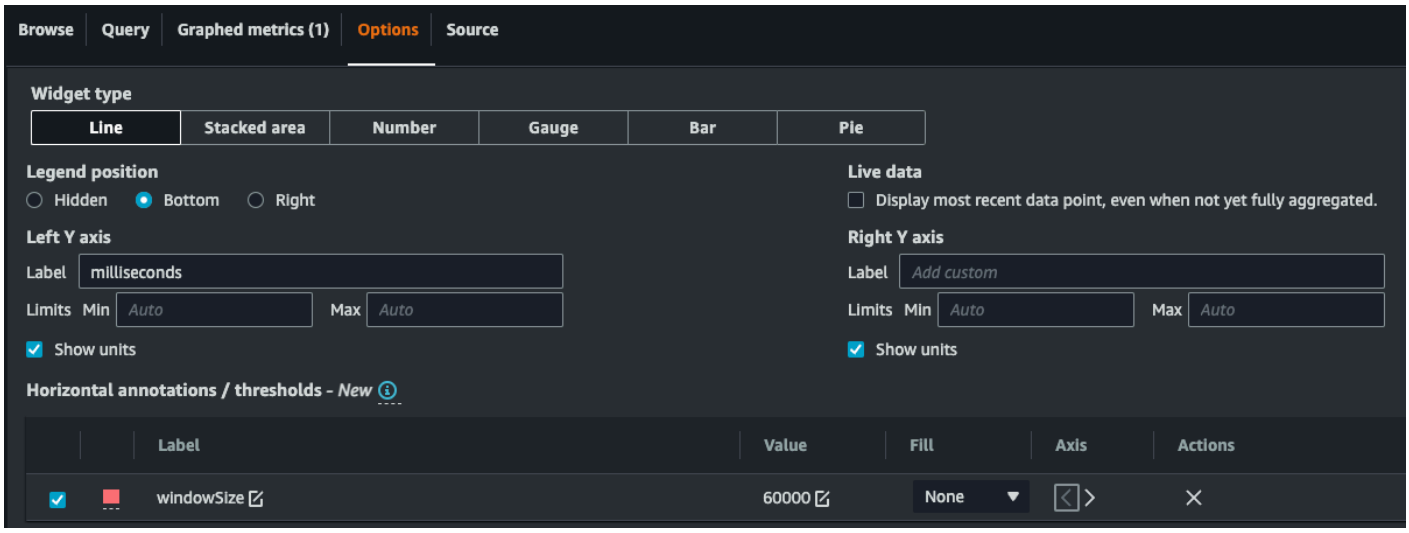
1. Amazon CloudWatch 콘솔에서 지표로 이동한 다음 찾아보기 탭을 선택합니다. 그런 다음 '사용자 지정 네임스페이스'에서 Glue를 선택합니다.

2. 작업 지표를 선택하여 모든 작업에 대한 지표를 표시합니다.
3. JobName=glue-feb-monitoring과 JobRunId=ALL을 기준으로 지표를 필터링합니다. 아래 그림에 표시된 대로 '+' 기호를 클릭하여 검색 필터에 지표를 추가할 수 있습니다.
4. 관심 있는 지표의 확인란을 선택합니다. 아래 그림에서는 numberAllExecutors와 numberMaxNeededExecutors를 선택했습니다.

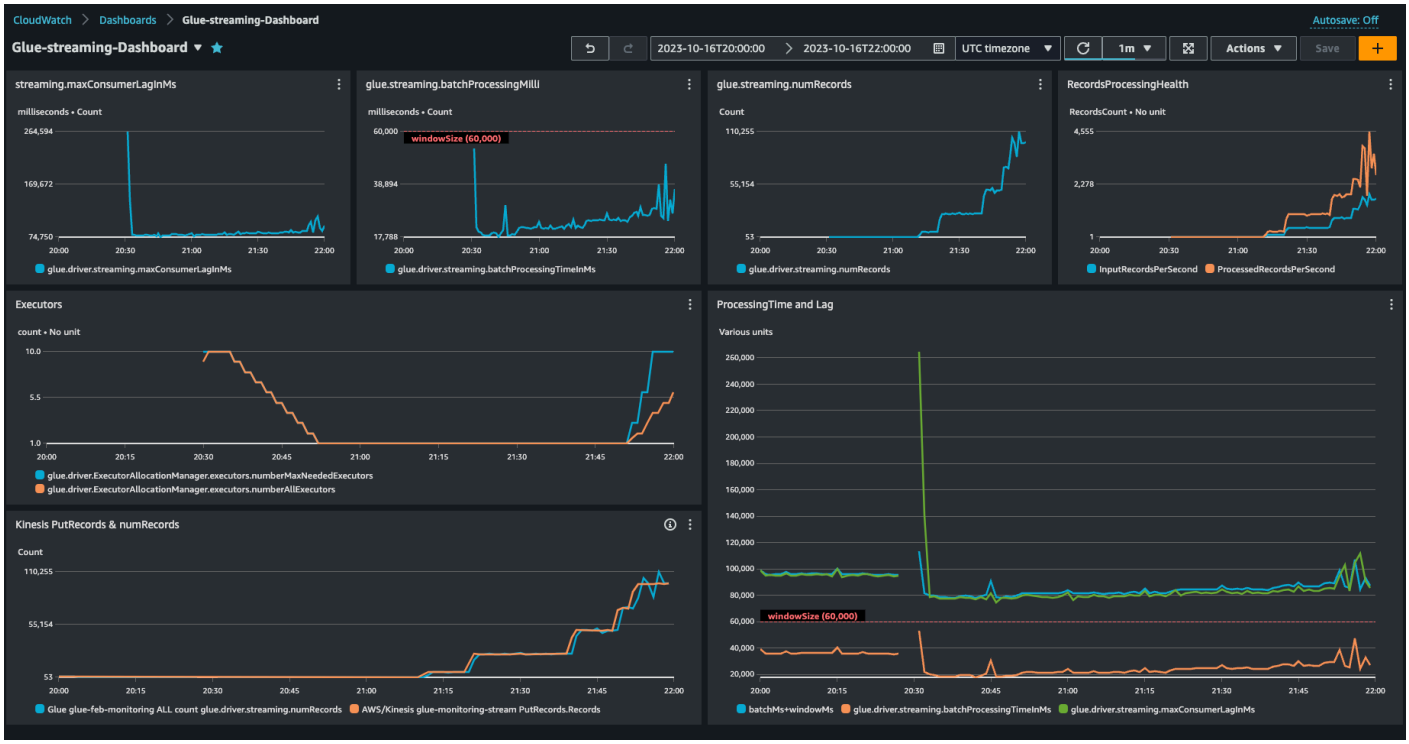


5. 이러한 지표를 선택한 후에는 그래프로 표시된 지표 탭으로 이동하여 통계를 적용할 수 있습니다.
6. 지표가 1분마다 내보내지기 때문에 batchProcessingTimeInMs와 maxConsumerLagInMs에 대해 1분 동안의 '평균'을 적용할 수 있습니다. numRecords에 대해 1분마다 '합계'를 적용할 수 있습니다.

7. 옵션 탭을 사용하여 그래프에 가로 `windowSize` 주석을 추가할 수 있습니다.



8. 지표를 선택했으면 대시보드를 생성하고 지표를 추가합니다. 다음은 샘플 대시보드입니다.

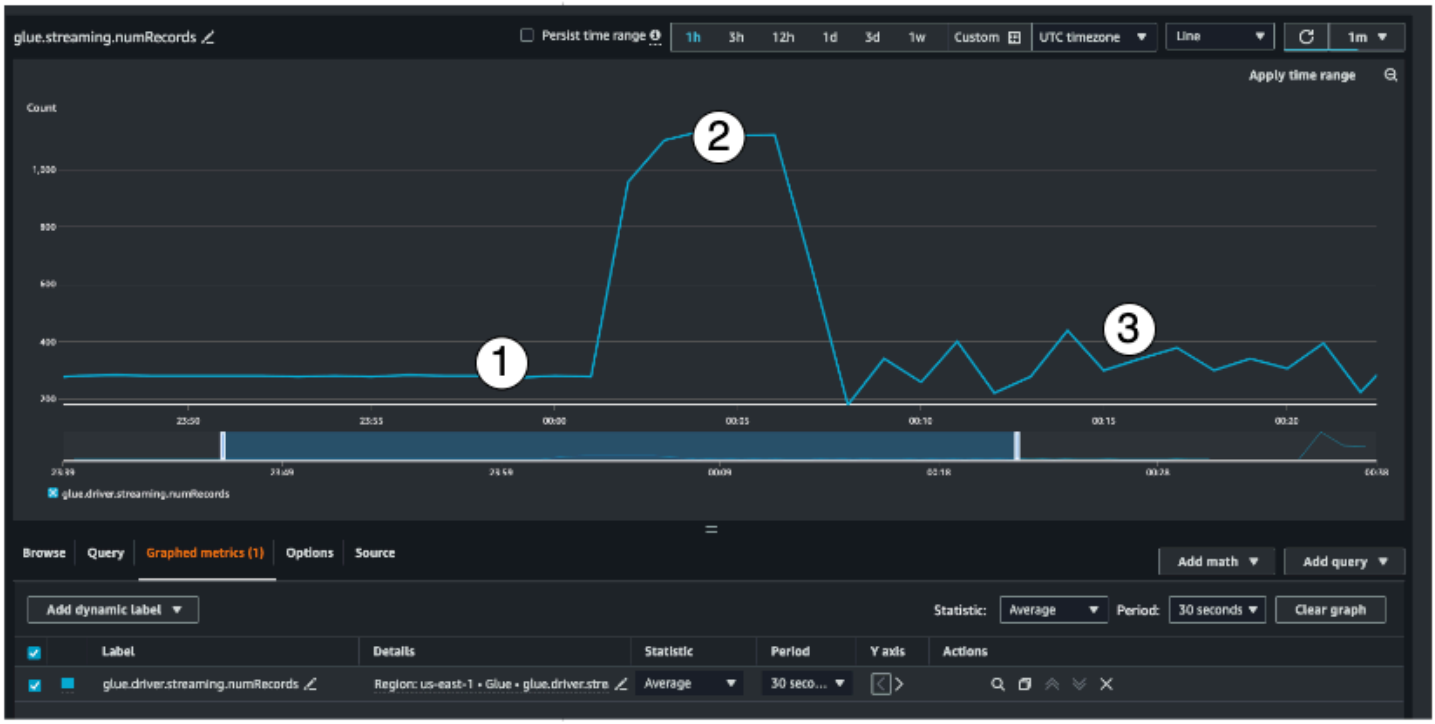


AWS Glue 스트리밍 지표 사용

이 섹션에서는 각 지표와 이러한 지표가 서로 어떻게 상호 연관되는지 설명합니다.

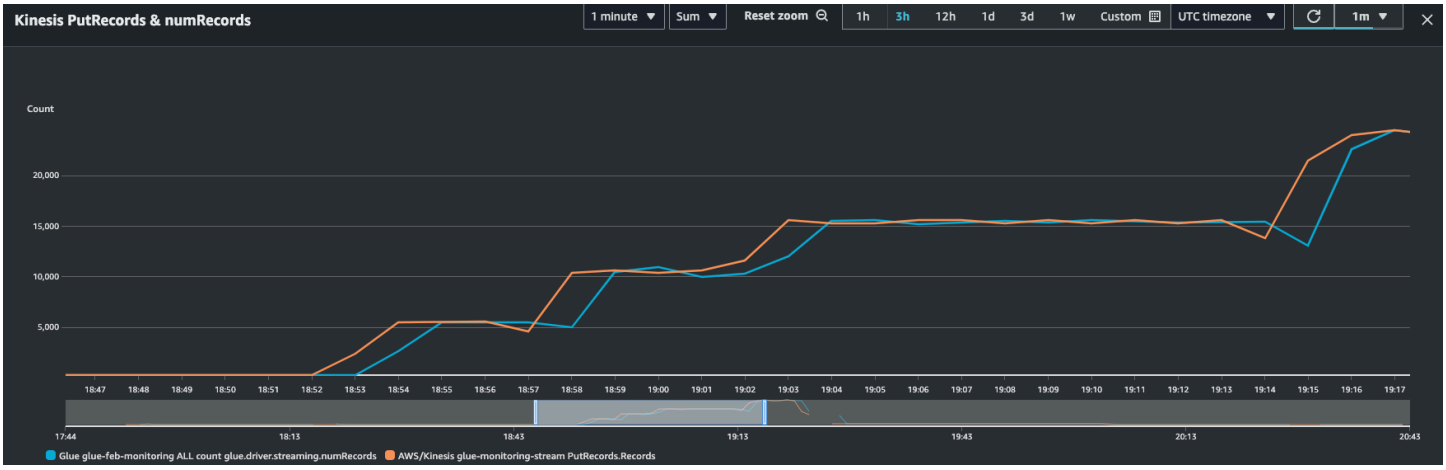
레코드 수(지표: streaming.numRecords)

이 지표는 처리 중인 레코드 수를 나타냅니다.



이 스트리밍 지표는 기간 중 처리하고 있는 레코드 수에 대한 가시성을 제공합니다. 이는 처리 중인 레코드 수와 함께 입력 트래픽의 동작을 이해하는 데도 도움이 됩니다.

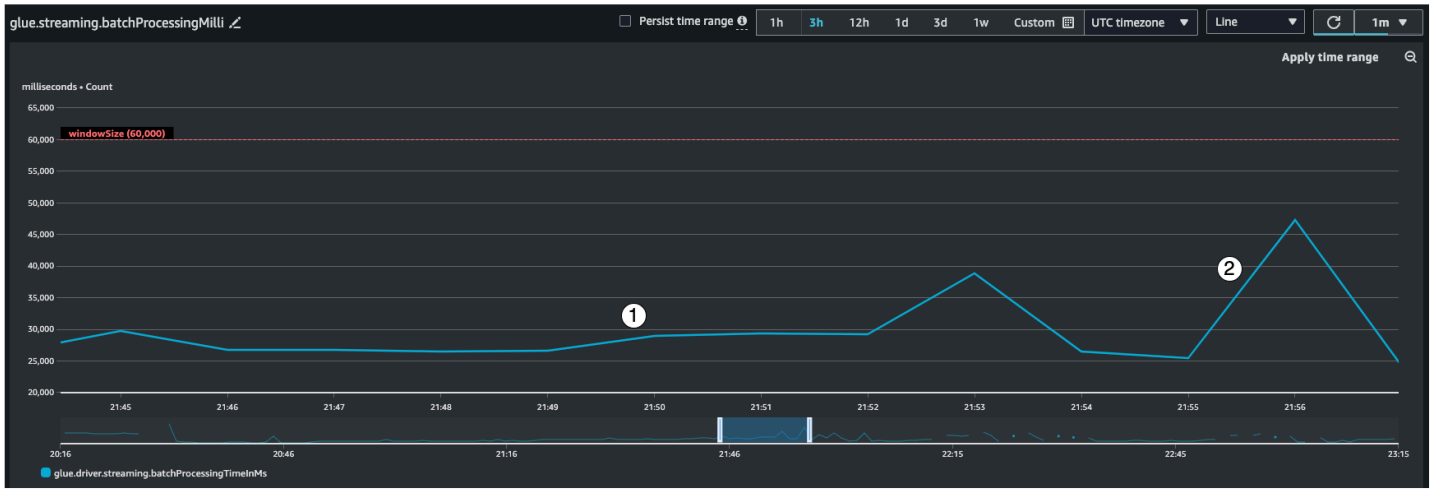
- 표시기 #1은 버스트가 없는 안정적인 트래픽의 예제를 보여줍니다. 일반적으로 이는 일정 간격으로 데이터를 수집하여 스트리밍 소스로 전송하는 IoT 센서와 같은 애플리케이션입니다.
- 표시기 #2는 안정된 부하에서 트래픽이 갑자기 급증하는 예제를 보여줍니다. 이는 블랙 프라이데이와 같은 마케팅 이벤트가 있고 클릭 수가 급증할 때 클릭스트림 애플리케이션에서 발생할 수 있습니다.
- 표시기 #3은 예측할 수 없는 트래픽의 예제를 보여줍니다. 트래픽을 예측할 수 없으면 문제가 있는 것입니다. 이는 입력 데이터의 특성일 뿐입니다. IoT 센서 예제로 돌아가서 수백 개의 센서가 스트리밍 소스로 날씨 변화 이벤트를 전송하고 있다고 가정해 보겠습니다. 날씨 변화를 예측할 수 없으므로 데이터도 예측할 수 없습니다. 트래픽 패턴을 파악하는 것이 실행기 크기 조정의 핵심입니다. 입력이 급증하는 경우 자동 크기 조정 사용을 고려할 수 있습니다(나중에 자세히 설명).



이 지표를 Kinesis PutRecords 지표와 결합하여 수집하고 있는 이벤트 수와 읽고 있는 레코드 수가 거의 동일한지 확인할 수 있습니다. 이는 지연을 이해하려고 할 때 특히 유용합니다. 수집 속도가 증가함에 따라 AWS Glue에서 읽는 numRecords도 증가합니다.

배치 처리 시간(지표: streaming.batchProcessingTimeInMs)

배치 처리 시간 지표는 클러스터가 과소 프로비저닝되었는지 과잉 프로비저닝되었는지 확인하는 데 도움이 됩니다.

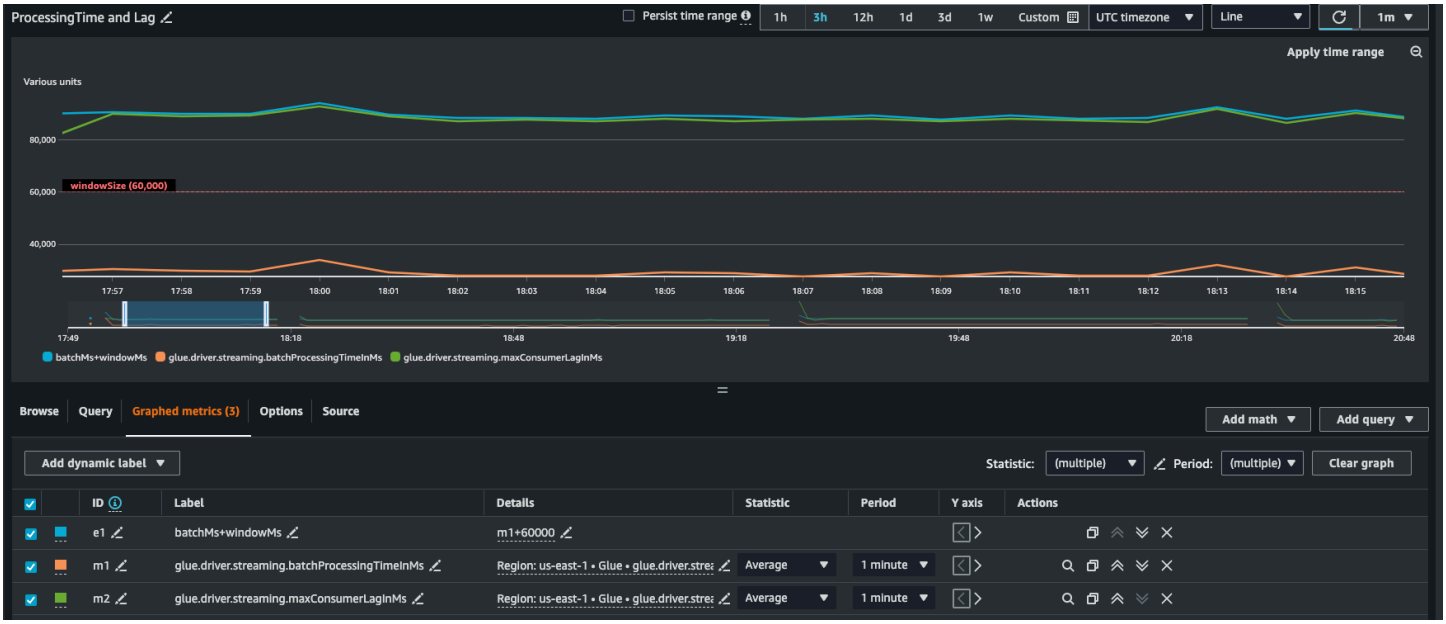


이 지표는 레코드의 각 마이크로 배치를 처리하는 데 걸린 시간(밀리초)을 나타냅니다. 여기서 주요 목표는 이 시간을 모니터링하여 windowSize 간격보다 작은지 확인하는 것입니다. 다음 기간 간격에서 복구되는 한 batchProcessingTimeInMs가 일시적으로 초과되어도 괜찮습니다. 표시기 #1은 작업을 처리하는 데 걸리는 시간이 다소 안정적임을 보여줍니다. 그러나 입력 레코드 수가 늘어나면 표시기 #2와 같이 작업을 처리하는 데 걸리는 시간이 늘어납니다. numRecords가 증가하지 않지만 처리 시간이 증가하는 경우 실행기의 작업 처리를 더 자세히 살펴봐야 합니다. batchProcessingTimeInMs가

10분 이상 120%를 넘지 않도록 임곗값과 경보를 설정하는 것이 좋습니다. 경보 설정에 대한 자세한 내용은 Amazon CloudWatch 경보 사용을 참조하세요.

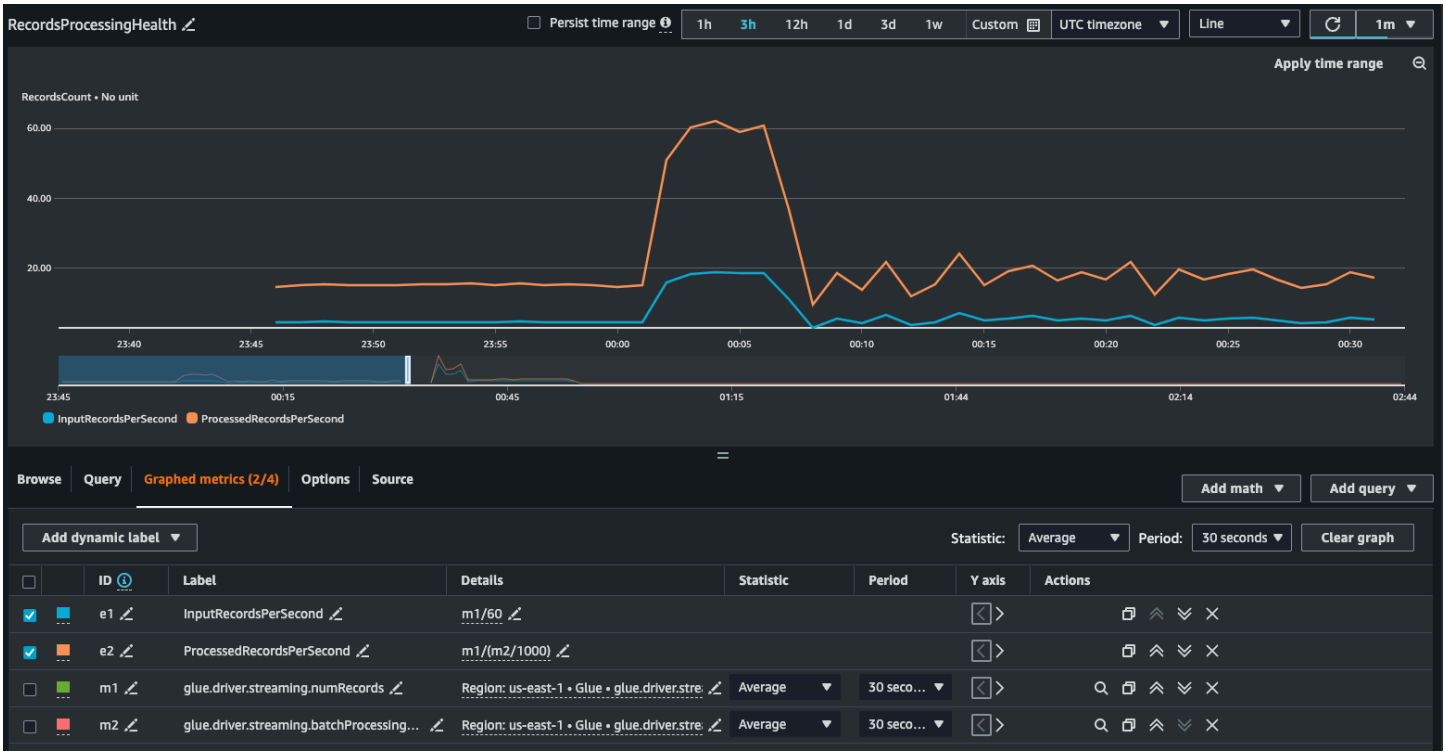
소비자 지연(지표: streaming.maxConsumerLagInMs)

소비자 지연 지표는 이벤트 처리에 지연이 있는지 파악하는 데 도움이 됩니다. 지연이 너무 길면 올바른 windowSize가 있더라도 비즈니스가 의존하는 처리 SLA를 놓칠 수 있습니다. `emitConsumerLagMetrics` 연결 옵션을 사용하여 이 지표를 명시적으로 활성화해야 합니다. 자세한 내용은 KinesisStreamingSourceOptions를 참조하세요.



파생된 지표

더 심층적인 인사이트를 얻으려면 Amazon CloudWatch에서 스트리밍 작업에 대해 더 자세히 이해하기 위해 파생된 지표를 생성할 수 있습니다.



파생된 지표로 그래프를 작성하여 더 많은 DPU를 사용해야 하는지 결정할 수 있습니다. 자동 크기 조정은 이를 자동으로 수행하는 데 도움이 되지만 파생된 지표를 사용하여 자동 크기 조정이 효과적으로 작동하는지 확인할 수 있습니다.

- InputRecordsPerSecond는 입력 레코드를 가져오는 속도를 나타냅니다. 입력 레코드 수 (glue.driver.streaming.numRecords)/WindowSize로 파생됩니다.
- ProcessingRecordsPerSecond는 레코드가 처리되는 속도를 나타냅니다. 입력 레코드 수 (glue.driver.streaming.numRecords)/batchProcessingTimeInMs로 파생됩니다.

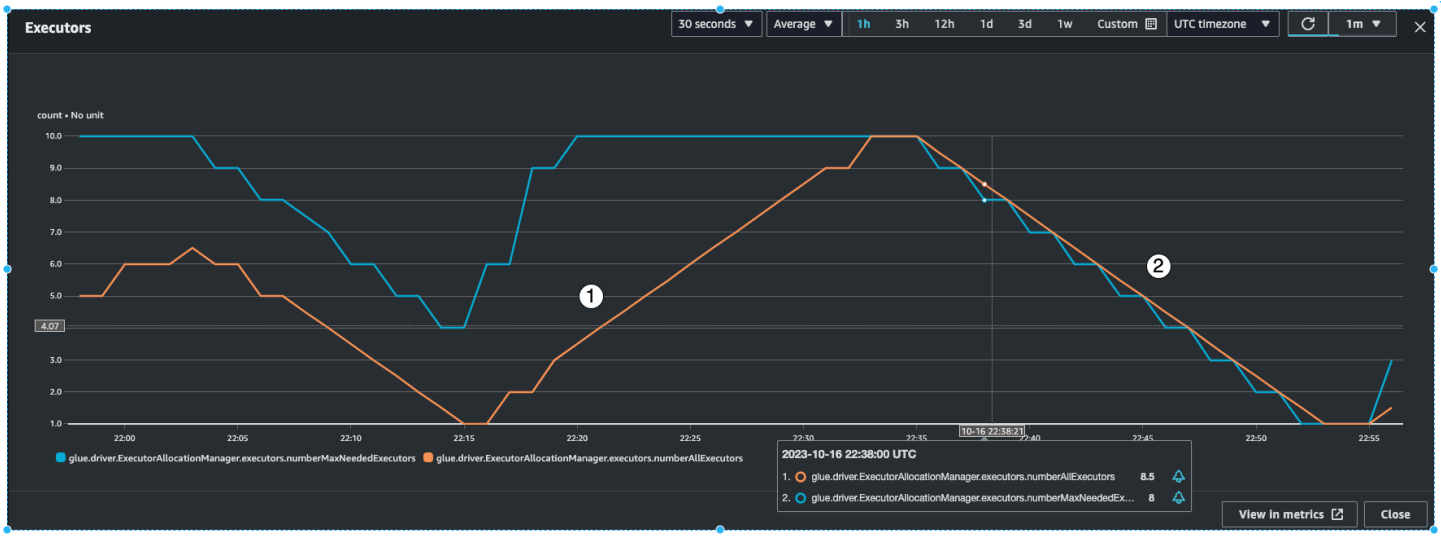
입력 속도가 처리 속도보다 높으면 작업을 처리하기 위해 용량을 더 추가하거나 병렬 처리를 늘려야 할 수 있습니다.

자동 크기 조정 지표

입력 트래픽이 급증하는 경우 자동 크기 조정 활성화를 고려하고 최대 워커 수를 지정해야 합니다. 이를 통해 `numberAllExecutors`와 `numberMaxNeededExecutors`의 두 가지 추가 지표를 얻을 수 있습니다.

- numberAllExecutors는 능동적으로 실행 중인 작업 실행기 수입니다.
- numberMaxNeededExecutors는 현재 로드를 충족하는 데 필요한 최대 작업 (능동 실행 및 보류) 실행기 수입니다.

이 두 지표는 자동 크기 조정이 제대로 작동하는지 파악하는 데 도움이 됩니다.



AWS Glue는 몇 가지 마이크로 배치를 통해 batchProcessingTimeInMs 지표를 모니터링하고 두 가지 중 하나를 수행합니다. batchProcessingTimeInMs가 windowSize에 더 가까울 경우 실행기를 스케일 아웃하고, batchProcessingTimeInMs가 windowSize보다 비교적 낮을 경우 실행기를 스케일 인합니다. 또한 실행기의 단계적 크기 조정을 위한 알고리즘을 사용합니다.

- 표시기 #1은 로드 처리에 필요한 최대 실행기를 따라잡기 위해 활성 실행기가 어떻게 스케일 업되었는지 보여줍니다.
- 표시기 #2는 batchProcessingTimeInMs가 낮았기 때문에 활성 실행기가 어떻게 스케일 인되었는지 보여줍니다.

이러한 지표를 사용하여 현재 실행기 수준의 병렬 처리를 모니터링하고 이에 따라 자동 크기 조정 구성의 최대 워커 수를 조정할 수 있습니다.

최고의 성능을 얻는 방법

Spark는 Amazon Kinesis 스트림에서 읽기 위해 샤드당 하나의 작업을 생성하려고 합니다. 각 샤드의 데이터가 파티션이 됩니다. 그런 다음 각 워커의 코어 수에 따라 이러한 작업을 실행기/워커 전체에 분배합니다(워커당 코어 수는 선택하는 워커 유형(G.025X, G.1X 등)에 따라 다름). 그러나 작업이 분배되는 방식은 비결정적입니다. 모든 작업은 해당 코어에서 병렬로 실행됩니다. 샤드가 사용 가능한 실행기 코어 수보다 많으면 작업이 대기열에 추가됩니다.

위의 지표와 샤드 수를 조합해서 사용하여 버스트를 위한 여유 공간이 있는 안정적인 로드를 위해 실행기를 프로비저닝할 수 있습니다. 대략적인 워커 수 확인을 위해 작업을 몇 번 반복해서 실행하는 것이

좋습니다. 불안정하거나 급증하는 워크로드의 경우 자동 크기 조정과 최대 워커 수를 설정하여 동일한 작업을 수행할 수 있습니다.

비즈니스의 SLA 요구 사항에 따라 `windowSize`를 설정합니다. 예를 들어, 비즈니스상 처리된 데이터가 120초를 초과해서는 안 되는 경우, 평균 소비자 지연이 120초 미만이 되도록 `windowSize`를 60초 이상으로 설정합니다(위의 소비자 지연 섹션 참조). 여기에서 샤드의 `numRecords`와 수에 따라 DPU 단위로 용량을 계획하여 `batchProcessingTimeInMs`가 대부분의 경우 `windowSize`의 70% 미만이 되도록 합니다.

> **Note**
>
> 핫 샤드는 데이터 스큐를 일으킬 수 있으며, 이는 일부 샤드/파티션이 다른 샤드/파티션보다 훨씬 크다는 것을 의미합니다. 이로 인해 병렬로 실행되는 일부 작업의 경우 시간이 오래 걸려 작업이 지연될 수 있습니다. 결과적으로 이전 배치의 모든 작업이 완료될 때까지 다음 배치를 시작할 수 없으며 이는 `batchProcessingTimeInMillis`와 최대 지연에 영향을 미칩니다.

제로 ETL 통합

제로 ETL은 일반적인 수집 및 복제 사용 사례에서 ETL 데이터 파이프라인을 구축할 필요성을 최소화하는 AWS의 완전 관리형 통합 세트입니다. 이를 통해 여러 운영, 트랜잭션 및 애플리케이션 소스에서 Amazon SageMaker Lakehouse 및 Amazon Redshift의 데이터를 사용할 수 있습니다. 제로 ETL 통합을 통해 분석, AI/ML 및 보고를 위한 최신 데이터를 확보할 수 있습니다. 비즈니스 대시보드, 최적화된 게임 경험, 데이터 품질 모니터링, 고객 행동 분석과 같은 사용 사례에 대해 더 정확하고 시기적절한 인사이트를 얻을 수 있습니다. 더 신뢰성 있게 데이터 기반 예측을 수행하고, 고객 경험을 개선하고, 비즈니스 전반에서 데이터 기반 인사이트를 촉진할 수 있습니다.

Amazon Redshift는 속도가 빠른 페타바이트 규모의 완전 관리형 데이터 웨어하우스 서비스로, 간편하고 비용 효율적으로 모든 데이터를 기존 비즈니스 인텔리전스 도구를 사용하여 효율적으로 분석할 수 있게 해줍니다.

Amazon SageMaker Lakehouse는 Amazon Simple Storage Service(S3) 데이터 레이크와 Amazon Redshift 데이터 웨어하우스의 모든 데이터를 통합하여 단일 데이터 사본에 기반하는 강력한 분석 및 AI/ML 애플리케이션을 구축할 수 있도록 지원합니다. SageMaker Lakehouse는 모든 Apache Iceberg 호환 도구 및 엔진을 사용하여 현재 위치의 데이터에 액세스하고 쿼리할 수 있는 유연성을 제공합니다. SageMaker Lakehouse를 사용하면 Apache Iceberg 호환 도구 및 엔진을 사용하여 현재 위치의 데이터에 액세스하고 쿼리할 수 있는 유연성도 얻을 수 있습니다. 또한 모든 분석 도구 및 엔진의 모든 데이터에 적용되는 통합되고 세분화된 액세스 제어를 사용하여 데이터를 보호할 수 있습니다. 이제 권한을 정의하고 조직 전체에서 데이터를 자신 있게 공유하세요.

AWS Glue의 제로 ETL 기능

AWS Glue의 제로 ETL 통합은 AWS 데이터 서비스 및 타사 애플리케이션에서 AWS 대상으로의 데이터 수집 및 복제를 간소화합니다.

AWS Glue의 제로 ETL 소스에서 지원하는 AWS 서비스는 다음과 같습니다.

- Amazon DynamoDB

제로 ETL에서 지원하는 타사 애플리케이션은 다음과 같습니다.

- Facebook 광고
- Instagram 광고
- Salesforce

- Salesforce Marketing Cloud Account Engagement
- SAP OData
- ServiceNow
- Zendesk
- Zoho CRM

AWS Glue의 제로 ETL 대상에서 지원하는 AWS 서비스는 다음과 같습니다.

- Amazon Redshift
- Amazon SageMaker Lakehouse

> **Note**
>
> AWS Glue에서 Amazon DynamoDB 소스를 포함하는 제로 ETL 통합을 생성할 때 대상은 Amazon SageMaker Lakehouse에서 지원합니다.

제로 ETL 통합을 설정하기 위한 사전 조건

소스와 대상 간에서 통합을 설정하려면 AWS Glue가 소스의 데이터에 액세스하고 대상에 쓰는 데 사용하는 IAM 역할 구성, 중간 또는 대상 위치의 데이터를 암호화하기 위한 KMS 키 사용 등과 같은 몇 가지 사전 조건이 필요합니다.

주제

- 소스 리소스 설정
- 대상 리소스 설정
- Amazon Redshift 데이터 웨어하우스 생성
- 제로 ETL 통합을 위한 VPC 설정
- 제로 ETL 교차 계정 통합 설정

소스 리소스 설정

소스에 필요한 다음 설정 작업을 수행합니다.

소스 역할 설정

이 섹션에서는 제로 ETL 통합이 연결에 액세스할 수 있도록 소스 역할을 전달하는 방법을 설명합니다. 이 방법도 SaaS 소스에만 적용됩니다.

> **Note**
>
> 몇 개의 연결로만 액세스를 제한하려면 먼저 연결을 생성하여 연결 ARN을 가져올 수 있습니다. 제로 ETL 통합을 위한 소스 구성을(를) 참조하세요.

통합이 연결에 액세스할 수 있는 권한이 있는 역할을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GlueConnections",
            "Effect": "Allow",
            "Action": [
                "glue:GetConnections",
                "glue:GetConnection"
            ],
            "Resource": [
                "arn:aws:glue:*:<accountId>:catalog",
                "arn:aws:glue:us-east-1:<accountId>:connection/*"
            ]
        },
        {
            "Sid": "GlueActionBasedPermissions",
            "Effect": "Allow",
            "Action": [
                // Fetch entities:
                "glue:ListEntities",
                // Refresh connection credentials:
                "glue:RefreshOAuth2Tokens"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "glue.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

대상 리소스 설정

AWS Glue 데이터 카탈로그 또는 Amazon Redshift 데이터 웨어하우스 통합 대상에 필요한 다음 설정 작업을 수행합니다.

AWS Glue 데이터베이스 대상과 통합하는 경우:

- AWS Glue 데이터베이스 설정
- 카탈로그 리소스 기반 액세스(RBAC) 정책 제공
- 대상 IAM 역할 생성

Amazon Redshift 데이터 웨어하우스 대상과 통합하는 경우:

- https://docs.aws.amazon.com/glue/latest/dg/zero-etl-prerequisites.html#zero-etl-setup-target-redshift-data-warehouse

AWS Glue 데이터베이스 설정

AWS Glue 데이터베이스를 사용하는 통합의 경우:

Amazon S3 위치를 사용하여 AWS Glue Data Catalog에서 대상 데이터베이스를 설정하려면:

1. AWS Glue 콘솔 홈 페이지의 Data Catalog에서 데이터베이스를 선택합니다.
2. 오른쪽 상단 모서리에서 데이터 추가를 선택합니다. 데이터베이스를 이미 생성한 경우 Amazon S3 URI를 사용하는 위치가 데이터베이스에 대해 설정되어 있는지 확인합니다.
3. 이름과 위치(Amazon S3 URI)를 입력합니다. 제로 ETL 통합에는 위치가 필요합니다. 완료되면 데이터베이스 생성을 클릭합니다.

> **Note**
>
> Amazon S3 버킷은 AWS Glue 데이터베이스와 동일한 리전에 있어야 합니다.

AWS Glue에서 새 데이터베이스를 생성하는 방법에 대한 자세한 내용은 AWS Glue Data Catalog 시작하기를 참조하세요.

`create-database` CLI를 사용하여 AWS Glue에서 데이터베이스를 생성할 수도 있습니다. `--database-input`의 `LocationUri`는 필수입니다.

Iceberg 테이블 최적화

대상 데이터베이스에서 AWS Glue에 의해 테이블이 생성된 후 압축을 활성화하여 Amazon Athena에서 쿼리 속도를 높일 수 있습니다. 압축을 위한 리소스(IAM 역할) 설정에 대한 자세한 내용은 테이블 최적화 필수 조건을 참조하세요.

통합에 의해 생성된 AWS Glue 테이블에서 압축을 설정하는 방법에 대한 자세한 내용은 Iceberg 테이블 최적화를 참조하세요.

카탈로그 리소스 기반 액세스(RBAC) 정책 제공

AWS Glue 데이터베이스를 사용하는 통합의 경우 카탈로그 RBAC 정책에 다음 권한을 추가하여 소스와 대상 간의 통합을 허용합니다.

> **Note**
>
> 교차 계정 통합의 경우 Alice(통합을 생성하는 사용자) 역할 정책과 카탈로그 리소스 정책 모두 리소스에 대해 `glue:CreateInboundIntegration`을 허용해야 합니다. 동일한 계정의 경우에는 리소스 정책 또는 리소스에 대해 `glue:CreateInboundIntegration`을 허용하는 역할 정책으로 충분합니다. 두 시나리오 모두 여전히 `glue:AuthorizeInboundIntegration`에 대해 `glue.amazonaws.com`을 허용해야 합니다.

Data Catalog에서 카탈로그 설정에 액세스할 수 있습니다. 계속해서 다음 권한을 제공하고 누락된 정보를 입력합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    { // Allow Alice to create Integration on Target Database
      "Principal": {
        "AWS": [
          "arn:aws:iam::<source-account-id>:user/Alice"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:CreateInboundIntegration"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<Target-Account-Id>:catalog",
        "arn:aws:glue:<region>:<Target-Account-Id>:database/DatabaseName"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:dynamodb:<region>:<Account>:table/<table-name>"
        }
      }
    },
    { // Allow Glue to Authorize the Inbound Integration on behalf of Bob
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:AuthorizeInboundIntegration"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<Target-Account-Id>:catalog",
        "arn:aws:glue:<region>:<Target-Account-Id>:database/DatabaseName"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:dynamodb:<region>:<account-id>:table/<table-name>"
```

```
        }
      }
    }
  ]
}
```

대상 IAM 역할 생성

다음과 같은 권한 및 신뢰 관계를 가진 대상 IAM 역할을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::<target iceberg table s3 bucket>",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::<target iceberg table s3 bucket>/prefix/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "glue:GetDatabase"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<account-id>:catalog",
                "arn:aws:glue:<region>:<account-id>:database/DatabaseName"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "glue:CreateTable",
                "glue:GetTable",
                "glue:UpdateTable",
                "glue:GetTableVersion",
```

```
                "glue:GetTableVersions",
                "glue:GetResourcePolicy"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<account-id>:catalog",
                "arn:aws:glue:<region>:<account-id>:database/<DatabaseName>",
                "arn:aws:glue:<region>:<account-id>:table/<DatabaseName>/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "AWS/Glue/ZeroETL"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

AWS Glue 서비스가 역할을 수임할 수 있도록 다음 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Amazon Redshift 데이터 웨어하우스 생성

제로 ETL 통합 대상이 Amazon Redshift 데이터 웨어하우스이지만 아직 없는 경우 데이터 웨어하우스를 생성합니다. Amazon Redshift Serverless 작업 그룹을 생성하려면 네임스페이스가 있는 작업 그룹 생성을 참조하세요. Amazon Redshift 클러스터를 생성하려면 클러스터 생성을 참조하세요.

통합에 성공하려면 대상 Amazon Redshift 작업 그룹 또는 클러스터에 `enable_case_sensitive_identifier` 파라미터가 켜져 있어야 합니다. 대소문자 구분 활성화에 대한 자세한 내용은 Amazon Redshift 관리 안내서의 데이터 웨어하우스에 대소문자 구분 기능 사용 설정을 참조하세요.

Amazon Redshift 작업 그룹 또는 클러스터 설정이 완료되면 데이터 웨어하우스를 구성해야 합니다. 자세한 내용은 Amazon Redshift 관리 안내서의 제로 ETL 통합 시작하기를 참조하세요.

제로 ETL 통합을 위한 VPC 설정

제로 ETL 통합을 위한 VPC를 설정하려면:

1. VPC > VPC로 이동하여 VPC 생성을 선택합니다.
   a. VPC 등을 선택합니다.
   b. VPC 이름을 설정합니다.
   c. IPv4 CIDR을 10.0.0.0/16으로 설정합니다.
   d. AZ 수를 1로 설정합니다.
   e. 퍼블릭 및 프라이빗 서브넷 수를 1로 설정합니다.
   f. NAT 게이트웨이를 없음으로 설정합니다.
   g. VPC 엔드포인트를 S3 게이트웨이로 설정합니다.
   h. DNS 호스트 이름 및 DNS 확인을 활성화합니다.
2. 엔드포인트로 이동하고 엔드포인트 생성을 선택합니다.
3. VPC의 프라이빗 서브넷에서 다음 서비스에 대한 엔드포인트를 생성합니다(기본 보안 그룹 사용).
   a. com.amazonaws.us-east-1.lambda
   b. com.amazonaws.us-east-1.glue

c. com.amazonaws.us-east-1.sts

AWS Glue 연결을 생성합니다.

1. AWS Glue > 데이터 연결로 이동하고 연결 생성을 선택합니다.
2. 네트워크를 선택합니다.
3. 생성한 VPC, 서브넷(프라이빗) 및 기본 보안 그룹을 선택합니다.

VPC의 대상 역할 설정

대상 역할에는 제로 ETI 통합에 필요한 다른 권한 외에 다음과 같은 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CustomerVpc",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags",
                "ec2:DescribeRouteTables",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "glue:GetConnection"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

대상 레그 리소스 속성 설정

CLI를 사용하는 경우 대상 레그 리소스 속성을 생성한 대상 AWS Glue 데이터베이스로 설정합니다. 대상 역할 ARN과 AWS Glue 연결 이름을 전달합니다.

```
aws glue create-integration-resource-property \
--resource-arn arn:aws:glue:us-east-1:<account-id>:database/exampletarget \
--target-processing-properties '{"RoleArn" : "arn:aws:iam::<account-id>:role/example-
role", "ConnectionName":"example-vpc-3"}' \
--endpoint-url https://example.amazonaws.com --region us-east-1
```

가능한 클라이언트 오류

다음은 VPC로 구성된 통합에서 발생할 수 있는 클라이언트 오류입니다.

오류 메시지	필요한 작업
제공된 역할은 연결 시 glue:GetConnection을 수행할 권한이 없습니다. 역할 정책에 이 권한을 추가한 다음 통합이 복구될 때까지 기다립니다.	역할 정책 업데이트
제공된 역할은 ec2:DescribeSubnets를 수행할 권한이 없습니다. 역할 정책에 이 권한을 추가한 다음 통합이 복구될 때까지 기다립니다.	역할 정책 업데이트
제공된 역할은 ec2:DescribeSecurityGroups를 수행할 권한이 없습니다. 역할 정책에 이 권한을 추가한 다음 통합이 복구될 때까지 기다립니다.	역할 정책 업데이트
제공된 역할은 ec2:DescribeVpcEndpoints를 수행할 권한이 없습니다. 역할 정책에 이 권한을 추가한 다음 통합이 복구될 때까지 기다립니다.	역할 정책 업데이트
제공된 역할은 ec2:DescribeRouteTables를 수행할 권한이 없습니다. 역할 정책에 이 권한을 추가한 다음 통합이 복구될 때까지 기다립니다.	역할 정책 업데이트
제공된 역할은 ec2:CreateTags를 수행할 권한이 없습니다. 역할 정책에 이 권한을 추가한 다음 통합이 복구될 때까지 기다립니다.	역할 정책 업데이트

오류 메시지	필요한 작업
제공된 역할은 ec2:CreateNetworkInterface를 수행할 권한이 없습니다. 역할 정책에 이 권한을 추가한 다음 통합이 복구될 때까지 기다립니다.	역할 정책 업데이트
제공된 연결 서브넷에 유효한 S3 엔드포인트 또는 NAT 게이트웨이가 포함되어 있지 않습니다. 서브넷을 업데이트한 다음 통합이 복구될 때까지 기다립니다.	VPC 서브넷 엔드포인트 업데이트
연결 서브넷을 찾을 수 없습니다. 연결 서브넷을 업데이트한 다음 통합이 복구될 때까지 기다립니다.	AWS Glue 연결 업데이트
연결 보안 그룹을 찾을 수 없습니다. 연결 보안 그룹을 업데이트한 다음 통합이 복구될 때까지 기다립니다.	AWS Glue 연결 업데이트
제공된 VPC 연결을 통해 S3에 연결할 수 없습니다. 서브넷 구성을 업데이트한 다음 통합이 복구될 때까지 기다립니다.	VPC 서브넷 엔드포인트 업데이트
제공된 VPC 연결을 통해 Lambda에 연결할 수 없습니다. 서브넷 구성을 업데이트한 다음 통합이 복구될 때까지 기다립니다.	VPC 서브넷 엔드포인트 업데이트

제로 ETL 교차 계정 통합 설정

제로 ETL 교차 계정 통합을 설정하려면:

1. 카탈로그 리소스 기반 액세스(RBAC) 정책 제공에 설명된 대로 대상 리소스 정책을 구성합니다. 소스 계정 역할이 대상 리소스에 명시적으로 허용되는지 확인합니다.
2. 소스 계정 역할(통합을 생성하는 데 사용되는 역할)에 다음이 있는지 확인합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

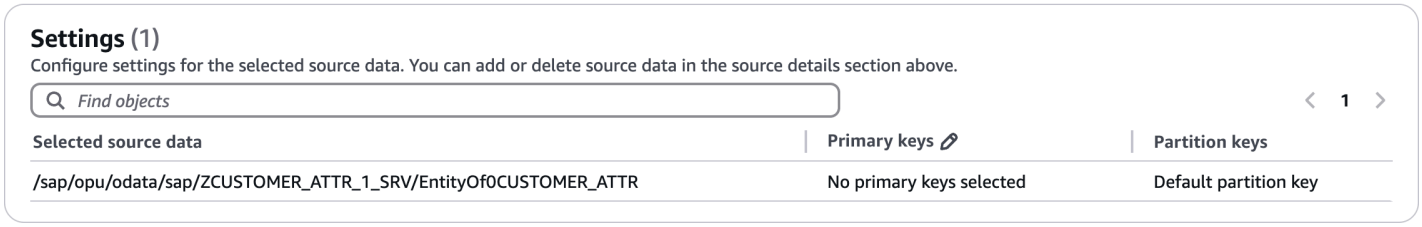
```
    {
        "Sid": "Stmt123456789012",
        "Action": [
            "glue:CreateInboundIntegration"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:glue:<region>:<target-account-id>:catalog",
            "arn:aws:glue:<region>:<target-account-id>:database/DatabaseName"
        ]
    }]
}
```

3. 통합 생성에 설명된 대로 통합을 생성합니다.

제로 ETL 통합을 위한 소스 구성

특수 SAP 엔터티 지원

대부분의 SaaS 엔터티에 대해 데이터 처리 중에 유효한 프라이머리 키 세트를 결정하지만, 유효한 프라이머리 키 세트를 입력으로 제공하는 추가 단계가 필요한 일부 엔터티, 특히 `EntityOf`로 시작하는 SAP 엔터티가 있습니다. `EntityOf` 엔터티를 선택하면 프라이머리 키 세트를 제공하라는 메시지가 표시됩니다.



Amazon DynamoDB 소스 구성

소스 Amazon DynamoDB 테이블의 데이터에 액세스하려면 AWS Glue에 테이블을 설명하고 테이블에서 데이터를 내보낼 수 있는 액세스 권한이 필요합니다. Amazon DynamoDB는 최근에 리소스 기반 액세스(RBAC) 정책을 구성할 수 있는 기능을 도입했습니다.

다음 예제 리소스 기반 액세스(RBAC) 정책에서는 통합을 위해 와일드카드(*)를 사용합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "1111",
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Resource":"*",
        "Action": [
            "dynamodb:ExportTableToPointInTime",
            "dynamodb:DescribeTable",
            "dynamodb:DescribeExport"
        ],
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "<account-id>"
            },
            "StringLike": {
                "aws:SourceArn": "arn:aws:glue:<region>:<account-id>:integration:*"
            }
        }
    }]
}
```

1. 복제하려는 DynamoDB의 경우 위의 RBAC 정책 템플릿을 테이블에 대한 리소스 기반 정책에 붙여 넣고 필드를 채웁니다.
2. 정책을 제한하려면 통합을 생성한 후 정책을 업데이트하고 전체 `integrationArn`을 지정한 다음 `StringLike` 대신 `StringEquals` 조건을 사용해야 합니다.
3. DynamoDB 테이블에 대해 시점 복구(PITR)가 활성화되어 있는지 확인합니다.
4. 리소스 기반 액세스(RBAC) 정책에 `Describe Export`를 추가했는지 확인합니다.

다음 명령을 사용하여 RBAC 정책을 테이블에 추가할 수도 있습니다.

```
aws dynamodb put-resource-policy \
--resource-arn arn:aws:dynamodb:<region>:<account-id>:table/<ddb-table-name> \
--policy file://resource-policy-with-condition.json \
--region <region>
```

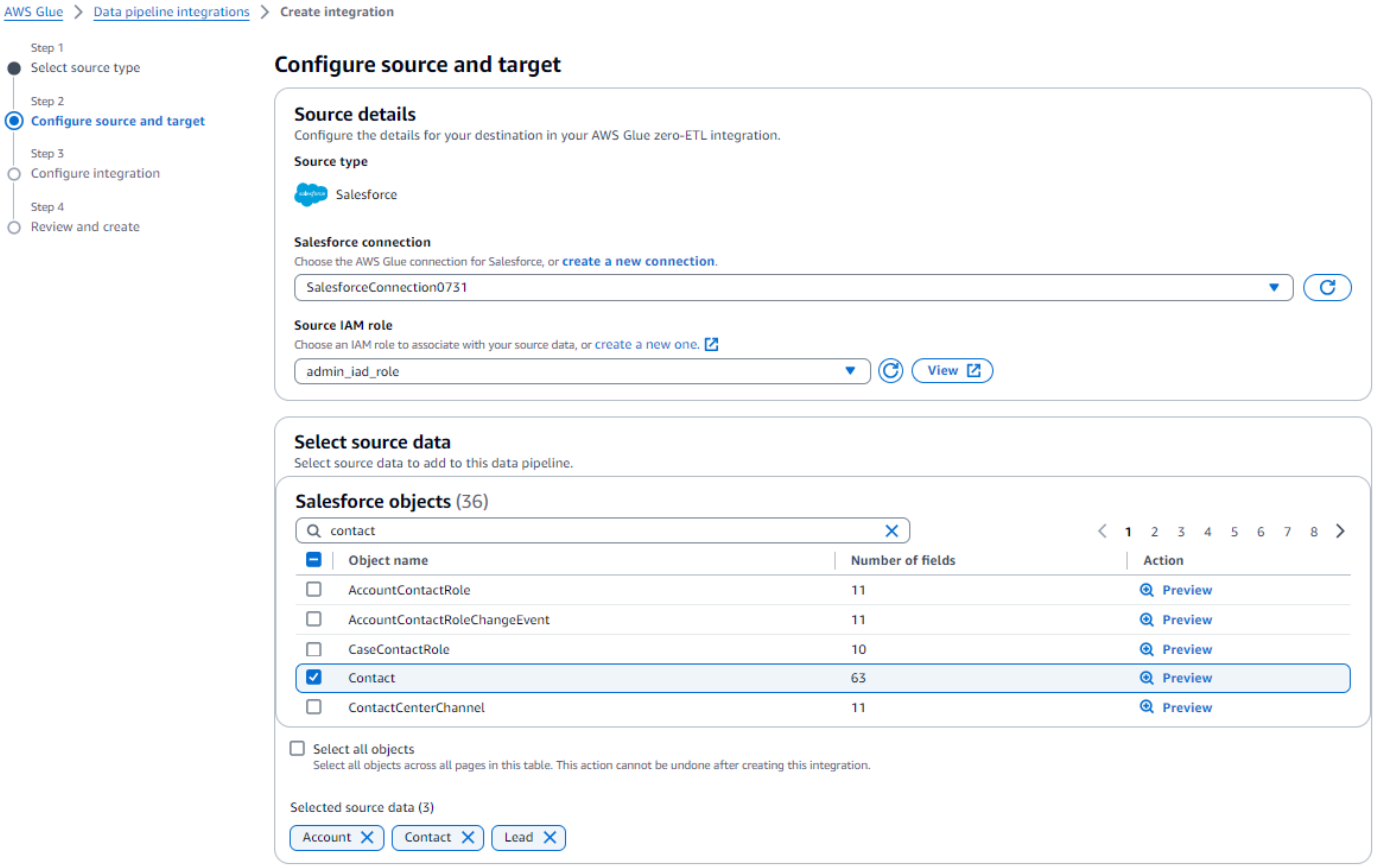
정책이 올바르게 적용되었는지 확인하려면 다음 명령을 사용하여 테이블에 대한 리소스 정책을 가져옵니다.

```
aws dynamodb get-resource-policy \
--resource-arn arn:aws:dynamodb:<region>:<account-id>:table/<ddb-table-name>
--region <region>
```

Salesforce 소스 구성

Salesforce 소스에 대한 연결을 생성하려면 Salesforce에 연결 섹션을 참조하세요.

연결을 생성한 후에는 복제할 소스 데이터를 지정할 수 있습니다.



제로 ETL 통합을 사용하면 지원되는 엔터티에 대해 DDL 작업을 수행할 수 있습니다. 지원되지 않는 엔터티 목록은 Salesforce에서 지원되지 않는 엔터티 및 필드 섹션을 참조하세요.

Salesforce Marketing Cloud Account Engagement 소스 구성

Salesforce Marketing Cloud Account Engagement 소스에 대한 연결을 생성하려면 Salesforce Marketing Cloud Account Engagement에 연결 섹션을 참조하세요.

제로 ETL 통합을 사용하면 지원되는 다음 엔터티에 대해 DDL 작업을 수행할 수 있습니다.

엔터티 레이블	엔터티 이름
캠페인	campaign
나열	list
동적 콘텐츠	dynamic-content
멤버쉽 나열	list-membership
잠재 고객	prospect
User	사용자
EmailTemplate	email-template
EngagementStudioProgram	engagement-studio-program
랜딩 페이지	landing-page
이메일 나열	list-email

추가 Salesforce 구성

Salesforce 제로 ETL에는 Glue 데이터베이스에 대한 Lake Formation 권한이 필요합니다. 그렇지 않으면 로그에서 `IngestionFailed`를 가져오고 다음 오류와 발생합니다.

```
"errorMessage": "Insufficient lake formation permissions on Target Glue database."
```

SAP OData 소스 구성

SAP OData 소스에 대한 연결을 생성하려면 SAP OData에 연결 섹션을 참조하세요.

제로 ETL 통합의 SAP OData 커넥터는 `EntityOf`로 시작하는 엔터티를 지원하지 않습니다.

ServiceNow 소스 구성

ServiceNow 소스에 대한 연결을 생성하려면 에 연결 ServiceNow 섹션을 참조하세요.

Zendesk 소스 구성

Zendesk 소스에 대한 연결을 생성하려면 Zendesk에 연결 섹션을 참조하세요.

제로 ETL 통합을 사용하면 지원되는 엔터티에 대해 다음 DDL 작업을 수행할 수 있습니다.

엔터티 레이블	엔터티 이름	생성 지원	업데이트 지원	삭제 지원
Tickets	tickets	Y	Y	Y
User	사용자	Y	Y	Y
만족도 등급	satisfaction-rating	Y	Y	N
문서	문서	Y	Y	N
Organization	조직	Y	Y	Y
호출	calls	Y	Y	N
콜 레그	legs	Y	Y	N

Zoho CRM 소스 구성

Zoho CRM 소스에 대한 연결을 생성하려면 Zoho CRM에 연결 섹션을 참조하세요.

제로 ETL 통합을 사용하면 지원되는 엔터티에 대해 다음 DDL 작업을 수행할 수 있습니다.

엔터티 레이블	엔터티 이름	DML-Insert 지원	DML-Modify 지원	DML-Delete 지원	DDL-Insert 지원	DDL-Modify 지원	DDL-Delete 지원
리드	lead	Y	Y	Y	Y	Y	Y
Accounts	account	Y	Y	Y	Y	Y	Y
Contacts	contact	Y	Y	Y	Y	Y	Y
Campaigns	campaign	Y	Y	Y	Y	Y	Y

엔터티 레이블	엔터티 이름	DML-Insert 지원	DML-Modify 지원	DML-Delete 지원	DDL-Insert 지원	DDL-Modify 지원	DDL-Delete 지원
업무	task	Y	Y	Y	Y	Y	Y
이벤트	이벤트	Y	Y	Y	Y	Y	Y
호출	call	Y	Y	Y	Y	Y	Y
Solutions	솔루션	Y	Y	Y	Y	Y	Y
Products	product	Y	Y	Y	Y	Y	Y
공급업체	공급업체	Y	Y	Y	Y	Y	Y
견적	quote	Y	Y	Y	Y	Y	Y
판매 주문	sales-order	Y	Y	Y	Y	Y	Y
구매 주문	purchase-order	Y	Y	Y	Y	Y	Y
인보이스	인보이스	Y	Y	Y	Y	Y	Y
Cases	case	Y	Y	Y	Y	Y	Y
Price Books	price-book	Y	Y	Y	Y	Y	Y

Facebook Ads 소스 구성

Facebook Ads 소스에 대한 연결을 생성하려면 Facebook Ads에 연결 섹션을 참조하세요.

제로 ETL 통합을 사용하면 지원되는 엔터티에 대해 다음 DDL 작업을 수행할 수 있습니다.

엔터티 레이블	엔터티 이름	생성 지원	업데이트 지원	삭제 지원
광고 세트	*/adsets	Y	Y	Y

엔터티 레이블	엔터티 이름	생성 지원	업데이트 지원	삭제 지원
캠페인	*/campaigns	Y	Y	Y
광고	*/ads	Y	Y	Y

Instagram Ads 소스 구성

Instagram Ads 소스에 대한 연결을 생성하려면 Instagram Ads에 연결 섹션을 참조하세요.

제로 ETL 통합을 사용하면 지원되는 엔터티에 대해 다음 DDL 작업을 수행할 수 있습니다.

엔터티 이름	생성 지원	업데이트 지원	삭제 지원
*/adsets	Y	Y	Y
*/campaigns	Y	Y	Y
*/ads	Y	Y	Y

Salesforce에서 지원되지 않는 엔터티 및 필드

다음 Salesforce 엔터티 또는 필드는 Salesforce 소스를 사용하는 제로 ETL 통합에 사용할 수 없습니다.

```
AccountChangeEvent, AccountContactRoleChangeEvent, AccountHistory, AccountShare,
 ActiveFeatureLicenseMetric, ActivePermSetLicenseMetric, ActiveProfileMetric,
 ActivityFieldHistory, amzsec__asi_Telemetry_Data_Store__ChangeEvent,
 amzsec__asi_Telemetry_Data_Store__History, amzsec__asi_Telemetry_Data_Store__Share,
 amzsec__asi_Telemetry_Job_Log__ChangeEvent, amzsec__asi_Telemetry_Job_Log__History,
 amzsec__asi_Telemetry_Job_Log__Share, amzsec__asi_Telemetry_Requirement__ChangeEvent,
 amzsec__asi_Telemetry_Requirement__History, amzsec__asi_Telemetry_Requirement__Share,
 ApexClass, ApexComponent, ApexLog, ApexPage, ApexTestQueueItem, ApexTestResult,
 ApexTrigger, AssetChangeEvent, AssetHistory, AssetRelationshipHistory,
 AssetShare, AssignmentRule, AssociatedLocationHistory, AsyncApexJob,
 AuditTrailFileExportShare, AuthorizationFormConsentChangeEvent,
 AuthorizationFormConsentHistory, AuthorizationFormConsentShare,
 AuthorizationFormDataUseHistory, AuthorizationFormDataUseShare,
 AuthorizationFormHistory, AuthorizationFormShare, AuthorizationFormTextHistory,
```

```
AuthProvider, AuthSession, BatchJobHistory, BatchJobPartFailedRecordHistory,
BatchJobPartHistory, BatchJobShare, BrandTemplate, BriefcaseAssignmentChangeEvent,
BriefcaseDefinitionChangeEvent, BusinessBrandShare, BusinessHours,
BusinessProcess, CalcMatrixColumnRangeHistory, CalcProcStepRelationshipHistory,
CalculationMatrixColumnHistory, CalculationMatrixHistory, CalculationMatrixRowHistory,
CalculationMatrixShare, CalculationMatrixVersionHistory, CalculationProcedureHistory,
CalculationProcedureShare, CalculationProcedureStepHistory,
CalculationProcedureVariableHistory, CalculationProcedureVersionHistory, Calendar,
CalendarViewShare, CallCenter, CallCoachConfigModifyEvent, CampaignChangeEvent,
CampaignHistory, CampaignMemberChangeEvent, CampaignMemberStatusChangeEvent,
CampaignShare, CaseChangeEvent, CaseHistory, CaseHistory2 CaseHistory2ChangeEvent,
CaseRelatedIssueChangeEvent, CaseRelatedIssueHistory, CaseShare, CaseStatus,
CaseTeamMember, CaseTeamRole, CaseTeamTemplate, CaseTeamTemplateMember,
CaseTeamTemplateRecord, CategoryNode, ChangeRequestChangeEvent, ChangeRequestHistory,
ChangeRequestRelatedIssueChangeEvent, ChangeRequestRelatedIssueHistory,
ChangeRequestRelatedItemChangeEvent, ChangeRequestRelatedItemHistory,
ChangeRequestShare, ChatRetirementRdyMetrics, ChatterActivity, ClientBrowser,
CollaborationGroup, CollaborationGroupMember, CollaborationGroupMemberRequest,
CollaborationInvitation, CommSubscriptionChannelTypeHistory,
CommSubscriptionChannelTypeShare, CommSubscriptionConsentChangeEvent,
CommSubscriptionConsentHistory, CommSubscriptionConsentShare, CommSubscriptionHistory,
CommSubscriptionShare, CommSubscriptionTimingHistory, Community, ConnectedApplication,
ContactChangeEvent, ContactHistory, ContactPointAddressChangeEvent,
ContactPointAddressHistory, ContactPointAddressShare, ContactPointConsentChangeEvent,
ContactPointConsentHistory, ContactPointConsentShare, ContactPointEmailChangeEvent,
ContactPointEmailHistory, ContactPointEmailShare, ContactPointPhoneChangeEvent,
ContactPointPhoneHistory, ContactPointPhoneShare, ContactPointTypeConsentChangeEvent,
ContactPointTypeConsentHistory, ContactPointTypeConsentShare, ContactRequestShare,
ContactShare, ContentDocumentChangeEvent, ContentDocumentHistory,
ContentDocumentLink, ContentDocumentLinkChangeEvent, ContentDocumentSubscription,
ContentFolderItem, ContentFolderLink, ContentFolderMember, ContentNote,
ContentNotification, ContentTagSubscription, ContentUserSubscription,
ContentVersionChangeEvent, ContentVersionComment, ContentVersionHistory,
ContentVersionRating, ContentWorkspace, ContentWorkspaceMember,
ContentWorkspacePermission, ContentWorkspaceSubscription, ContractChangeEvent,
ContractHistory, ContractLineItemChangeEvent, ContractLineItemHistory,
ContractStatus, Conversation, ConversationParticipant, CronJobDetail,
CronTrigger, CustomBrand, CustomBrandAsset, CustomerShare, CustomHttpHeader,
DashboardComponent, DataUseLegalBasisHistory, DataUseLegalBasisShare,
DataUsePurposeHistory, DataUsePurposeShare, DecisionTableRecordset, DeleteEvent,
DocumentAttachmentMap, Domain, DomainSite, DTRecordsetReplicaShare, EmailBounceEvent,
EmailMessageChangeEvent, EmailServicesAddress, EmailServicesFunction,
EmailTemplate, EmailTemplateChangeEvent, EngagementAttendeeChangeEvent,
EngagementAttendeeHistory, EngagementChannelTypeHistory, EngagementChannelTypeShare,
```

```
EngagementInteractionChangeEvent, EngagementInteractionHistory,
EngagementInteractionShare, EngagementInterface, EngagementTopicChangeEvent,
EngagementTopicHistory, EntitlementChangeEvent, EntitlementHistory,
EntitlementTemplate, EntityMilestoneHistory, EntitySubscription, EventChangeEvent,
EventRelationChangeEvent, EventRelayConfigChangeEvent, ExpressionSetHistory,
ExpressionSetShare, ExpressionSetVersionHistory, ExternalEventMappingShare,
FeedAttachment, FeedLike, FeedPollChoice, FeedPollVote, FeedSignal,
FieldPermissions, FieldSecurityClassification, FiscalYearSettings,
FlowInterviewLogShare, FlowInterviewShare, FlowOrchestrationEvent,
FlowOrchestrationInstanceShare, FlowOrchestrationStageInstanceShare,
FlowOrchestrationStepInstanceShare, FlowOrchestrationWorkItemShare,
FlowRecordShare, FlowRecordVersionChangeEvent, FlowTestResultShare, Folder,
Group, GroupMember, Holiday, IdeaComment, IdpEventLog, ImageHistory, ImageShare,
IncidentChangeEvent, IncidentHistory, IncidentRelatedItemChangeEvent,
IncidentRelatedItemHistory, IncidentShare, IndividualChangeEvent,
IndividualHistory, IndividualShare, KnowledgeableUser, LeadChangeEvent, LeadHistory,
LeadShare, LeadStatus, LightningExitByPageMetrics, LightningToggleMetrics,
LightningUsageByAppTypeMetrics, LightningUsageByBrowserMetrics,
LightningUsageByFlexiPageMetrics, LightningUsageByPageMetrics, ListEmailChangeEvent,
ListEmailShare, ListView, LocationChangeEvent, LocationHistory, LocationShare,
LocationTrustMeasureShare, LoginHistory, LoginIp, MacroChangeEvent,
MacroHistory, MacroInstructionChangeEvent, MacroShare, MacroUsageShare,
ManagedContentVariantChangeEvent, MessagingEndUserHistory, MessagingEndUserShare,
MessagingSessionHistory, MessagingSessionShare, MilestoneType, MLEngagementEvent,
ObjectPermissions, OpportunityChangeEvent, OpportunityContactRoleChangeEvent,
OpportunityFieldHistory, OpportunityLineItemChangeEvent, OpportunityShare,
OpportunityStage, OrderChangeEvent, OrderHistory, OrderItemChangeEvent,
OrderItemHistory, OrderShare, OrderStatus, Organization, OrgEmailAddressSecurity,
OrgWideEmailAddress, OutgoingEmail, OutgoingEmailRelation, PackageLicense,
PartnerRole, PartyConsentChangeEvent, PartyConsentHistory, PartyConsentShare,
Period, PermissionSet, PermissionSetAssignment, PermissionSetTabSetting,
Pricebook2ChangeEvent, Pricebook2History, PricebookEntryChangeEvent,
PricebookEntryHistory, PrivacyJobSessionShare, PrivacyObjectSessionShare,
PrivacyRTBFRequestHistory, PrivacyRTBFRequestShare, PrivacySessionRecordFailureShare,
ProblemChangeEvent, ProblemHistory, ProblemIncidentChangeEvent,
ProblemIncidentHistory, ProblemRelatedItemChangeEvent, ProblemRelatedItemHistory,
ProblemShare, ProcessDefinition, ProcessExceptionEvent, ProcessExceptionShare,
ProcessInstanceChangeEvent, ProcessInstanceStep, ProcessInstanceStepChangeEvent,
ProcessNode, Product2ChangeEvent, Product2History, ProductEntitlementTemplate,
Profile, ProfileSkillEndorsementHistory, ProfileSkillHistory,
ProfileSkillShare, ProfileSkillUserHistory, PromptActionShare, PromptErrorShare,
QueueSobject, QuickTextChangeEvent, QuickTextHistory, QuickTextShare,
QuickTextUsageShare, RecentlyViewed, RecommendationChangeEvent,
RecordActionHistory, RecordAlertHistory, RecordAlertShare, RecordType,
```

```
 ScorecardShare, SellerHistory, SellerShare, ServiceContractChangeEvent,
 ServiceContractHistory, ServiceContractShare, SetupAuditTrail, SetupEntityAccess,
 SharingRecordCollectionShare, Site, SiteHistory, SiteRedirectMapping,
 SocialPersonaHistory, SocialPostChangeEvent, SocialPostHistory, SocialPostShare,
 SolutionHistory, SolutionStatus, StaticResource, StreamingChannelShare,
 TableauHostMappingShare, TaskChangeEvent, TaskPriority, TaskStatus,
 ThreatDetectionFeedback, TimelineObjectDefinitionChangeEvent, TodayGoalShare,
 Topic, TopicUserEvent, Translation, User, UserAppMenuCustomizationShare,
 UserChangeEvent, UserDefinedLabelAssignmentShare, UserDefinedLabelShare,
 UserEmailPreferredPersonShare, UserLicense, UserLogin, UserPackageLicense,
 UserPreference, UserPrioritizedRecordShare, UserProvisioningRequestShare,
 UserRole, UserShare, VideoCallChangeEvent, VideoCallParticipantChangeEvent,
 VideoCallRecordingChangeEvent, VideoCallShare, VisualforceAccessMetrics,
 VoiceCallChangeEvent, VoiceCallRecordingChangeEvent, VoiceCallShare, Vote,
 WebLink, WorkAccessShare, WorkBadgeDefinitionHistory, WorkBadgeDefinitionShare,
 WorkOrderChangeEvent, WorkOrderHistory, WorkOrderLineItemChangeEvent,
 WorkOrderLineItemHistory, WorkOrderLineItemStatus, WorkOrderShare, WorkOrderStatus,
 WorkPlanChangeEvent, WorkPlanHistory, WorkPlanShare, WorkPlanTemplateChangeEvent,
 WorkPlanTemplateEntryChangeEvent, WorkPlanTemplateEntryHistory,
 WorkPlanTemplateHistory, WorkPlanTemplateShare, WorkStepChangeEvent, WorkStepHistory,
 WorkStepStatus, WorkStepTemplateChangeEvent, WorkStepTemplateHistory,
 WorkStepTemplateShare, WorkThanksShare
```

ServiceNow에서 지원되지 않는 엔터티 및 필드

다음 ServiceNow 엔터티 또는 필드는 ServiceNow 소스를 사용하는 제로 ETL 통합에 사용할 수 없습니다.

```
ais_acl_overrides, ais_async_genius_result, ais_async_request, ais_connection,
 ais_genius_result_configuration_parameters, ais_partition_health,
 ais_partition_health_response, ais_publish_history, ais_relevancy_training_execution,
 ais_relevancy_training_staging, ais_search_profile_relevancy_model,
 catalog_draft_entities, clone_log, clone_log0000, clone_log0001, clone_log0002,
 clone_log0003, clone_log0004, clone_log0005, clone_log0006, clone_log0007,
 cmdb_ie_context, cmdb_ie_log, cmdb_ie_run, cmdb_ire_partial_payloads_index,
 cmdb_qb_result_base$par1, cmdb$par1, discovery_log, discovery_log0000,
 discovery_log0001, discovery_log0002, discovery_log0003, discovery_log0004,
 discovery_log0005, discovery_log0006, discovery_log0007, ecc_agent_log,
 entitlement_data, entl_subscription_map, gs_entitlement_plugin_mapping,
 ih_transaction_exclusion, import_log, import_log0000, import_log0001, import_log0002,
 import_log0003, import_log0004, import_log0005, import_log0006, import_log0007,
 jrobin_archive, jrobin_database, jrobin_datasource, jrobin_definition, jrobin_graph,
 jrobin_graph_line, jrobin_graph_set, jrobin_graph_set_member, jrobin_shard,
```

```
jrobin_shard_location, license_role_discovery_run, logger_configuration_validation,
m2m_analytics_event_logger, m2m_user_consent_info, ml_artifact_object_store,
np$sys_gen_ai_filter_sample, np$sys_ui_element, np$sys_ui_list_element,
one_api_service_plan_feature_invocation, one_api_service_plan_invocation,
open_nlu_predict_log, open_nlu_predict_log0000, open_nlu_predict_log0001,
open_nlu_predict_log0002, open_nlu_predict_log0003, open_nlu_predict_log0004,
open_nlu_predict_log0005, open_nlu_predict_log0006, open_nlu_predict_log0007,
pa_diagnostic_log, pa_diagnostic_log0000, pa_diagnostic_log0001,
pa_diagnostic_log0002, pa_diagnostic_log0003, pa_diagnostic_log0004,
pa_diagnostic_log0005, pa_diagnostic_log0006, pa_diagnostic_log0007, pa_favorites,
pa_job_log_rows, pa_job_log_rows0000, pa_job_log_rows0001, pa_job_log_rows0002,
pa_job_log_rows0003, pa_job_log_rows0004, pa_job_log_rows0005, pa_job_log_rows0006,
pa_job_log_rows0007, pa_migration_ignored_scores, pa_scores_l1, pa_scores_l2,
pa_scores_migration_groups, par_dashboard_conversion_backup, promin_log,
promin_log0000, promin_log0001, promin_log0002, promin_log0003, promin_log0004,
promin_log0005, promin_log0006, promin_log0007, promin_request_object,
proposed_change_verification_log, proposed_change_verification_log0000,
proposed_change_verification_log0001, proposed_change_verification_log0002,
proposed_change_verification_log0003, proposed_change_verification_log0004,
proposed_change_verification_log0005, proposed_change_verification_log0006,
proposed_change_verification_log0007, protected_table_log, protected_table_log0000,
protected_table_log0001, protected_table_log0002, protected_table_log0003,
protected_table_log0004, protected_table_log0005, protected_table_log0006,
protected_table_log0007, pwd_history, qb_query_results, scan_log,
scan_log0000, scan_log0001, scan_log0002, scan_log0003, scan_log0004,
scan_log0005, scan_log0006, scan_log0007, schema_validator_error,
sla_repair_log_entry, sla_repair_log_entry0000, sla_repair_log_entry0001,
sla_repair_log_entry0002, sla_repair_log_entry0003, sla_repair_log_entry0004,
sla_repair_log_entry0005, sla_repair_log_entry0006, sla_repair_log_entry0007,
sla_repair_log_message, sla_repair_log_message0000, sla_repair_log_message0001,
sla_repair_log_message0002, sla_repair_log_message0003, sla_repair_log_message0004,
sla_repair_log_message0005, sla_repair_log_message0006, sla_repair_log_message0007,
sn_bm_client_activity, sn_ci_analytics_st_actionable_notifs,
sn_ci_analytics_st_conv_completion_by_cat, sn_ci_analytics_st_conv_dynamic_property,
sn_ci_analytics_st_conversation, sn_ci_analytics_st_count_by_date,
sn_ci_analytics_st_event_occurrence, sn_ci_analytics_st_event_property_value_trend,
sn_ci_analytics_st_issue_auto_resolution, sn_ci_analytics_st_no_clicks,
sn_ci_analytics_st_no_results, sn_ci_analytics_st_property_summary_by_event,
sn_ci_analytics_st_session_count_per_locale, sn_ci_analytics_st_session_duration,
sn_ci_analytics_st_spokes_usage, sn_ci_analytics_st_topic_execution_stats,
sn_ci_analytics_st_topic_occurrence, sn_ci_analytics_st_trending_content,
sn_ci_analytics_st_trending_queries, sn_ci_analytics_st_users,
sn_cs_plugin_signatures, sn_cs_telemetry_log, sn_dfc_application, sn_dfc_product,
sn_employee_position, sn_entitlement_st_subscription_application_family,
```

```
sn_entitlement_st_subscription_application_users, sn_hr_sp_st_relevant_for_you,
sn_instance_clone_log, sn_instance_clone_log0000, sn_instance_clone_log0001,
sn_instance_clone_log0002, sn_instance_clone_log0003, sn_instance_clone_log0004,
sn_instance_clone_log0005, sn_instance_clone_log0006, sn_instance_clone_log0007,
sn_km_mr_st_kb_knowledge, sn_me_st_topic, sn_rf_conditional_definition,
sn_rf_evaluation_type, sn_rf_evaluation_type_input, sn_rf_recommendation_action,
sn_rf_recommendation_experience, sn_rf_recommendation_history,
sn_rf_recommendation_rule, sn_rf_record_display_configuration, sn_rf_trend_definition,
sn_sub_man_st_account_level_entitlement, sn_sub_man_st_gen_ai_metadata,
sn_sub_man_st_instance_used_assist_count, sn_sub_man_st_now_assist_creator_instances,
sn_sub_man_st_now_assists_aggregate, sn_sub_man_st_subscribed_groups,
sn_sub_man_st_subscription_insights, sn_sub_man_st_subscription_license_detail_metric,
sn_sub_man_st_unallocated_group_recommendation, sn_sub_man_st_unconfirmed_user_group,
sn_wn_user_app_activity, sn_wn_user_content_activity, snc_monitorable_item,
snpar_sched_export_v_scheduled_export_visualization, spotlight,
spotlight_audit, spotlight_copy_log_row, spotlight_copy_log_row0000,
spotlight_copy_log_row0001, spotlight_copy_log_row0002, spotlight_copy_log_row0003,
spotlight_copy_log_row0004, spotlight_copy_log_row0005, spotlight_copy_log_row0006,
spotlight_copy_log_row0007, spotlight_job_log_row, spotlight_job_log_row0000,
spotlight_job_log_row0001, spotlight_job_log_row0002, spotlight_job_log_row0003,
spotlight_job_log_row0004, spotlight_job_log_row0005, spotlight_job_log_row0006,
spotlight_job_log_row0007, st_dfc_performance_metric, st_license_detail_metric,
st_on_call_hour, st_sc_wizard_question, st_sys_catalog_items_and_variable_sets,
st_sys_design_system_icon, subscription_instance_stats, svc_container_config,
svc_environment_config, svc_layer_config, svc_model_assoc_ci,
svc_model_checkpoint_attr, svc_model_obj_cluster, svc_model_obj_constraint,
svc_model_obj_deployable, svc_model_obj_element, svc_model_obj_impact,
svc_model_obj_impactrule, svc_model_obj_package, svc_model_obj_path,
svc_model_obj_relation, svc_model_obj_service, sys_administrative_script_transaction,
sys_amb_message, sys_amb_processor, sys_analytics_batch_state,
sys_analytics_config, sys_analytics_data_points_error, sys_analytics_event,
sys_analytics_logger, sys_analytics_logger_field, sys_app_payload_loader_rule,
sys_app_payload_unloader_rule, sys_app_scan_payload, sys_app_scan_variable,
sys_app_scan_variable_type, sys_archive_destroy_log, sys_archive_destroy_run,
sys_archive_log, sys_archive_run, sys_atf_transaction_log, sys_attachment_doc,
sys_attachment_doc_v2, sys_attachment_soft_deleted, sys_audit, sys_audit_relation,
sys_auth_policy_api_allowed, sys_aw_registered_scripting_modal,
sys_cache_flush, sys_data_egress_source, sys_dm_delete_count,
sys_export_set_log, sys_export_set_log0000, sys_export_set_log0001,
sys_export_set_log0002, sys_export_set_log0003, sys_export_set_log0004,
sys_export_set_log0005, sys_export_set_log0006, sys_export_set_log0007,
sys_flow_compiled_flow, sys_flow_compiled_flow_chunk, sys_flow_context_chunk,
sys_flow_context_chunk_archive, sys_flow_context_inputs_chunk,
sys_flow_execution_history, sys_flow_log, sys_flow_log0000, sys_flow_log0001,
```

```
sys_flow_log0002, sys_flow_log0003, sys_flow_plan_context_binding,
sys_flow_report_doc, sys_flow_report_doc_chunk, sys_flow_report_doc_chunk_archive,
sys_flow_runtime_state_chunk, sys_flow_runtime_value_chunk,
sys_flow_subflow_plan_chunk, sys_flow_trigger_plan_chunk, sys_flow_val_listener,
sys_flow_value, sys_flow_value_chunk, sys_gen_ai_config_example,
sys_gen_ai_feature_mapping, sys_gen_ai_strategy_mapping, sys_gen_ai_usage_log,
sys_generative_ai_capability_definition, sys_generative_ai_log,
sys_generative_ai_response_validator, sys_generative_ai_validator,
sys_geo_routing, sys_geo_routing_config, sys_hop_token, sys_hub_action_plan_chunk,
sys_hub_snapshot_chunk, sys_journal_field, sys_journal_field_edit, sys_json_chunk,
sys_kaa_policy, sys_kaa_subidentity_assertion, sys_kaa_user_policy_mapping,
sys_mapplication, sys_mass_encryption_job, sys_notification_execution_log,
sys_notification_execution_log0000, sys_notification_execution_log0001,
sys_notification_execution_log0002, sys_notification_execution_log0003,
sys_notification_execution_log0004, sys_notification_execution_log0005,
sys_notification_execution_log0006, sys_notification_execution_log0007,
sys_nowmq_message, sys_nowmq_provider_param_definition, sys_orchestrator_action,
sys_pd_asset_configuration, sys_pd_context_chunk, sys_pd_context_log,
sys_pd_snapshot_chunk, sys_pd_trigger_license, sys_processing_framework_job,
sys_query_index_hint, sys_query_rewrite, sys_query_string_log,
sys_replication_queue, sys_replication_queue0, sys_replication_queue1,
sys_replication_queue2, sys_replication_queue3, sys_replication_queue4,
sys_replication_queue5, sys_replication_queue6, sys_replication_queue7,
sys_request_performance, sys_rollback_blacklisted, sys_rollback_conflict,
sys_rollback_incremental, sys_rollback_log, sys_rollback_log0000,
sys_rollback_log0001, sys_rollback_log0002, sys_rollback_log0003,
sys_rollback_log0004, sys_rollback_log0005, sys_rollback_log0006,
sys_rollback_log0007, sys_rollback_run, sys_rollback_schema_change,
sys_rollback_schema_conflict, sys_rollback_sequence, sys_scheduler_assignment,
sys_scheduler_memory_pressure_job_log, sys_script_adapter_rule,
sys_script_batch_adapter_rule, sys_search_source_filter, sys_service_authentication,
sys_signing_job, sys_suggestion_reader, sys_sync_history_review, sys_trend,
sys_unreferenced_preview, sys_unreferenced_record_rule, sys_upgrade_manifest,
sys_upgrade_state, sys_ux_asset_cache_buster, sys_ux_lib_component_prop,
sys_ux_lib_presource, sys_ux_page_action, sys_ux_page_action_binding,
sysevent_queue_runtime, syslog, syslog_app_scope0000, syslog_app_scope0001,
syslog_app_scope0002, syslog_app_scope0003, syslog_app_scope0004,
syslog_app_scope0005, syslog_app_scope0006, syslog_app_scope0007, syslog_email0000,
syslog_email0001, syslog_email0002, syslog_email0003, syslog_email0004,
syslog_email0005, syslog_email0006, syslog_email0007, syslog_transaction,
syslog_transaction0000, syslog_transaction0001, syslog_transaction0002,
syslog_transaction0003, syslog_transaction0004, syslog_transaction0005,
syslog_transaction0006, syslog_transaction0007, syslog0000, syslog0001, syslog0002,
syslog0003, syslog0004, syslog0005, syslog0006, syslog0007, ts_attachment,
```

```
ts_c_1_0, ts_c_1_1, ts_c_1_2, ts_c_1_3, ts_c_1_4, ts_c_1_5, ts_c_1_6, ts_c_1_7,
ts_c_1_8, ts_c_1_9, ts_c_10_0, ts_c_10_1, ts_c_10_2, ts_c_10_3, ts_c_10_4,
ts_c_10_5, ts_c_10_6, ts_c_10_7, ts_c_10_8, ts_c_10_9, ts_c_11_0, ts_c_11_1,
ts_c_11_2, ts_c_11_3, ts_c_11_4, ts_c_11_5, ts_c_11_6, ts_c_11_7, ts_c_11_8,
ts_c_11_9, ts_c_2_0, ts_c_2_1, ts_c_2_2, ts_c_2_3, ts_c_2_4, ts_c_2_5, ts_c_2_6,
ts_c_2_7, ts_c_2_8, ts_c_2_9, ts_c_3_0, ts_c_3_1, ts_c_3_2, ts_c_3_3, ts_c_3_4,
ts_c_3_5, ts_c_3_6, ts_c_3_7, ts_c_3_8, ts_c_3_9, ts_c_39_0, ts_c_39_1, ts_c_39_2,
ts_c_39_3, ts_c_39_4, ts_c_39_5, ts_c_39_6, ts_c_39_7, ts_c_39_8, ts_c_39_9,
ts_c_40_0, ts_c_40_1, ts_c_40_2, ts_c_40_3, ts_c_40_4, ts_c_40_5, ts_c_40_6,
ts_c_40_7, ts_c_40_8, ts_c_40_9, ts_c_41_0, ts_c_41_1, ts_c_41_2, ts_c_41_3,
ts_c_41_4, ts_c_41_5, ts_c_41_6, ts_c_41_7, ts_c_41_8, ts_c_41_9, ts_c_42_0,
ts_c_42_1, ts_c_42_2, ts_c_42_3, ts_c_42_4, ts_c_42_5, ts_c_42_6, ts_c_42_7,
ts_c_42_8, ts_c_42_9, ts_c_43_0, ts_c_43_1, ts_c_43_2, ts_c_43_3, ts_c_43_4,
ts_c_43_5, ts_c_43_6, ts_c_43_7, ts_c_43_8, ts_c_43_9, ts_c_44_0, ts_c_44_1,
ts_c_44_2, ts_c_44_3, ts_c_44_4, ts_c_44_5, ts_c_44_6, ts_c_44_7, ts_c_44_8,
ts_c_44_9, ts_c_45_0, ts_c_45_1, ts_c_45_2, ts_c_45_3, ts_c_45_4, ts_c_45_5,
ts_c_45_6, ts_c_45_7, ts_c_45_8, ts_c_45_9, ts_c_6_0, ts_c_6_1, ts_c_6_2,
ts_c_6_3, ts_c_6_4, ts_c_6_5, ts_c_6_6, ts_c_6_7, ts_c_6_8, ts_c_6_9, ts_c_7_0,
ts_c_7_1, ts_c_7_2, ts_c_7_3, ts_c_7_4, ts_c_7_5, ts_c_7_6, ts_c_7_7, ts_c_7_8,
ts_c_7_9, ts_c_8_0, ts_c_8_1, ts_c_8_2, ts_c_8_3, ts_c_8_4, ts_c_8_5, ts_c_8_6,
ts_c_8_7, ts_c_8_8, ts_c_8_9, ts_c_attachment, ts_chain, ts_deleted_doc,
ts_document, ts_field, ts_index_stats, ts_phrase, ts_search_stats, ts_v4_attachment,
ts_word, ua_app_metadata, ua_audit_stats, ua_extra_page, ua_monitor_property,
ua_monitor_property_audit, ua_shared_service, ua_sn_table_inventory, ua_sp_known_bot,
ua_upload_log, v_ais_result_improvement_rule_condition_builder_values,
v_cluster_nodes, v_cxs_search_resource, v_db_index, v_db_trigger, v_file_load_order,
v_interaction_context, v_iostats, v_mysql_proclist, v_mysql_status, v_mysql_variables,
v_on_call_report_cache, v_pa_par_combined_dashboard, v_pd_activity_condition_to_run,
v_pd_activity_start_rule_with_condition, v_pd_lane_condition_to_run,
v_sql_debug, v_st_kb_category, v_st_kb_most_viewed, v_st_kb_recently_viewed,
v_st_km_genai_mra_similar_task, v_st_popular_item, v_st_recent_item, v_st_sc_cat_item,
v_st_sc_catalog, v_st_sc_category, validator_run_summary, vtb_card_history,
wf_log, wf_log0000, wf_log0001, wf_log0002, wf_log0003, wf_log0004, wf_log0005,
wf_log0006, wf_log0007, spotlight_criteria, pa_snapshots, pa_widget_indicators,
pa_widgets, sn_cim_register, global, gsw_change_log, gsw_content, gsw_content_group,
gsw_content_information, gsw_status_of_content, multi_factor_browser_fingerprint,
multi_factor_criteria, pa_filters, password_policy, plan_execution,
plan_mysql, plan_oracle, plan_postgres, sc_rest_api_without_access_policy,
sn_actsub_activity, sn_actsub_activity_fanout, sn_actsub_activity_stream,
sn_actsub_activity_type, sn_actsub_atype_attributes, sn_actsub_atype_notif_pref,
sn_actsub_module, sn_actsub_notif_object, sn_actsub_subobject_stream,
sn_actsub_subscribable_object, sn_actsub_subscription_notif_pref,
sn_actsub_user_stream, sn_appclient_store_outbound_http_quota,
sn_appcreator_app_template, sn_critical_update, sn_docker_spoke_images,
```

```
sn_employee_app, sn_employee_app_access, sn_employee_app_access_criteria,
sn_entitlement_genai_assist_counts, sn_entitlement_genai_creator_user_counts,
sn_entitlement_genai_creator_users, sn_mif_instance, sn_mif_sync_data,
sn_mif_sync_status, sn_mif_table_registration, sn_mif_trust_config,
sn_vsc_best_practice_configurations, sn_vsc_best_practice_goals,
sn_vsc_changed_hardening_settings, sn_vsc_changed_scan_findings,
sn_vsc_check_security_area, sn_vsc_elevation_event, sn_vsc_event,
sn_vsc_export_event, sn_vsc_export_setting, sn_vsc_harc_compliance_status_lookup,
sn_vsc_hardening_compliance_scores, sn_vsc_impersonation_event,
sn_vsc_instance_hardening_settings, sn_vsc_login_event,
sn_vsc_scan_comparisons, sn_vsc_scan_summary, sn_vsc_security_check_categories,
sn_vsc_security_check_configurations, sn_vsc_security_configuration_groups,
sn_vsc_security_privacy_capabilities, sn_vsc_updated_settings,
sn_vsc_user_comparisons, sys_app_hash_inventory, sys_coalesce_strategy_deferred,
sys_flow_secure_data, sys_formula_function, sys_geocoding_request,
sys_global_file_hash, sys_import_set_row, sys_index, sys_index_explain,
sys_installation_schedule, sys_installation_schedule_item, sys_offline_app,
sys_package, sys_package_dependency_item, sys_package_dependency_m2m,
sys_plugins, sys_querystat, sys_reap_package, sys_scoped_plugin,
sys_stage_storage_alias, sys_storage_alias, sys_storage_table_alias, sys_store_app,
sys_table_partition, sys_upgrade_history_log, sys_user_public_credential,
sys_webauthn_authentication_request, sys_webauthn_registration_request,
syslog_app_scope, syslog_email, syslog_page_timing, ua_instance_state_config,
v_expression_cache, v_private_cache, v_shared_cache, sys_amb_message0002,
sys_amb_message0004, sys_amb_message0005, sys_metadata, v_par_unified_report_viz
```

제로 ETL 통합 대상 구성

제로 ETL 통합을 위해 대상을 구성할 때 AWS에서 제공하는 몇 가지 옵션이 있습니다. 대상은 암호화된 Amazon Redshift 데이터 웨어하우스 또는 Amazon SageMaker Lakehouse 카탈로그일 수 있습니다.

제로 ETL 통합의 대상을 선택하기 전에 다음 대상 리소스 중 하나를 구성해야 합니다.

제로 ETL 통합의 대상에 대한 구성 옵션은 다음을 포함합니다.

- 일반 Amazon S3 스토리지로 구성된 Amazon SageMaker Lakehouse 카탈로그 및 데이터베이스. 일반 S3 스토리지를 사용한 Amazon SageMaker Lakehouse 카탈로그 구성을(를) 참조하세요.
- Amazon S3 Tables 버킷으로 구성된 Amazon SageMaker Lakehouse 카탈로그. Amazon S3 Tables를 대상으로 구성을(를) 참조하세요.

- Amazon Redshift 관리형 스토리지로 구성된 Amazon SageMaker Lakehouse 카탈로그. Amazon Redshift 관리형 스토리지로 Amazon SageMaker Lakehouse 카탈로그 구성을(를) 참조하세요.
- Redshift 네임스페이스로 식별되는 Amazon Redshift 데이터 웨어하우스. Amazon Redshift 데이터 웨어하우스 대상 구성을(를) 참조하세요.

> **Note**
>
> 생성 후에는 제로 ETL 통합의 대상을 수정할 수 없습니다.

일반 S3 스토리지를 사용한 Amazon SageMaker Lakehouse 카탈로그 구성

이 섹션에서는 제로 ETL 통합에서 Amazon SageMaker Lakehouse 카탈로그 대상의 스토리지로 일반 Amazon S3 버킷을 구성하는 데 필요한 사전 조건과 설정 단계에 대해 설명합니다.

통합 설정을 위한 사전 조건

일반 S3 스토리지를 사용하여 Amazon SageMaker Lakehouse 카탈로그와 제로 ETL 통합을 생성하기 전에 다음 설정 작업을 완료해야 합니다.

1. AWS Glue 데이터베이스 설정
2. 카탈로그 RBAC 정책 제공
3. 대상 IAM 역할 생성

일반 Amazon S3 스토리지로 Amazon SageMaker Lakehouse 카탈로그를 구성한 후 대상과의 통합 구성(으)로 이동하여 통합 설정을 완료할 수 있습니다.

Amazon S3 Tables를 대상으로 구성

이 섹션에서는 제로 ETL 통합을 위한 대상으로 Amazon S3 Tables를 구성하는 데 필요한 사전 조건과 설정 단계에 대해 설명합니다.

통합 설정을 위한 사전 조건

Amazon S3 Tables를 대상으로 하는 제로 ETL 통합을 생성하기 전에 다음 설정 작업을 완료해야 합니다.

1. Amazon S3 Tables 버킷 설정

2. 카탈로그 RBAC 정책 제공
3. 대상 IAM 역할 생성

Amazon S3 Tables 버킷 설정

1. Amazon S3 Tables 시작하기의 지침에 따라 계정에 S3 Tables 버킷을 생성합니다.
2. Amazon S3 Tables와 AWS 서비스 통합 지침에 따라 S3 Tables 버킷과의 분석 통합을 사용하도록 설정합니다.

카탈로그 RBAC 정책 제공

소스와 Amazon S3 Tables 카탈로그 대상 간의 통합을 허용하려면 카탈로그 RBAC 정책에 다음 권한을 추가해야 합니다.

대상 AWS Glue 카탈로그 리소스 정책에는 AuthorizeInboundIntegration에 대한 Glue 서비스 권한이 포함되어야 합니다. 또한 통합을 생성하는 소스 보안 주체나 대상 AWS Glue 리소스 정책에는 CreateInboundIntegration 권한이 필요합니다.

> **Note**
>
> 교차 계정 시나리오의 경우 소스 주 계정과 대상 AWS Glue 카탈로그 리소스 정책 모두에 리소스에 대한 glue:CreateInboundIntegration 권한이 포함되어야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    // Optional for same account but mandatory for cross account scenarios
    // Allow Alice to create Integration on Target Catalog
      "Principal": {
        "AWS": [
          "arn:aws:iam::<source-account-id>:user/Alice"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:CreateInboundIntegration"
      ],
```

```
      "Resource": [
        "arn:aws:glue:<region>:<Target-Account-Id>:catalog/<s3tablescatalog>/*"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:dynamodb:<region>:<Account>:table/<table-name>"
        }
      }
    },
    { // Required: Allow Glue to Authorize the Inbound Integration on behalf of Bob
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:AuthorizeInboundIntegration"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<Target-Account-Id>:catalog/<s3tablescatalog>/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:dynamodb:<region>:<account-id>:table/<table-name>"
        }
      }
    }
  ]
}
```

> **Note**
>
> <s3tablescatalog>를 S3 Tables의 카탈로그 이름으로 바꿉니다.

대상 IAM 역할 생성

다음과 같은 권한 및 신뢰 관계를 가진 대상 IAM 역할을 생성합니다.

IAM 정책 예제:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3tables:ListTableBuckets",
                "s3tables:GetTableBucket",
                "s3tables:GetTableBucketEncryption",
                "s3tables:GetNamespace",
                "s3tables:CreateNamespace",
                "s3tables:ListNamespaces",
                "s3tables:CreateTable",
                "s3tables:GetTable",
                "s3tables:GetTableEncryption",
                "s3tables:ListTables",
                "s3tables:GetTableMetadataLocation",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:GetTableData",
                "s3tables:PutTableData"
            ],
            "Resource": "arn:aws:s3tables:<region>:<account-id>:bucket/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "AWS/Glue/ZeroETL"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
```

```
}
```

대상 IAM 역할에 다음 신뢰 정책을 추가하여 AWS Glue 서비스가 역할을 수임할 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> S3 Tables 버킷 리소스 정책에 이 대상 IAM 역할에 대한 명시적인 거부 문이 없는지 확인합니다. 명시적으로 거부하면 모든 허용 권한이 무시되고 통합이 제대로 작동하지 않습니다.

Amazon Redshift 관리형 스토리지로 Amazon SageMaker Lakehouse 카탈로그 구성

이 섹션에서는 Amazon Redshift 관리형 스토리지(RMS)를 제로 ETL 통합의 대상으로 사용하여 Amazon SageMaker Lakehouse 카탈로그를 구성하기 위한 사전 조건 및 설정 단계에 대해 설명합니다.

통합 설정을 위한 사전 조건

Redshift 관리형 스토리지를 사용하여 Amazon SageMaker Lakehouse 카탈로그와의 제로 ETL 통합을 생성하기 전에 다음 설정 작업을 완료해야 합니다.

1. Amazon Redshift 클러스터 또는 서버리스 작업 그룹 설정
2. Lake Formation과의 Amazon Redshift 통합 등록
3. Lake Formation에서 관리형 카탈로그 생성

4. IAM 권한 구성

Amazon Redshift 관리형 스토리지 설정

제로 ETL 통합을 위한 Amazon Redshift 관리형 스토리지를 설정하려면 다음 단계를 따르세요.

1. 기존 Amazon Redshift 클러스터 또는 Serverless 작업 그룹을 생성하거나 사용합니다. 통합을 성공적으로 수행하려면 대상 Amazon Redshift 작업 그룹이나 클러스터에서 `enable_case_sensitive_identifier` 파라미터가 켜져 있는지 확인합니다. 대소문자 구분 활성화에 대한 자세한 내용은 Amazon Redshift 관리 안내서의 데이터 웨어하우스에 대소문자 구분 기능 사용 설정을 참조하세요.
2. Redshift에서 AWS Lake Formation의 카탈로그로 통합을 등록합니다. AWS Glue 데이터 카탈로그에 클러스터 및 네임스페이스 등록을 참조하세요.
3. AWS Lake Formation에서 페더레이션 또는 관리형 카탈로그를 생성합니다. 자세한 내용은 다음을 참조하세요.
   - AWS Glue 데이터 카탈로그로 Amazon Redshift 데이터 가져오기
   - AWS Glue 데이터 카탈로그에서 Amazon Redshift 관리형 카탈로그 생성
4. 대상 역할에 대한 IAM 권한을 구성합니다. 이 역할에는 Redshift 및 Lake Formation 리소스 모두에 액세스할 수 있는 권한이 필요합니다. 이 역할에는 적어도 다음 사항이 있어야 합니다.
   - Redshift 클러스터 또는 작업 그룹에 액세스할 수 있는 권한
   - Lake Formation 카탈로그에 액세스할 수 있는 권한
   - 카탈로그에서 테이블을 생성하고 관리할 수 있는 권한
   - 모니터링을 위한 CloudWatch 및 CloudWatch Logs 권한

Amazon Redshift 관리형 스토리지로 Amazon SageMaker Lakehouse 카탈로그를 구성한 후 대상과의 통합 구성(으)로 이동하여 통합 설정을 완료할 수 있습니다.

Amazon Redshift 데이터 웨어하우스 대상 구성

이 섹션에서는 제로 ETL 통합을 위한 대상으로 Amazon Redshift 데이터 웨어하우스를 구성하는 데 필요한 사전 조건과 설정 단계에 대해 설명합니다.

통합 설정을 위한 사전 조건

Amazon Redshift 데이터 웨어하우스 대상과 제로 ETL 통합을 생성하기 전에 다음 설정 작업을 완료해야 합니다.

1. Amazon Redshift 클러스터 또는 서버리스 작업 그룹 설정
2. 대/소문자 구분 구성
3. IAM 권한 구성

Amazon Redshift 데이터 웨어하우스 설정

제로 ETL 통합을 위한 Amazon Redshift 데이터 웨어하우스를 설정하려면 다음 단계를 따르세요.

1. Amazon Redshift 콘솔로 이동하여 클러스터 생성을 클릭하거나 기존 클러스터를 사용합니다. Amazon Redshift Serverless의 경우 작업 그룹 생성을 클릭합니다.
2. 새 클러스터를 생성하는 경우 적절한 클러스터 크기를 선택하고 클러스터가 암호화되어 있는지 확인합니다. 서버리스의 경우 요구 사항에 따라 작업 그룹 설정을 구성합니다.
3. 통합을 성공적으로 수행하려면 대상 Amazon Redshift 작업 그룹이나 클러스터에서 `enable_case_sensitive_identifier` 파라미터가 켜져 있는지 확인합니다. 대소문자 구분 활성화에 대한 자세한 내용은 Amazon Redshift 관리 안내서의 데이터 웨어하우스에 대소문자 구분 기능 사용 설정을 참조하세요.
4. 제로 ETL 통합이 Amazon Redshift 데이터 웨어하우스에 액세스할 수 있도록 IAM 권한을 구성합니다. 다음 권한이 있는 IAM 역할을 생성해야 합니다.
   - Amazon Redshift 클러스터 또는 작업 그룹에 액세스할 수 있는 권한
   - Amazon Redshift에서 데이터베이스 및 테이블을 생성하고 관리할 수 있는 권한
   - 모니터링을 위한 CloudWatch 및 Amazon CloudWatch Logs 권한
5. Amazon Redshift 작업 그룹 또는 클러스터 설정이 완료되면 제로 ETL 통합을 위해 데이터 웨어하우스를 구성해야 합니다. 자세한 내용은 Amazon Redshift 관리 안내서의 제로 ETL 통합 시작하기를 참조하세요.

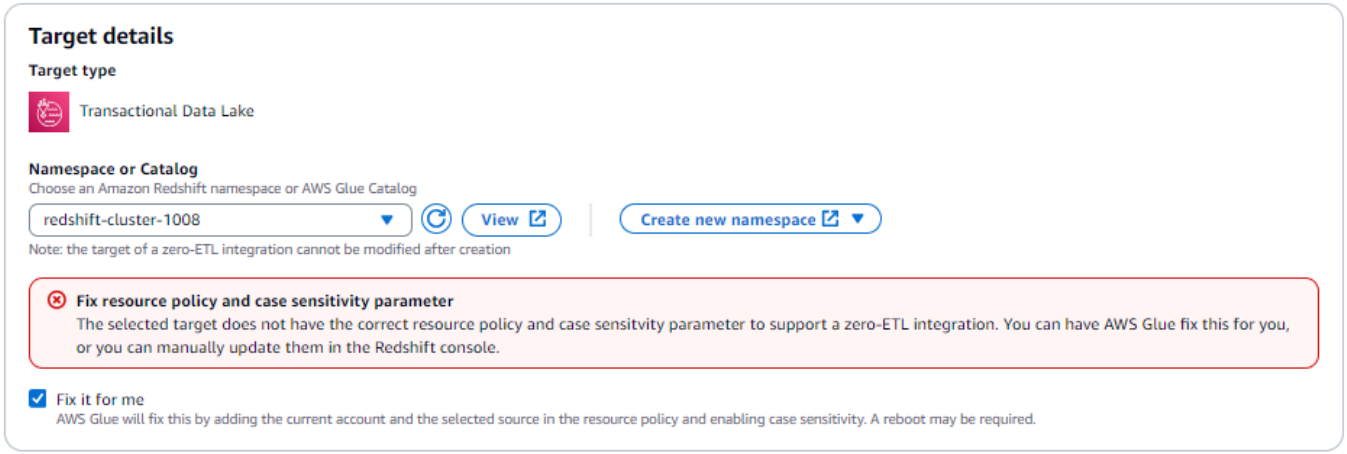
> **Note**
>
> Amazon Redshift 데이터 웨어하우스를 대상으로 사용하는 경우 통합은 지정된 데이터베이스에 복제된 데이터를 저장하기 위한 스키마를 생성합니다. 스키마 이름은 통합 이름에서 파생됩니다.

Amazon Redshift 데이터 웨어하우스를 구성한 후 대상과의 통합 구성(으)로 이동하여 통합 설정을 완료할 수 있습니다.

대상과의 통합 구성

대상 리소스를 구성하고, 연결을 선택하고, 소스 IAM 역할을 지정한 후, 다음 단계에 따라 통합 설정을 완료합니다.

1. 이전 단계에서 구성한 대상을 지정합니다.
2. AWS Glue 수정 요청 옵션을 선택합니다. Amazon Redshift 대상의 경우 다음과 같은 결과로 이어집니다.
   - Amazon Redshift 클러스터 또는 서버리스 작업 그룹에 권한이 있는 서비스 보안 주체를 적용합니다.
   - Amazon Redshift 클러스터 또는 서버리스 작업 그룹에 권한이 있는 AWS Glue 소스 ARN을 적용합니다.
   - `enable_case_sensitive_identifier = true`를 사용하여 새 파라미터 그룹을 연결합니다.



3. 통합 이름을 제공하고 통합 생성 및 시작을 선택합니다.
4. 통합이 활성 상태가 되면 통합 세부 정보 페이지로 이동하여 통합에서 데이터베이스 생성을 선택합니다.
5. 마지막으로 Redshift 쿼리 편집기로 이동한 다음 데이터베이스에 연결하여 스냅샷 및 증분 데이터를 검증할 수 있습니다.

> **Note**
>
> 네임스페이스 또는 카탈로그 이름에는 소문자 영숫자와 밑줄만 사용할 수 있습니다. 이는 AWS Glue Data Catalog에서 특수 문자를 포함한 원하는 이름을 사용하여 데이터베이스를 생성할 수 있도록 허용하는 것과 다릅니다.

파티션 사양 및 스키마 중첩 해제 안내서

DynamoDB 및 SaaS 애플리케이션과 같은 NoSQL 데이터 소스로 작업하는 경우 데이터는 종종 분석에 고유한 문제를 초래합니다.

1. 동일한 테이블 내의 레코드에 다른 스키마가 있을 수 있음
2. 동일한 테이블 내에 중첩된 레코드가 다르게 표현될 수 있음
3. 맵 및 배열과 같은 복잡한 중첩 구조에 효율적인 쿼리를 위한 변환이 필요함
4. 규모에 맞는 쿼리 성능을 보장하려면 최적의 데이터 구성이 필요함

AWS Glue 제로 ETL 통합은 두 가지 강력한 기능을 통해 이러한 문제를 해결합니다.

- 스키마 중첩 해제: 복잡한 중첩된 데이터 구조를 분석하기 적합한 형식으로 자동으로 평면화하고 데이터 구조 보존과 쿼리 단순성 최적화 사이의 균형을 맞추기 위해 구성 가능한 수준의 중첩 해제 기능을 제공합니다.
- 데이터 파티셔닝: 지정된 열 또는 시간 기반 차원을 기반으로 데이터를 논리적 파티션으로 구성하여 쿼리 실행 중에 파티션 정리를 활성화함으로써 쿼리 성능을 개선하고 비용을 절감합니다.

이러한 데이터 소스를 효과적으로 쿼리하기 위해 AWS Glue 제로 ETL은 대상 AWS Glue 데이터베이스에서 복제되는 소스 데이터에 대한 기본 스키마 처리 및 파티셔닝 스키마를 제공합니다. CreateIntegrationTableProperty API를 통해 각 테이블에 대한 스키마 중첩 해제 및 파티셔닝 설정을 구성하여 분석 워크로드에 맞게 데이터를 구조화하고 구성하는 방법을 세부적으로 제어할 수 있습니다.

기본 중첩 해제 및 파티셔닝 동작

1. 대상 테이블에 대한 중첩 해제 옵션이 제공되지 않은 경우 AWS Glue 제로 ETL은 기본적으로 전체 중첩 해제로 설정됩니다.
2. 대상 테이블에 PartitionSpec이 제공되지 않은 경우 AWS Glue 제로 ETL은 기본적으로 버킷 파티셔닝으로 설정됩니다.

스키마 중첩 해제

제로 ETL을 통해 분석 서비스와 통합할 때 대상 테이블에 중첩 구조가 표시되는 방식을 선택할 수 있습니다. AWS Glue 제로 ETL은 복잡한 데이터 구조를 분석에 더 적합한 형식으로 평면화하기 위해 스키마 중첩 해제 옵션을 제공합니다.

중첩 해제 옵션

소스와 제로 ETL 통합을 생성할 때 다음의 중첩 해제 옵션 중에서 선택할 수 있습니다. 이러한 옵션은 CreateIntegrationTableProperty API를 호출할 때 사용하는 특정 열거 값에 해당합니다.

중첩 해제 없음(기본값)

API 값: **`NO_UNNEST`**

DynamoDB 항목의 원래 중첩 구조를 보존합니다. 맵 및 목록은 대상에 구조화된 열로 저장됩니다.

최적의 용도: 분석 도구가 중첩된 데이터를 처리할 수 있는 경우 DynamoDB 데이터의 정확한 구조를 보존하는 데 가장 적합합니다.

단일 수준의 중첩 해제

API 값: **`TOP_LEVEL`**

중첩 맵의 최상위 수준을 개별 열로 평면화합니다. 목록 구조는 중첩 상태로 유지됩니다.

최적의 용도: DynamoDB 항목에 일관된 스키마가 있는 경우 데이터 구조의 보존과 쿼리 단순성 사이의 균형을 맞추는 데 가장 적합합니다.

모든 수준의 중첩 해제

API 값: **`FULL`**

모든 중첩 구조(맵 및 목록)를 점 표기법을 사용하여 개별 열로 재귀적으로 평면화하고 이름을 지정합니다.

최적의 용도: 심층 중첩 구조와 평평한 스키마를 선호하는 분석 도구와 함께 작업하는 경우 쿼리 단순화를 극대화하는 데 가장 적합합니다.

> **Note**
>
> DynamoDB 데이터에 변동 가능 또는 심층 중첩 구조가 있다면 완전한 중첩 해제로 인해 매우 넓은 테이블과 많은 열이 생성될 수 있습니다.

Example API에서 중첩 해제 옵션 사용

CreateIntegrationTableProperty API를 통해 스키마 중첩 해제를 구성할 때 UnnestSpec 파라미터에서 중첩 해제 옵션을 지정합니다.

```
aws glue create-integration-table-property \
  --resource-arn "arn:aws:glue:us-east-1:123456789012:integration/my-integration" \
  --table-name "my-table" \
  --table-properties '{
    "unnestSpec": "FULL"
  }'
```

중첩 해제 예제

다음 구조가 있는 DynamoDB 항목을 고려하세요.

```
{
  "ProductId": "P12345",
  "ProductDetails": {
    "Name": "Smartphone",
    "Brand": "TechCo",
    "Specifications": {
      "Color": "Black",
      "Storage": "128GB"
    }
  },
  "Reviews": [
    {
      "Rating": 5,
      "Comment": "Great product!"
    },
    {
      "Rating": 4,
      "Comment": "Good value."
    }
  ]
}
```

중첩 해제 없음 예제

중첩 해제를 수행하지 않으면 대상 테이블에는 다음과 같은 열이 포함됩니다.

- ProductId(문자열)

- ProductDetails(구조체)
- Reviews(구조체 배열)

쿼리는 구조체와 배열 액세스 패턴을 사용해야 합니다.

```
SELECT
  ProductId,
  ProductDetails.Name,
  ProductDetails.Specifications.Color
FROM product_table;
```

단일 수준 중첩 해제 예제

단일 수준의 중첩을 해제하면 대상 테이블에는 다음과 같은 열이 포함됩니다.

- ProductId(문자열)
- ProductDetails_Name(문자열)
- ProductDetails_Brand(문자열)
- ProductDetails_Specifications(구조체)
- Reviews(구조체 배열)

첫 번째 수준에서는 쿼리가 간소화됩니다.

```
SELECT
  ProductId,
  ProductDetails_Name,
  ProductDetails_Specifications.Color
FROM product_table;
```

모든 수준 중첩 해제 예제

모든 수준의 중첩을 해제하면 대상 테이블에는 다음과 같은 열이 포함됩니다.

Column Name(유형)	값
ProductId(문자열)	P12345
ProductDetails_Name(문자열)	스마트폰

Column Name(유형)	값
ProductDetails_Brand(문자열)	TechCo
ProductDetails_Specifications_Color(문자열)	검은색
ProductDetails_Specifications_Storage(문자열)	128GB
Reviews_0_Rating(숫자)	5
Reviews_0_Comment(문자열)	훌륭한 결과물입니다.
Reviews_1_Rating(숫자)	4
Reviews_1_Comment(문자열)	좋은 값입니다.

쿼리는 완전히 평면화됩니다.

```
SELECT
  ProductId,
  ProductDetails_Name,
  ProductDetails_Specifications_Color
FROM product_table;
```

제로 ETL 통합을 위한 데이터 파티셔닝

데이터 파티셔닝이란 무엇인가요?

데이터 파티셔닝은 대규모 데이터세트를 파티션이라고 하는 더 작고 관리하기 쉬운 세그먼트로 나누는 기법입니다. AWS Glue 제로 ETL 통합의 컨텍스트에서 파티셔닝은 특정 열 값 또는 해당 값의 변환을 기반으로 대상 위치에서 데이터를 구성합니다.

데이터 파티셔닝의 이점

효과적인 데이터 파티셔닝은 분석 워크로드에 몇 가지 주요 이점을 제공합니다.

- 쿼리 성능 향상: 쿼리는 관련 없는 파티션을 건너뛸 수 있으므로(파티션 정리) 스캔해야 하는 데이터의 양이 줄어듭니다.
- 비용 절감: 스캔하는 데이터가 줄어들면 분석 쿼리에 대한 컴퓨팅 및 I/O 비용을 절감할 수 있습니다.

- 확장성 향상: 파티셔닝을 통해 데이터 세그먼트를 병렬 처리할 수 있으므로 분석 워크로드를 더 효율적으로 확장할 수 있습니다.
- 간소화된 데이터 수명 주기 관리: 파티션 수준에서 보존 정책을 관리할 수 있으므로 이전 데이터를 더 간편하게 보관하거나 삭제할 수 있습니다.

주요 파티셔닝 개념

파티션 열

레코드가 파티션으로 구성되는 방식을 결정하는 데 사용되는 데이터의 열입니다. 효과적인 파티션 열은 일반적인 쿼리 패턴과 일치해야 하며 적절한 카디널리티가 있어야 합니다.

파티션 함수

파티션 열 값에 변환을 적용하여 실제 파티션 경계를 생성합니다. 예를 들어 자격 증명(원시 값 사용), 시간 기반 함수(연도, 월, 일, 시간), 버킷팅이 있습니다.

파티션 정리

쿼리 엔진이 쿼리와 관련된 데이터가 포함되지 않은 파티션을 식별하고 건너뛰어 성능을 개선하는 프로세스입니다.

파티션 세부 수준

데이터가 파티셔닝되는 세부 수준입니다. 더 세밀한 세분화(파티션 수가 많을수록)는 쿼리 성능을 개선할 수 있지만 메타데이터 오버헤드가 증가할 수 있습니다. 덜 세밀한(파티션 수가 적을수록) 메타데이터 오버헤드는 줄어들지만 필요 이상으로 많은 데이터를 스캔하게 될 수 있습니다.

AWS Glue 제로 ETL 통합의 파티셔닝

AWS Glue 제로 ETL 통합은 고급 파티셔닝 기능을 제공하는 Apache Iceberg 테이블 형식을 사용합니다. 제로 ETL 통합을 생성하면 다음을 수행할 수 있습니다.

- 데이터 소스에 최적화된 기본 파티셔닝 전략 사용
- 쿼리 패턴에 맞는 사용자 지정 파티셔닝 사양 정의
- 파티션 열에 변환 적용(타임스탬프 기반 파티셔닝에 특히 유용)
- 다중 수준 파티셔닝을 위한 여러 파티션 전략 결합

파티셔닝 구성은 제로 ETL 통합을 설정할 때 `CreateIntegrationTableProperty` API를 통해 지정됩니다. 구성이 완료되면 AWS Glue는 이러한 파티셔닝 전략을 자동으로 적용하여 대상 위치에서 데이터를 구성합니다.

파티셔닝 사양 API 참조

CreateIntegrationTableProperties API에서 다음 파라미터를 사용하여 파티셔닝을 구성합니다.

PartitionSpec

대상 위치에서 데이터를 파티셔닝하는 방법을 정의하는 파티션 사양의 배열입니다.

```
{
  "partitionSpec": [
    {
      "fieldName": "timestamp_col",
      "functionSpec": "month",
      "conversionSpec": "epoch_milli"
    },
    {
      "fieldName": "category",
      "functionSpec": "identity"
    },
    {
      "fieldName": "user_id",
      "functionSpec": "bucket",
      "bucketCount": 16
    },
    {
      "fieldName": "product_code",
      "functionSpec": "truncate",
      "width": 4
    }
  ]
}
```

FieldName

파티셔닝에 사용할 열 이름을 지정하는 UTF-8 문자열(1~128바이트)입니다.

FunctionSpec

파티셔닝 함수를 지정합니다. 유효한 값:

- identity - 변환 없이 소스 값을 직접 사용
- year - 타임스탬프 값에서 연도 추출(예: 2023)
- month - 타임스탬프 값에서 월 추출(예: 2023-01)
- day - 타임스탬프 값에서 날짜 추출(예: 2023-01-15)
- hour - 타임스탬프 값에서 시간 추출(예: 2023-01-15-14)
- bucket - 해시 기반 파티셔닝을 사용하여 지정된 수의 버킷에 값 배포. bucketCount 파라미터가 필요합니다.
- truncate - 문자열 값을 지정된 너비로 자름. width 파라미터가 필요합니다.

> **Note**
>
> 시간 기반 함수(year, month, day, hour)는 소스 타임스탬프 형식을 지정하는 데 ConversionSpec 파라미터가 필요합니다. bucket 및 truncate 함수는 설명에 명시된 대로 추가 파라미터가 필요합니다.

ConversionSpec

소스 데이터의 타임스탬프 형식을 지정하는 UTF-8 문자열입니다. 유효한 값은 다음과 같습니다.

- epoch_sec - 초 단위의 Unix 에포크 타임스탬프
- epoch_milli - 밀리초 단위의 Unix 에포크 타임스탬프
- iso - ISO 8601 형식의 타임스탬프

BucketCount

bucket 함수를 사용할 때 생성할 버킷의 수를 지정하는 정수입니다. functionSpec이 bucket으로 설정되는 경우 이 파라미터가 필요합니다. 값은 양의 정수여야 합니다. 해시 기반 버킷팅은 지정된 수의 버킷에 값을 균등하게 배포합니다.

Example 버킷 파티셔닝 예제

```
{
  "partitionSpec": [
    {
      "fieldName": "user_id",
      "functionSpec": "bucket",
      "bucketCount": 16
    }
```

```
  ]
}
```

너비

`truncate` 함수를 사용할 때 자르기 너비를 지정하는 정수입니다. `functionSpec`이 `truncate`로 설정되는 경우 이 파라미터가 필요합니다. 이 값은 문자열 값의 시작 부분부터 유지할 문자 수를 나타내는 양의 정수여야 합니다.

Example 자르기 파티셔닝 예제

```
{
  "partitionSpec": [
    {
      "fieldName": "product_code",
      "functionSpec": "truncate",
      "width": 4
    }
  ]
}
```

파티셔닝 전략

기본 파티셔닝

파티션 열을 지정하지 않으면 AWS Glue 제로 ETL은 데이터 소스에 최적화된 기본 파티셔닝 전략을 적용합니다.

- 프라이머리 키 기반 파티셔닝: 프라이머리 키가 있는 소스(예: DynamoDB 테이블)의 경우 AWS Glue 제로 ETL은 파티션 폭발을 방지하기 위해 버킷팅이 있는 프라이머리 키를 사용하여 데이터를 자동으로 파티셔닝합니다.

기본 파티셔닝은 수동으로 구성할 필요 없이 일반적인 쿼리 패턴에 잘 작동하도록 설계되었습니다. 그러나 특정 쿼리 패턴 또는 성능 요구 사항의 경우 사용자 지정 파티셔닝 전략을 정의할 수 있습니다.

사용자 정의 파티셔닝 전략

AWS Glue 제로 ETL을 사용하면 `PartitionSpec` 파라미터를 사용하여 사용자 지정 파티셔닝 전략을 정의할 수 있습니다. 하나 이상의 파티션 열을 지정하고 각 열에 서로 다른 파티셔닝 함수를 적용할 수 있습니다.

자격 증명 파티셔닝은 열의 원시 값을 사용하여 파티션을 생성합니다. 이 전략은 범주, 리전 또는 상태 필드와 같이 카디널리티가 낮거나 중간 정도인 열에 유용합니다.

Example 자격 증명 파티셔닝 예제

```
{
  "partitionSpec": [
    {
      "fieldName": "category",
      "functionSpec": "identity"
    }
  ]
}
```

이렇게 하면 "category" 열의 각 고유 값에 대해 별도의 파티션이 생성됩니다.

> **Warning**
>
> 카디널리티가 높은 열(예: 프라이머리 키 또는 타임스탬프)에서 자격 증명 파티셔닝을 사용하면 파티션 폭발로 인해 성능이 저하되고 메타데이터 오버헤드가 증가할 수 있으므로 자격 증명 파티셔닝을 사용하지 마세요.

시간 기반 파티셔닝은 다양한 세부 수준(연도, 월, 일 또는 시간)의 타임스탬프 값을 기반으로 데이터를 구성합니다. 이 전략은 시계열 데이터에 이상적이며 효율적인 시간 범위 쿼리를 지원합니다.

시간 기반 파티셔닝을 사용하는 경우 AWS Glue 제로 ETL은 파티션 함수를 적용하기 전에 다양한 타임스탬프 형식을 표준화된 형식으로 자동 변환할 수 있습니다. 이 변환은 `ConversionSpec` 파라미터를 사용하여 지정됩니다.

Example 시간 기반 파티셔닝 예제

```
{
  "partitionSpec": [
    {
      "fieldName": "created_at",
      "functionSpec": "month",
      "conversionSpec": "epoch_milli"
    }
  ]
```

```
}
```

이렇게 하면 Unix 에포크 타임스탬프를 포함하는 "created_at" 열을 기반으로 데이터를 월별로 파티셔닝합니다.

AWS Glue 제로 ETL은 다음과 같은 시간 기반 파티션 함수를 지원합니다.

- 연도: 데이터를 연도별로 파티셔닝(예: 2023, 2024)
- 월: 데이터를 월별로 파티셔닝(예: 2023-01, 2023-02)
- 일: 데이터를 일별로 파티셔닝(예: 2023-01-01, 2023-01-02)
- 시간: 데이터를 시간별로 파티셔닝(예: 2023-01-01-01, 2023-01-01-02)

AWS Glue 제로 ETL은 `ConversionSpec` 파라미터를 통해 다음과 같은 타임스탬프 형식을 지원합니다.

- epoch_sec: Unix 에포크 타임스탬프(초 단위)
- epoch_milli: Unix 에포크 타임스탬프(밀리초 단위)
- iso: ISO 8601 형식의 타임스탬프

> **Note**
>
> 원본 열 값은 소스 데이터에서 변경되지 않습니다. AWS Glue는 대상 데이터베이스 테이블에서 파티션 열 값만 타임스탬프 유형으로 변환합니다. 변환은 파티셔닝 프로세스에만 적용됩니다.

버킷 파티셔닝은 해시 기반 배포를 사용하여 고정된 수의 버킷에 데이터를 균등하게 분산합니다. 이 전략은 파티션 수를 제어하고 균일한 데이터 배포를 보장하려는 카디널리티가 높은 열에 유용합니다.

Example 버킷 파티셔닝 예제

```
{
  "partitionSpec": [
    {
      "fieldName": "user_id",
      "functionSpec": "bucket",
      "bucketCount": 16
```

```
    }
  ]
}
```

이렇게 하면 "user_id" 열의 해시 값을 기반으로 16개의 버킷이 생성되어 실제 사용자 ID 값에 관계없이 균등하게 배포됩니다.

버킷 파티셔닝은 다음과 같은 경우에 특히 효과적입니다.

- 카디널리티가 높은 열(예: 사용자 ID, 세션 ID 또는 프라이머리 키)
- 예측 가능한 파티션 수가 필요한 시나리오
- 고정된 파티션 경계에서 병렬 처리를 통해 이점을 얻을 수 있는 워크로드

자르기 파티셔닝은 잘린 문자열 값을 기반으로 파티션을 생성하며 첫 번째 N개의 문자만 유지합니다. 이 전략은 유사한 값을 함께 그룹화하려는 문자열 열에 유용합니다.

Example 자르기 파티셔닝 예제

```
{
  "partitionSpec": [
    {
      "fieldName": "product_code",
      "functionSpec": "truncate",
      "width": 4
    }
  ]
}
```

이렇게 하면 "product_code" 열의 처음 4자를 기반으로 파티션이 생성됩니다. 예를 들어 "ELEC12345" 및 "ELEC67890"은 동일한 "ELEC" 파티션에 그룹화됩니다.

자르기 파티셔닝은 다음 경우에 특히 유용합니다.

- 제품 코드, SKU 또는 의미 있는 접두사가 포함된 기타 구조화된 식별자
- 지역 그룹화가 유용한 지역 코드 또는 우편번호
- 접두사 기반 그룹화가 쿼리 패턴과 일치하는 모든 문자열 열

다중 수준 파티셔닝은 여러 파티션 전략을 결합하여 계층 기반 파티셔닝 체계를 생성합니다. 이는 동일한 데이터세트에 대해 서로 다른 유형의 쿼리를 최적화하는 데 유용합니다.

Example 다중 수준 파티셔닝 예제

```
{
  "partitionSpec": [
    {
      "fieldName": "created_at",
      "functionSpec": "month",
      "conversionSpec": "iso"
    },
    {
      "fieldName": "region",
      "functionSpec": "identity"
    },
    {
      "fieldName": "user_id",
      "functionSpec": "bucket",
      "bucketCount": 8
    }
  ]
}
```

이렇게 하면 먼저 ("created_at" 열에서) 월별, 지역별, 마지막으로 사용자 ID 버킷별로 3단계 파티셔닝 체계가 생성됩니다. 이를 통해 날짜 범위, 특정 리전, 사용자 세그먼트 또는 이러한 차원의 조합을 기준으로 필터링하는 효율적인 쿼리가 가능합니다.

다중 수준 파티셔닝 체계를 설계할 때는 다음 사항을 고려하세요.

- 파티션 계층 구조에서 선택성이 높은 열을 먼저 배치
- 파티션 세부 수준과 파티션 수 간 균형 맞추기
- 파티셔닝 체계를 가장 일반적인 쿼리 패턴에 맞도록 조정

모범 사례

파티션 열 선택

- `identity` 파티션 함수에는 카디널리티가 높은 열을 사용하지 마세요. ID 파티셔닝과 함께 높은 카디널리티 열을 사용하면 작은 파티션이 많이 생성되어 수집 성능이 크게 저하될 수 있습니다. 카디널리티가 높은 열에는 다음이 포함될 수 있습니다.
  - 프라이머리 키
  - 타임스탬프 필드(예: `LastModifiedTimestamp`, `CreatedDate`)

  - 시스템 생성 타임스탬프
- 동일한 열에서 여러 타임스탬프 파티션을 선택하지 마세요. 예시:

```
"partitionSpec": [
      {"fieldName": "col1", "functionSpec": "year", "conversionSpec" :
 "epoch_milli"},
      {"fieldName": "col1", "functionSpec": "month", "conversionSpec" :
 "epoch_milli"},
      {"fieldName": "col1", "functionSpec": "day", "conversionSpec" : "epoch_milli"},
      {"fieldName": "col1", "functionSpec": "hour", "conversionSpec" : "epoch_milli"}
]
```

파티션 FunctionSpec/ConversionSpec 선택

- 타임스탬프 기반 파티셔닝 함수를 사용할 때 타임스탬프 기반 파티셔닝에 대해 선택한 열 값의 형식을 나타내는 올바른 ConversionSpec(epoch_sec | epoch_milli | iso)을 지정합니다. AWS Glue 제로 ETL은 파티셔닝 전에 이 파라미터를 사용하여 소스 데이터를 타임스탬프 형식으로 올바르게 변환합니다.
- 데이터 볼륨에 따라 적절한 세분화(년/월/일/시간)를 사용합니다.
- ISO 타임스탬프를 사용할 때 시간대 영향을 고려합니다. AWS Glue 제로 ETL은 선택한 타임스탬프 열의 모든 레코드 값을 UTC 시간대로 채웁니다.
- `bucket` 파티셔닝을 사용하는 경우 쿼리 병렬 처리 요구 사항에 맞는 버킷 수를 선택합니다. 최적의 해시 분포를 위한 일반적인 버킷 수는 2의 거듭제곱(8, 16, 32)입니다.
- `truncate` 파티셔닝의 경우 데이터 패턴에 따라 의미 있는 그룹화를 생성하는 너비 값을 선택합니다. 자르기 너비가 문자열 값의 중요한 접두사 부분을 캡처하는지 확인합니다.
- 불필요한 파티션 오버헤드가 발생할 수 있으므로 매우 작은 데이터세트의 경우에는 `bucket` 또는 `truncate` 함수를 유일한 파티션 전략으로 사용하지 마세요.

오류 처리

NEEDS_ATTENTION 상태

다음과 같은 경우 통합이 NEEDS_ATTENTION 상태가 됩니다.

- 지정된 파티션 열이 소스에 없음
- 파티션 열의 타임스탬프 변환 실패

제한 사항

파티셔닝 제한 사항

- 통합이 생성된 후에는 나면 파티션 사양을 변경할 수 없습니다. 다른 파티셔닝 전략을 사용하려면 새로운 통합을 생성해야 합니다.
- 파티션 열의 최대 수는 10개로 제한됩니다.

교차 계정 통합 제한 사항

- 교차 계정 통합을 생성할 때 AWS Glue 콘솔에는 통합이 존재하지 않는 계정에 호스팅된 대상 AWS Glue 테이블에 대해 UnnestSpec 및 PartitionSpec을 구성하기 위한 CreateIntegrationTableProperty API를 호출하지 않는다는 제한 사항이 있습니다.

  해결 방법: CreateIntegrationTableProperty API는 대상 데이터베이스가 있는 계정에서 CX에 의해 호출되어야 합니다.

다중 통합 제한 사항

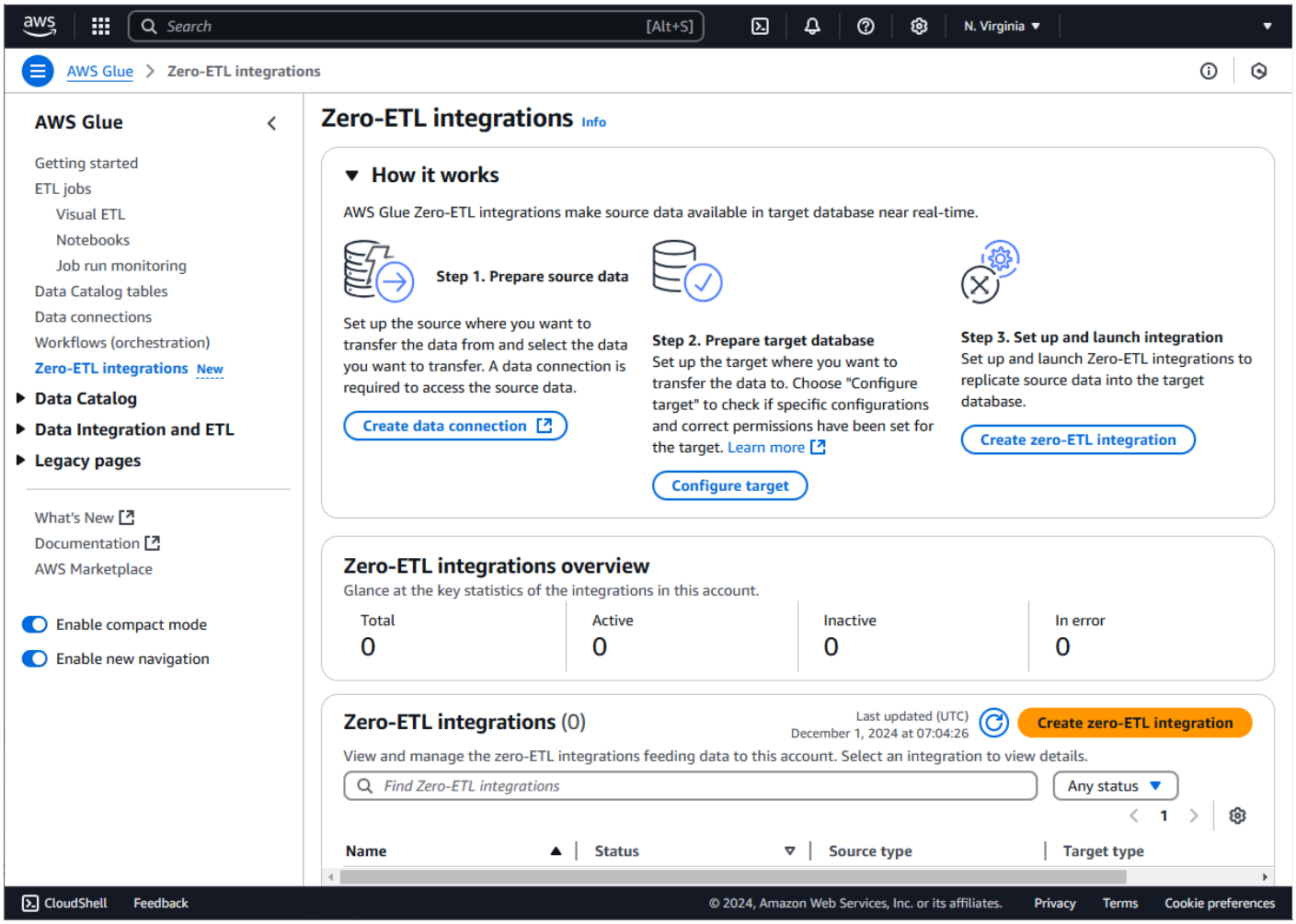
- 동일한 소스를 다른 스키마 중첩 해제/파티션 구성으로 복제하려면 각 통합마다 새로운 AWS Glue 데이터베이스를 별도로 생성해야 합니다. 나중에 개별 AWS Glue 데이터베이스에서 각 테이블에 대해 원하는 스키마 중첩 해제 및 파티셔닝 구성을 사용하여 CreateIntegrationTableProperty를 호출합니다.

일반적인 통합 작업

통합 생성

이 섹션에서는 통합을 생성하는 일반적인 단계를 설명합니다. 이 예제에서는 Amazon DynamoDB를 소스로 사용합니다.

1. AWS Glue 콘솔 홈 페이지에서 제로 ETL 통합을 선택합니다.
2. 제로 ETL 통합 홈 페이지에서 모든 통합을 볼 수 있습니다. 새 통합을 생성하려면 제로 ETL 통합 생성을 선택합니다.



3. 소스 유형을 선택하라는 메시지가 표시됩니다. 소스를 선택하고 다음을 클릭합니다. SaaS 통합 소스에 대해서는 소스 구성 섹션을 참조하세요.
4. 소스 및 대상 구성 페이지에서 복제할 테이블 또는 엔터티를 선택합니다. Amazon DynamoDB의 경우 PITR 및 RBAC 정책이 구성되어 있는지 확인합니다.
5. 통합 대상을 지정합니다.
   - AWS Glue Data Catalog 대상의 경우 데이터를 복제할 AWS Glue 데이터베이스를 선택합니다.
   - Amazon Redshift 데이터 웨어하우스 대상의 경우 Redshift 클러스터 네임스페이스 또는 Redshift Serverless 작업 그룹 네임스페이스를 선택합니다.

   자세한 내용은 대상과의 통합 구성 섹션을 참조하세요.
6. 사전 조건에서 생성한 대상 IAM 역할을 제공합니다.
7. 대상에 저장되는 데이터에 대해 선택적인 대상 KMS 키를 구성하려면 활성화된 KMS 키를 제공합니다. 마찬가지로 대상 네트워크 연결을 구성하려면 AWS Glue 연결을 선택합니다.

8. 대상 수정 버튼은 이 설명서의 사전 조건 섹션에 있는 일부 단계를 구성합니다. 즉, 1) 카탈로그 RBAC 정책을 제공하며 2) Amazon S3 URI가 제공되지 않은 경우 자동으로 생성하고, 그렇지 않으면 제공된 URI를 사용합니다.
9. Redshift 데이터 웨어하우스 대상과 통합하는 경우:
10. 소스 및 대상 구성 페이지의 출력 설정 섹션에서 대상의 데이터에 사용할 스키마 중첩 해제 옵션을 선택합니다. 데이터에서 고객 파티션 키를 사용하려면 사용자 지정 파티션 키 지정을 선택하고 최대 10개의 키를 제공합니다. 그렇지 않으면 단순히 복제되는 DynamoDB 테이블에 할당된 파티션 키를 사용할 수 있습니다.
11. 보안 및 데이터 암호화 섹션에서 데이터를 대상으로 복제하는 중개 프로세스에 사용할 KMS 키를 제공할 수 있습니다. 그렇지 않으면 AWS 관리형 KMS 키가 사용됩니다. 현재는 15분 복제 설정만 지원합니다. 통합 세부 정보에 제로 ETL 통합의 이름을 입력합니다.
12. 제공된 모든 세부 정보가 올바른지 검토하고 확인합니다. 모두 확인했으면 통합 생성 및 시작을 클릭합니다.
13. 제로 ETL 홈 페이지에서 생성한 통합을 선택하면 통합 세부 정보가 표시됩니다. "상태"는 통합 상태를 나타냅니다.

통합 수정

기존 통합을 수정할 수 있습니다.

1. 통합 세부 정보 페이지의 오른쪽 상단 모서리에서 편집을 선택합니다.
2. 소스 및 대상 편집 페이지에서 대상 IAM 역할 및 대상 네트워크 연결을 변경할 수 있습니다. 통합을 생성한 후에는 다른 필드를 편집할 수 없습니다. 다음을 클릭합니다.
3. 통합 및 구성 편집 페이지에서 통합의 이름과 설명을 편집할 수도 있습니다. 다음을 클릭합니다.
4. 편집을 검토하고 확인한 후 통합 업데이트를 클릭합니다.

통합 삭제

삭제는 통합의 말기 상태입니다. 삭제한 후에는 통합을 되살릴 수 없습니다. 통합을 삭제하면 모든 내부 메타데이터와 중간 저장 데이터가 삭제됩니다.

이 프로세스 중에 대상 테이블에 데이터를 쓰는 실행 중인 모든 작업이 종료됩니다. AWS Glue는 Data Catalog에 위치하는 대상 AWS Glue 데이터베이스와 계정의 Amazon S3 버킷에 있는 관련 데이터를 삭제하거나 정리하지 않습니다. 필요한 경우 이들을 명시적으로 정리해야 합니다.

통합을 삭제하려면:

1. 통합 세부 정보 페이지에서 삭제를 클릭합니다.
2. "Delete"를 입력하고 삭제를 클릭합니다. 참고: 이는 되돌릴 수 없는 작업입니다.
3. 통합 세부 정보 페이지에서 상태가 "삭제하는 중"으로 표시됩니다. 통합이 실제로 삭제되면 더 이상 제로 ETL 통합 홈 페이지에 표시되지 않습니다.

API를 사용하여 통합 생성

다음 API를 사용하여 AWS Glue에서 제로 ETL 통합을 생성하고 관리할 수 있습니다.

- CreateIntegration
- CreateIntegrationTableProperties
- CreateIntegrationResourceProperty
- UpdateIntegrationTableProperties
- UpdateIntegrationResourceProperty
- ModifyingIntegration
- DeleteIntegration
- DeleteIntegrationTableProperties
- DescribeIntegrations
- DescribeInboundIntegrations
- GetIntegrationTableProperties
- GetIntegrationResourceProperty

자세한 내용은 AWS Glue의 통합 API 섹션을 참조하세요.

통합 모니터링

통합 상태

다음 통합 상태는 통합을 설명합니다.

- `Creating` - 통합이 생성 중입니다.
- `Active` - 통합이 트랜잭션 데이터를 대상으로 전송하고 있습니다.
- `Modifying` - 통합이 수정 중입니다.
- `Syncing` - 통합에 복구 가능한 오류가 발생하여 데이터를 다시 시드하고 있습니다.
- `Needs attention` - 통합에 수동 개입이 필요한 이벤트 또는 오류가 발생하여 이를 해결해야 합니다. 문제를 해결하기 위해 통합 세부 정보에 있는 오류 메시지의 지침을 따릅니다.
- `Failed` - 통합에서 복구할 수 없는 이벤트 또는 오류가 발생했습니다. 통합을 삭제하고 다시 만들어야 합니다.
- `Deleting` - 통합이 삭제되고 있습니다.

통합에 대한 Amazon CloudWatch 로그 보기

AWS Glue 제로 ETL 통합은 데이터 이동을 파악할 수 있도록 Amazon CloudWatch 로그를 생성합니다. 성공한 각 수집 또는 소스의 문제가 있는 데이터 레코드로 인해 발생하는 모든 실패, 스키마 변경 또는 권한 부족으로 인한 데이터 쓰기 오류와 관련된 로그 이벤트는 고객 계정에 생성된 기본 로그 그룹으로 내보냅니다.

생성된 각 통합의 로그 이벤트는 Amazon Cloudwatch의 `/aws-glue/zeroETL-integrations/logs/`에서 수집됩니다. 로그 그룹 내에서 로그 메시지는 로그 스트림으로 분할됩니다. 생성된 각 통합에는 해당 통합에 대한 모든 로그가 기록되는 전용 로그 스트림이 있습니다. 예를 들어, IntegrationArn `arn:aws:glue:us-east-1:123456789012:integration:03cabe77-79e7-4b7a-b3da-8c160bea6bbf`와의 통합에 대한 로그는 /aws-glue/zeroETL-integrations/logs/03cabe77-79e7-4b7a-b3da-8c160bea6bbf에서 찾을 수 있습니다. 통합이 생성될 때 생성된 {integrationArn}에서 {IntegrationId}를 참조할 수 있습니다.

> **Note**
>
> 교차 계정 시나리오의 경우 통합이 존재하는 소스 계정에서 소스 처리 로그를 내보내고 대상 데이터베이스가 존재하는 대상 계정에서 대상 처리 로그를 내보냅니다.

로깅을 활성화하는 데 필요한 IAM 권한

통합을 생성할 때 통합에 대한 CloudWatch 로깅을 활성화하려면 소스 및 대상 역할에 다음 IAM 권한이 필요합니다. AWS Glue 제로 ETL 통합은 소스 및 대상 역할에 제공된 이러한 권한을 사용하여 CloudWatch 로그를 고객 계정으로 내보냅니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
             ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

로그 메시지

로그 형식: 제로 ETL 통합은 네 가지 유형의 로그 메시지를 내보냅니다.

```
// Ingestion started
{
"integrationArn": "arn:aws:glue:us-east-2:123456789012:integration/1a012bba-123a-1bba-
ab1c-173de3b12345",
...
    "messageType": "IngestionStarted",
    "details": {
        "tableName": "testDDBTable",
        "message": "Ingestion Job started"
    }
}
// Data processing stats on successful table ingestion
{
...
    "messageType": "IngestionProcessingStats",
    "details": {
        "tableName": "testDDBTable",
        "insert_count": 100,
        "update_count": 10,
        "delete_count": 10
    }
}
```

```
// Ingestion failure logs for failed table-processing
{
...
    "messageType": "IngestionFailed",
    "details": {
        "tableName": "testDDBTable",
        "errorMessage": "Failed to ingest data with error: Target Glue database not
 found.",
        "error_code" : "client_error"
    }
}
// Ingestion completed notification with lastSyncedTimestamp
{
...
    "messageType": "IngestionCompleted",
    "details": {
        "tableName": "testDDBTable",
        "message": "Ingestion Job completed"
        "lastSyncedTimestamp": "1132344255745"
    }
}
```

통합에 대한 Amazon CloudWatch 지표 보기

통합이 완료되면 각 AWS Glue 작업 실행에 대해 계정에 생성된 Amazon Cloudwatch 지표를 볼 수 있습니다.

CloudWatch 지표 네임스페이스: "AWS/Glue/ZeroETL"

지표의 차원:

- `integrationArn`
- `loadType`
- `tableName`

지표 이름:

- `InsertCount` - 대상 Iceberg 테이블에 삽입된 레코드 수입니다.
- `UpdateCount` - 대상 Iceberg 테이블에서 업데이트된 레코드 수입니다.
- `DeleteCount` - 대상 Iceberg 테이블에서 삭제된 레코드 수입니다.

- `IngestionSucceeded` - 통합에서 수집이 성공한 경우 1을 카운트합니다.
- `IngestionFailed` - 통합에서 수집이 실패한 경우 1을 카운트합니다.
- `LastSyncTimestamp` - 소스가 대상에 동기화된 때의 타임스탬프입니다.

Amazon EventBridge를 사용하여 이벤트 알림 관리

제로 ETL 통합은 Amazon EventBridge를 통해 이벤트 알림을 관리하여 통합의 변경 사항에 대한 최신 정보를 제공합니다. Amazon EventBridge는 애플리케이션을 다양한 소스의 데이터와 연결하는 데 사용할 수 있는 서버리스 이벤트 버스 서비스입니다. 이 경우 이벤트 소스는 AWS Glue입니다. 환경에서 모니터링되는 변경 사항인 이벤트는 AWS Glue에서 EventBridge로 자동으로 전송됩니다. 이벤트는 거의 실시간으로 전송됩니다.

EventBridge는 특정 이벤트에 대해 수행할 작업을 지정하는 이벤트 규칙을 작성할 수 있는 환경을 제공합니다. 또한 EventBridge가 이벤트를 보낼 수 있는 리소스인 대상을 설정할 수도 있습니다. 대상에는 API 대상, Amazon CloudWatch 로그 그룹 등이 포함될 수 있습니다. 규칙에 대한 자세한 내용은 Amazon EventBridge 규칙을 참조하세요. 대상에 대한 자세한 내용은 Amazon EventBridge 대상을 참조하세요.

모든 제로 ETL 알림을 캡처하려면 다음과 일치하는 Eventbridge 규칙을 생성합니다.

```
{
  "source": [{
    "prefix": "aws.glue-zero-etl“
  }],
  "detail-type": [{
    "prefix": "Glue Zero ETL“
  }]
}
```

다음 테이블에는 추가 메타데이터와 함께 제로 ETL 통합 이벤트가 나와 있습니다.

고객에게 표시되는 세부 정보 유형	설명
Glue 제로 ETL 수집 완료	엔터티에 대한 개별 실행이 성공적으로 완료되었습니다.
Glue 제로 ETL 수집 실패	엔터티에 대한 개별 실행이 성공적으로 완료되지 않았습니다(클라이언트 또는 시스템 오류).

고객에게 표시되는 세부 정보 유형	설명
Glue 제로 ETL 통합 재동기화 완료	통합이 RESYNCED 상태가 되었습니다.
Glue 제로 ETL 통합 실패	오류로 인해 통합 상태가 FAILED로 변경되었습니다.
Glue 제로 ETL 통합 주의 필요	오류로 인해 통합 상태가 NEEDS_ATTENTION으로 변경되었습니다.

제한 사항

다음은 제로 ETL 통합에 대한 일반적인 제한 사항 또는 고려 사항입니다.

- 리소스 속성은 해당 리소스와 일대일 관계를 갖습니다. 따라서 해당 리소스를 사용하여 생성된 모든 통합은 단일 리소스 속성을 준수해야 합니다. 따라서 리소스 속성을 수정하면 해당 리소스와 연결된 모든 통합에 영향을 미칩니다.
- 테이블 속성은 리소스 내의 해당 테이블 또는 객체와 일대일 관계를 갖습니다. 따라서 동일한 리소스를 오가는 동일한 테이블을 처리하는 모든 통합은 단일 테이블 속성을 준수해야 합니다.
- 소스에서 열의 이름을 바꿀 수 없습니다. 열의 이름을 바꾸면 AWS Glue에서 스키마 탐지가 정확하게 수행된다는 보장이 없으며 통합에 대한 영향이 정의되지 않습니다.
- 다음 고려 사항은 통합이 AWS Lake Formation 관리형 테이블에서 작동하는 방식에 적용됩니다. 기본적으로 IAM/AWS Glue 정책을 사용하여 테이블 및 데이터베이스를 관리합니다.
- AWS Lake Formation을 사용하여 해당 데이터베이스에서 테이블 생성을 관리하는 경우 역할에 테이블 및 데이터베이스를 생성, 수정 및 삭제할 수 있는 충분한 Lake Formation 권한이 부여되어 있는지 확인해야 합니다.
- 제로 ETL 요약 페이지에는 현재 어떠한 지표도 포함되어 있지 않습니다.

다음은 제로 ETL 통합의 소스별 제한 사항입니다.

- 이제 SAP OData 소스와의 제로 ETL 통합은 `EntityOf`로 시작하는 엔터티를 지원합니다. 프라이머리 키를 재정의하는 기능은 현재 SAPOData `EntityOf` 객체에만 지원됩니다. 이 속성이 설정되면 수정할 수 없습니다.
- S3를 통한 Amazon DynamoDB에서 Amazon SageMaker Lakehouse로의 제로 ETL 통합은 최대 50TB의 DynamoDB 테이블 크기를 지원합니다.

- 소스 DynamoDB 테이블은 Amazon 소유 또는 고객 관리형 AWS KMS 키로 암호화해야 합니다. 소스 DynamoDB 테이블에는 Amazon 관리형 암호화가 지원되지 않습니다.
- SAP OData는 OAuth 클라이언트와 엔터티의 조합 또는 기본 인증과 엔터티의 조합이 단일 델타 토큰만 가질 수 있는 델타 토큰을 사용하여 작동합니다. 동일한 클라이언트를 사용하는 서로 다른 두 통합에서 동일한 엔터티를 사용하지 마세요.
- 다음 Salesforce 엔터티 또는 필드는 Salesforce 소스를 사용하는 제로 ETL 통합에 사용할 수 없습니다.

```
AccountChangeEvent, AccountContactRoleChangeEvent, AccountHistory, AccountShare,
 ActiveFeatureLicenseMetric, ActivePermSetLicenseMetric, ActiveProfileMetric,
 ActivityFieldHistory, amzsec__asi_Telemetry_Data_Store__ChangeEvent,
 amzsec__asi_Telemetry_Data_Store__History, amzsec__asi_Telemetry_Data_Store__Share,
 amzsec__asi_Telemetry_Job_Log__ChangeEvent, amzsec__asi_Telemetry_Job_Log__History,
 amzsec__asi_Telemetry_Job_Log__Share,
 amzsec__asi_Telemetry_Requirement__ChangeEvent,
 amzsec__asi_Telemetry_Requirement__History,
 amzsec__asi_Telemetry_Requirement__Share, ApexClass, ApexComponent, ApexLog,
 ApexPage, ApexTestQueueItem, ApexTestResult, ApexTrigger, AssetChangeEvent,
 AssetHistory, AssetRelationshipHistory, AssetShare, AssignmentRule,
 AssociatedLocationHistory, AsyncApexJob, AuditTrailFileExportShare,
 AuthorizationFormConsentChangeEvent, AuthorizationFormConsentHistory,
 AuthorizationFormConsentShare, AuthorizationFormDataUseHistory,
 AuthorizationFormDataUseShare, AuthorizationFormHistory, AuthorizationFormShare,
 AuthorizationFormTextHistory, AuthProvider, AuthSession, BatchJobHistory,
 BatchJobPartFailedRecordHistory, BatchJobPartHistory, BatchJobShare,
 BrandTemplate, BriefcaseAssignmentChangeEvent, BriefcaseDefinitionChangeEvent,
 BusinessBrandShare, BusinessHours, BusinessProcess, CalcMatrixColumnRangeHistory,
 CalcProcStepRelationshipHistory, CalculationMatrixColumnHistory,
 CalculationMatrixHistory, CalculationMatrixRowHistory, CalculationMatrixShare,
 CalculationMatrixVersionHistory, CalculationProcedureHistory,
 CalculationProcedureShare, CalculationProcedureStepHistory,
 CalculationProcedureVariableHistory, CalculationProcedureVersionHistory, Calendar,
 CalendarViewShare, CallCenter, CallCoachConfigModifyEvent, CampaignChangeEvent,
 CampaignHistory, CampaignMemberChangeEvent, CampaignMemberStatusChangeEvent,
 CampaignShare, CaseChangeEvent, CaseHistory, CaseHistory2 CaseHistory2ChangeEvent,
 CaseRelatedIssueChangeEvent, CaseRelatedIssueHistory, CaseShare, CaseStatus,
 CaseTeamMember, CaseTeamRole, CaseTeamTemplate, CaseTeamTemplateMember,
 CaseTeamTemplateRecord, CategoryNode, ChangeRequestChangeEvent,
 ChangeRequestHistory, ChangeRequestRelatedIssueChangeEvent,
 ChangeRequestRelatedIssueHistory, ChangeRequestRelatedItemChangeEvent,
 ChangeRequestRelatedItemHistory, ChangeRequestShare, ChatRetirementRdyMetrics,
 ChatterActivity, ClientBrowser, CollaborationGroup, CollaborationGroupMember,
```

```
CollaborationGroupMemberRequest, CollaborationInvitation,
CommSubscriptionChannelTypeHistory, CommSubscriptionChannelTypeShare,
CommSubscriptionConsentChangeEvent, CommSubscriptionConsentHistory,
CommSubscriptionConsentShare, CommSubscriptionHistory, CommSubscriptionShare,
CommSubscriptionTimingHistory, Community, ConnectedApplication, ContactChangeEvent,
ContactHistory, ContactPointAddressChangeEvent, ContactPointAddressHistory,
ContactPointAddressShare, ContactPointConsentChangeEvent,
ContactPointConsentHistory, ContactPointConsentShare,
ContactPointEmailChangeEvent, ContactPointEmailHistory, ContactPointEmailShare,
ContactPointPhoneChangeEvent, ContactPointPhoneHistory, ContactPointPhoneShare,
ContactPointTypeConsentChangeEvent, ContactPointTypeConsentHistory,
ContactPointTypeConsentShare, ContactRequestShare, ContactShare,
ContentDocumentChangeEvent, ContentDocumentHistory, ContentDocumentLink,
ContentDocumentLinkChangeEvent, ContentDocumentSubscription, ContentFolderItem,
ContentFolderLink, ContentFolderMember, ContentNote, ContentNotification,
ContentTagSubscription, ContentUserSubscription, ContentVersionChangeEvent,
ContentVersionComment, ContentVersionHistory, ContentVersionRating,
ContentWorkspace, ContentWorkspaceMember, ContentWorkspacePermission,
ContentWorkspaceSubscription, ContractChangeEvent, ContractHistory,
ContractLineItemChangeEvent, ContractLineItemHistory, ContractStatus,
Conversation, ConversationParticipant, CronJobDetail, CronTrigger, CustomBrand,
CustomBrandAsset, CustomerShare, CustomHttpHeader, DashboardComponent,
DataUseLegalBasisHistory, DataUseLegalBasisShare, DataUsePurposeHistory,
DataUsePurposeShare, DecisionTableRecordset, DeleteEvent, DocumentAttachmentMap,
Domain, DomainSite, DTRecordsetReplicaShare, EmailBounceEvent,
EmailMessageChangeEvent, EmailServicesAddress, EmailServicesFunction,
EmailTemplate, EmailTemplateChangeEvent, EngagementAttendeeChangeEvent,
EngagementAttendeeHistory, EngagementChannelTypeHistory, EngagementChannelTypeShare,
EngagementInteractionChangeEvent, EngagementInteractionHistory,
EngagementInteractionShare, EngagementInterface, EngagementTopicChangeEvent,
EngagementTopicHistory, EntitlementChangeEvent, EntitlementHistory,
EntitlementTemplate, EntityMilestoneHistory, EntitySubscription, EventChangeEvent,
EventRelationChangeEvent, EventRelayConfigChangeEvent, ExpressionSetHistory,
ExpressionSetShare, ExpressionSetVersionHistory, ExternalEventMappingShare,
FeedAttachment, FeedLike, FeedPollChoice, FeedPollVote, FeedSignal,
FieldPermissions, FieldSecurityClassification, FiscalYearSettings,
FlowInterviewLogShare, FlowInterviewShare, FlowOrchestrationEvent,
FlowOrchestrationInstanceShare, FlowOrchestrationStageInstanceShare,
FlowOrchestrationStepInstanceShare, FlowOrchestrationWorkItemShare,
FlowRecordShare, FlowRecordVersionChangeEvent, FlowTestResultShare, Folder,
Group, GroupMember, Holiday, IdeaComment, IdpEventLog, ImageHistory, ImageShare,
IncidentChangeEvent, IncidentHistory, IncidentRelatedItemChangeEvent,
IncidentRelatedItemHistory, IncidentShare, IndividualChangeEvent,
IndividualHistory, IndividualShare, KnowledgeableUser, LeadChangeEvent, LeadHistory,
```

```
LeadShare, LeadStatus, LightningExitByPageMetrics, LightningToggleMetrics,
LightningUsageByAppTypeMetrics, LightningUsageByBrowserMetrics,
LightningUsageByFlexiPageMetrics, LightningUsageByPageMetrics, ListEmailChangeEvent,
ListEmailShare, ListView, LocationChangeEvent, LocationHistory, LocationShare,
LocationTrustMeasureShare, LoginHistory, LoginIp, MacroChangeEvent,
MacroHistory, MacroInstructionChangeEvent, MacroShare, MacroUsageShare,
ManagedContentVariantChangeEvent, MessagingEndUserHistory, MessagingEndUserShare,
MessagingSessionHistory, MessagingSessionShare, MilestoneType, MLEngagementEvent,
ObjectPermissions, OpportunityChangeEvent, OpportunityContactRoleChangeEvent,
OpportunityFieldHistory, OpportunityLineItemChangeEvent, OpportunityShare,
OpportunityStage, OrderChangeEvent, OrderHistory, OrderItemChangeEvent,
OrderItemHistory, OrderShare, OrderStatus, Organization, OrgEmailAddressSecurity,
OrgWideEmailAddress, OutgoingEmail, OutgoingEmailRelation,
PackageLicense, PartnerRole, PartyConsentChangeEvent, PartyConsentHistory,
PartyConsentShare, Period, PermissionSet, PermissionSetAssignment,
PermissionSetTabSetting, Pricebook2ChangeEvent, Pricebook2History,
PricebookEntryChangeEvent, PricebookEntryHistory, PrivacyJobSessionShare,
PrivacyObjectSessionShare, PrivacyRTBFRequestHistory, PrivacyRTBFRequestShare,
PrivacySessionRecordFailureShare, ProblemChangeEvent, ProblemHistory,
ProblemIncidentChangeEvent, ProblemIncidentHistory, ProblemRelatedItemChangeEvent,
ProblemRelatedItemHistory, ProblemShare, ProcessDefinition, ProcessExceptionEvent,
ProcessExceptionShare, ProcessInstanceChangeEvent, ProcessInstanceStep,
ProcessInstanceStepChangeEvent, ProcessNode, Product2ChangeEvent, Product2History,
ProductEntitlementTemplate, Profile, ProfileSkillEndorsementHistory,
ProfileSkillHistory, ProfileSkillShare, ProfileSkillUserHistory, PromptActionShare,
PromptErrorShare, QueueSobject, QuickTextChangeEvent, QuickTextHistory,
QuickTextShare, QuickTextUsageShare, RecentlyViewed, RecommendationChangeEvent,
RecordActionHistory, RecordAlertHistory, RecordAlertShare, RecordType,
ScorecardShare, SellerHistory, SellerShare, ServiceContractChangeEvent,
ServiceContractHistory, ServiceContractShare, SetupAuditTrail, SetupEntityAccess,
SharingRecordCollectionShare, Site, SiteHistory, SiteRedirectMapping,
SocialPersonaHistory, SocialPostChangeEvent, SocialPostHistory, SocialPostShare,
SolutionHistory, SolutionStatus, StaticResource, StreamingChannelShare,
TableauHostMappingShare, TaskChangeEvent, TaskPriority, TaskStatus,
ThreatDetectionFeedback, TimelineObjectDefinitionChangeEvent, TodayGoalShare,
Topic, TopicUserEvent, Translation, User, UserAppMenuCustomizationShare,
UserChangeEvent, UserDefinedLabelAssignmentShare, UserDefinedLabelShare,
UserEmailPreferredPersonShare, UserLicense, UserLogin, UserPackageLicense,
UserPreference, UserPrioritizedRecordShare, UserProvisioningRequestShare,
UserRole, UserShare, VideoCallChangeEvent, VideoCallParticipantChangeEvent,
VideoCallRecordingChangeEvent, VideoCallShare, VisualforceAccessMetrics,
VoiceCallChangeEvent, VoiceCallRecordingChangeEvent, VoiceCallShare, Vote,
WebLink, WorkAccessShare, WorkBadgeDefinitionHistory, WorkBadgeDefinitionShare,
WorkOrderChangeEvent, WorkOrderHistory, WorkOrderLineItemChangeEvent,
```

```
 WorkOrderLineItemHistory, WorkOrderLineItemStatus, WorkOrderShare,
 WorkOrderStatus, WorkPlanChangeEvent, WorkPlanHistory, WorkPlanShare,
 WorkPlanTemplateChangeEvent, WorkPlanTemplateEntryChangeEvent,
 WorkPlanTemplateEntryHistory, WorkPlanTemplateHistory, WorkPlanTemplateShare,
 WorkStepChangeEvent, WorkStepHistory, WorkStepStatus, WorkStepTemplateChangeEvent,
 WorkStepTemplateHistory, WorkStepTemplateShare, WorkThanksShare
```

- 다음 ServiceNow 엔터티 또는 필드는 ServiceNow 소스를 사용하는 제로 ETL 통합에 사용할 수 없습니다.

```
ais_acl_overrides, ais_async_genius_result, ais_async_request,
 ais_connection, ais_genius_result_configuration_parameters,
 ais_partition_health, ais_partition_health_response, ais_publish_history,
 ais_relevancy_training_execution, ais_relevancy_training_staging,
 ais_search_profile_relevancy_model, catalog_draft_entities, clone_log,
 clone_log0000, clone_log0001, clone_log0002, clone_log0003, clone_log0004,
 clone_log0005, clone_log0006, clone_log0007, cmdb_ie_context, cmdb_ie_log,
 cmdb_ie_run, cmdb_ire_partial_payloads_index, cmdb_qb_result_base
$par1, cmdb$par1, discovery_log, discovery_log0000, discovery_log0001,
 discovery_log0002, discovery_log0003, discovery_log0004, discovery_log0005,
 discovery_log0006, discovery_log0007, ecc_agent_log, entitlement_data,
 entl_subscription_map, gs_entitlement_plugin_mapping, ih_transaction_exclusion,
 import_log, import_log0000, import_log0001, import_log0002, import_log0003,
 import_log0004, import_log0005, import_log0006, import_log0007, jrobin_archive,
 jrobin_database, jrobin_datasource, jrobin_definition, jrobin_graph,
 jrobin_graph_line, jrobin_graph_set, jrobin_graph_set_member, jrobin_shard,
 jrobin_shard_location, license_role_discovery_run, logger_configuration_validation,
 m2m_analytics_event_logger, m2m_user_consent_info, ml_artifact_object_store,
 np$sys_gen_ai_filter_sample, np$sys_ui_element, np$sys_ui_list_element,
 one_api_service_plan_feature_invocation, one_api_service_plan_invocation,
 open_nlu_predict_log, open_nlu_predict_log0000, open_nlu_predict_log0001,
 open_nlu_predict_log0002, open_nlu_predict_log0003, open_nlu_predict_log0004,
 open_nlu_predict_log0005, open_nlu_predict_log0006, open_nlu_predict_log0007,
 pa_diagnostic_log, pa_diagnostic_log0000, pa_diagnostic_log0001,
 pa_diagnostic_log0002, pa_diagnostic_log0003, pa_diagnostic_log0004,
 pa_diagnostic_log0005, pa_diagnostic_log0006, pa_diagnostic_log0007, pa_favorites,
 pa_job_log_rows, pa_job_log_rows0000, pa_job_log_rows0001, pa_job_log_rows0002,
 pa_job_log_rows0003, pa_job_log_rows0004, pa_job_log_rows0005, pa_job_log_rows0006,
 pa_job_log_rows0007, pa_migration_ignored_scores, pa_scores_l1, pa_scores_l2,
 pa_scores_migration_groups, par_dashboard_conversion_backup, promin_log,
 promin_log0000, promin_log0001, promin_log0002, promin_log0003, promin_log0004,
 promin_log0005, promin_log0006, promin_log0007, promin_request_object,
 proposed_change_verification_log, proposed_change_verification_log0000,
```

```
proposed_change_verification_log0001, proposed_change_verification_log0002,
proposed_change_verification_log0003, proposed_change_verification_log0004,
proposed_change_verification_log0005, proposed_change_verification_log0006,
proposed_change_verification_log0007, protected_table_log, protected_table_log0000,
protected_table_log0001, protected_table_log0002, protected_table_log0003,
protected_table_log0004, protected_table_log0005, protected_table_log0006,
protected_table_log0007, pwd_history, qb_query_results, scan_log,
scan_log0000, scan_log0001, scan_log0002, scan_log0003, scan_log0004,
scan_log0005, scan_log0006, scan_log0007, schema_validator_error,
sla_repair_log_entry, sla_repair_log_entry0000, sla_repair_log_entry0001,
sla_repair_log_entry0002, sla_repair_log_entry0003, sla_repair_log_entry0004,
sla_repair_log_entry0005, sla_repair_log_entry0006, sla_repair_log_entry0007,
sla_repair_log_message, sla_repair_log_message0000, sla_repair_log_message0001,
sla_repair_log_message0002, sla_repair_log_message0003, sla_repair_log_message0004,
sla_repair_log_message0005, sla_repair_log_message0006, sla_repair_log_message0007,
sn_bm_client_activity, sn_ci_analytics_st_actionable_notifs,
sn_ci_analytics_st_conv_completion_by_cat, sn_ci_analytics_st_conv_dynamic_property,
sn_ci_analytics_st_conversation, sn_ci_analytics_st_count_by_date,
sn_ci_analytics_st_event_occurrence, sn_ci_analytics_st_event_property_value_trend,
sn_ci_analytics_st_issue_auto_resolution, sn_ci_analytics_st_no_clicks,
sn_ci_analytics_st_no_results, sn_ci_analytics_st_property_summary_by_event,
sn_ci_analytics_st_session_count_per_locale, sn_ci_analytics_st_session_duration,
sn_ci_analytics_st_spokes_usage, sn_ci_analytics_st_topic_execution_stats,
sn_ci_analytics_st_topic_occurrence, sn_ci_analytics_st_trending_content,
sn_ci_analytics_st_trending_queries, sn_ci_analytics_st_users,
sn_cs_plugin_signatures, sn_cs_telemetry_log, sn_dfc_application, sn_dfc_product,
sn_employee_position, sn_entitlement_st_subscription_application_family,
sn_entitlement_st_subscription_application_users, sn_hr_sp_st_relevant_for_you,
sn_instance_clone_log, sn_instance_clone_log0000, sn_instance_clone_log0001,
sn_instance_clone_log0002, sn_instance_clone_log0003, sn_instance_clone_log0004,
sn_instance_clone_log0005, sn_instance_clone_log0006, sn_instance_clone_log0007,
sn_km_mr_st_kb_knowledge, sn_me_st_topic, sn_rf_conditional_definition,
sn_rf_evaluation_type, sn_rf_evaluation_type_input, sn_rf_recommendation_action,
sn_rf_recommendation_experience, sn_rf_recommendation_history,
sn_rf_recommendation_rule, sn_rf_record_display_configuration,
sn_rf_trend_definition, sn_sub_man_st_account_level_entitlement,
sn_sub_man_st_gen_ai_metadata, sn_sub_man_st_instance_used_assist_count,
sn_sub_man_st_now_assist_creator_instances, sn_sub_man_st_now_assists_aggregate,
sn_sub_man_st_subscribed_groups, sn_sub_man_st_subscription_insights,
sn_sub_man_st_subscription_license_detail_metric,
sn_sub_man_st_unallocated_group_recommendation,
sn_sub_man_st_unconfirmed_user_group, sn_wn_user_app_activity,
sn_wn_user_content_activity, snc_monitorable_item,
snpar_sched_export_v_scheduled_export_visualization, spotlight,
```

```
spotlight_audit, spotlight_copy_log_row, spotlight_copy_log_row0000,
spotlight_copy_log_row0001, spotlight_copy_log_row0002, spotlight_copy_log_row0003,
spotlight_copy_log_row0004, spotlight_copy_log_row0005, spotlight_copy_log_row0006,
spotlight_copy_log_row0007, spotlight_job_log_row, spotlight_job_log_row0000,
spotlight_job_log_row0001, spotlight_job_log_row0002, spotlight_job_log_row0003,
spotlight_job_log_row0004, spotlight_job_log_row0005, spotlight_job_log_row0006,
spotlight_job_log_row0007, st_dfc_performance_metric, st_license_detail_metric,
st_on_call_hour, st_sc_wizard_question, st_sys_catalog_items_and_variable_sets,
st_sys_design_system_icon, subscription_instance_stats,
svc_container_config, svc_environment_config, svc_layer_config,
svc_model_assoc_ci, svc_model_checkpoint_attr, svc_model_obj_cluster,
svc_model_obj_constraint, svc_model_obj_deployable, svc_model_obj_element,
svc_model_obj_impact, svc_model_obj_impactrule, svc_model_obj_package,
svc_model_obj_path, svc_model_obj_relation, svc_model_obj_service,
sys_administrative_script_transaction, sys_amb_message, sys_amb_processor,
sys_analytics_batch_state, sys_analytics_config, sys_analytics_data_points_error,
sys_analytics_event, sys_analytics_logger, sys_analytics_logger_field,
sys_app_payload_loader_rule, sys_app_payload_unloader_rule, sys_app_scan_payload,
sys_app_scan_variable, sys_app_scan_variable_type, sys_archive_destroy_log,
sys_archive_destroy_run, sys_archive_log, sys_archive_run, sys_atf_transaction_log,
sys_attachment_doc, sys_attachment_doc_v2, sys_attachment_soft_deleted, sys_audit,
sys_audit_relation, sys_auth_policy_api_allowed, sys_aw_registered_scripting_modal,
sys_cache_flush, sys_data_egress_source, sys_dm_delete_count,
sys_export_set_log, sys_export_set_log0000, sys_export_set_log0001,
sys_export_set_log0002, sys_export_set_log0003, sys_export_set_log0004,
sys_export_set_log0005, sys_export_set_log0006, sys_export_set_log0007,
sys_flow_compiled_flow, sys_flow_compiled_flow_chunk, sys_flow_context_chunk,
sys_flow_context_chunk_archive, sys_flow_context_inputs_chunk,
sys_flow_execution_history, sys_flow_log, sys_flow_log0000, sys_flow_log0001,
sys_flow_log0002, sys_flow_log0003, sys_flow_plan_context_binding,
sys_flow_report_doc, sys_flow_report_doc_chunk, sys_flow_report_doc_chunk_archive,
sys_flow_runtime_state_chunk, sys_flow_runtime_value_chunk,
sys_flow_subflow_plan_chunk, sys_flow_trigger_plan_chunk, sys_flow_val_listener,
sys_flow_value, sys_flow_value_chunk, sys_gen_ai_config_example,
sys_gen_ai_feature_mapping, sys_gen_ai_strategy_mapping, sys_gen_ai_usage_log,
sys_generative_ai_capability_definition, sys_generative_ai_log,
sys_generative_ai_response_validator, sys_generative_ai_validator,
sys_geo_routing, sys_geo_routing_config, sys_hop_token, sys_hub_action_plan_chunk,
sys_hub_snapshot_chunk, sys_journal_field, sys_journal_field_edit, sys_json_chunk,
sys_kaa_policy, sys_kaa_subidentity_assertion, sys_kaa_user_policy_mapping,
sys_mapplication, sys_mass_encryption_job, sys_notification_execution_log,
sys_notification_execution_log0000, sys_notification_execution_log0001,
sys_notification_execution_log0002, sys_notification_execution_log0003,
sys_notification_execution_log0004, sys_notification_execution_log0005,
```

```
sys_notification_execution_log0006, sys_notification_execution_log0007,
sys_nowmq_message, sys_nowmq_provider_param_definition, sys_orchestrator_action,
sys_pd_asset_configuration, sys_pd_context_chunk, sys_pd_context_log,
sys_pd_snapshot_chunk, sys_pd_trigger_license, sys_processing_framework_job,
sys_query_index_hint, sys_query_rewrite, sys_query_string_log,
sys_replication_queue, sys_replication_queue0, sys_replication_queue1,
sys_replication_queue2, sys_replication_queue3, sys_replication_queue4,
sys_replication_queue5, sys_replication_queue6, sys_replication_queue7,
sys_request_performance, sys_rollback_blacklisted, sys_rollback_conflict,
sys_rollback_incremental, sys_rollback_log, sys_rollback_log0000,
sys_rollback_log0001, sys_rollback_log0002, sys_rollback_log0003,
sys_rollback_log0004, sys_rollback_log0005, sys_rollback_log0006,
sys_rollback_log0007, sys_rollback_run, sys_rollback_schema_change,
sys_rollback_schema_conflict, sys_rollback_sequence, sys_scheduler_assignment,
sys_scheduler_memory_pressure_job_log, sys_script_adapter_rule,
sys_script_batch_adapter_rule, sys_search_source_filter, sys_service_authentication,
sys_signing_job, sys_suggestion_reader, sys_sync_history_review, sys_trend,
sys_unreferenced_preview, sys_unreferenced_record_rule, sys_upgrade_manifest,
sys_upgrade_state, sys_ux_asset_cache_buster, sys_ux_lib_component_prop,
sys_ux_lib_presource, sys_ux_page_action, sys_ux_page_action_binding,
sysevent_queue_runtime, syslog, syslog_app_scope0000, syslog_app_scope0001,
syslog_app_scope0002, syslog_app_scope0003, syslog_app_scope0004,
syslog_app_scope0005, syslog_app_scope0006, syslog_app_scope0007, syslog_email0000,
syslog_email0001, syslog_email0002, syslog_email0003, syslog_email0004,
syslog_email0005, syslog_email0006, syslog_email0007, syslog_transaction,
syslog_transaction0000, syslog_transaction0001, syslog_transaction0002,
syslog_transaction0003, syslog_transaction0004, syslog_transaction0005,
syslog_transaction0006, syslog_transaction0007, syslog0000, syslog0001,
syslog0002, syslog0003, syslog0004, syslog0005, syslog0006, syslog0007,
ts_attachment, ts_c_1_0, ts_c_1_1, ts_c_1_2, ts_c_1_3, ts_c_1_4, ts_c_1_5,
ts_c_1_6, ts_c_1_7, ts_c_1_8, ts_c_1_9, ts_c_10_0, ts_c_10_1, ts_c_10_2,
ts_c_10_3, ts_c_10_4, ts_c_10_5, ts_c_10_6, ts_c_10_7, ts_c_10_8, ts_c_10_9,
ts_c_11_0, ts_c_11_1, ts_c_11_2, ts_c_11_3, ts_c_11_4, ts_c_11_5, ts_c_11_6,
ts_c_11_7, ts_c_11_8, ts_c_11_9, ts_c_2_0, ts_c_2_1, ts_c_2_2, ts_c_2_3,
ts_c_2_4, ts_c_2_5, ts_c_2_6, ts_c_2_7, ts_c_2_8, ts_c_2_9, ts_c_3_0,
ts_c_3_1, ts_c_3_2, ts_c_3_3, ts_c_3_4, ts_c_3_5, ts_c_3_6, ts_c_3_7, ts_c_3_8,
ts_c_3_9, ts_c_39_0, ts_c_39_1, ts_c_39_2, ts_c_39_3, ts_c_39_4, ts_c_39_5,
ts_c_39_6, ts_c_39_7, ts_c_39_8, ts_c_39_9, ts_c_40_0, ts_c_40_1, ts_c_40_2,
ts_c_40_3, ts_c_40_4, ts_c_40_5, ts_c_40_6, ts_c_40_7, ts_c_40_8, ts_c_40_9,
ts_c_41_0, ts_c_41_1, ts_c_41_2, ts_c_41_3, ts_c_41_4, ts_c_41_5, ts_c_41_6,
ts_c_41_7, ts_c_41_8, ts_c_41_9, ts_c_42_0, ts_c_42_1, ts_c_42_2, ts_c_42_3,
ts_c_42_4, ts_c_42_5, ts_c_42_6, ts_c_42_7, ts_c_42_8, ts_c_42_9, ts_c_43_0,
ts_c_43_1, ts_c_43_2, ts_c_43_3, ts_c_43_4, ts_c_43_5, ts_c_43_6, ts_c_43_7,
ts_c_43_8, ts_c_43_9, ts_c_44_0, ts_c_44_1, ts_c_44_2, ts_c_44_3, ts_c_44_4,
```

```
ts_c_44_5, ts_c_44_6, ts_c_44_7, ts_c_44_8, ts_c_44_9, ts_c_45_0, ts_c_45_1,
ts_c_45_2, ts_c_45_3, ts_c_45_4, ts_c_45_5, ts_c_45_6, ts_c_45_7, ts_c_45_8,
ts_c_45_9, ts_c_6_0, ts_c_6_1, ts_c_6_2, ts_c_6_3, ts_c_6_4, ts_c_6_5,
ts_c_6_6, ts_c_6_7, ts_c_6_8, ts_c_6_9, ts_c_7_0, ts_c_7_1, ts_c_7_2, ts_c_7_3,
ts_c_7_4, ts_c_7_5, ts_c_7_6, ts_c_7_7, ts_c_7_8, ts_c_7_9, ts_c_8_0, ts_c_8_1,
ts_c_8_2, ts_c_8_3, ts_c_8_4, ts_c_8_5, ts_c_8_6, ts_c_8_7, ts_c_8_8, ts_c_8_9,
ts_c_attachment, ts_chain, ts_deleted_doc, ts_document, ts_field, ts_index_stats,
ts_phrase, ts_search_stats, ts_v4_attachment, ts_word, ua_app_metadata,
ua_audit_stats, ua_extra_page, ua_monitor_property, ua_monitor_property_audit,
ua_shared_service, ua_sn_table_inventory, ua_sp_known_bot, ua_upload_log,
v_ais_result_improvement_rule_condition_builder_values, v_cluster_nodes,
v_cxs_search_resource, v_db_index, v_db_trigger, v_file_load_order,
v_interaction_context, v_iostats, v_mysql_proclist, v_mysql_status,
v_mysql_variables, v_on_call_report_cache, v_pa_par_combined_dashboard,
v_pd_activity_condition_to_run, v_pd_activity_start_rule_with_condition,
v_pd_lane_condition_to_run, v_sql_debug, v_st_kb_category, v_st_kb_most_viewed,
v_st_kb_recently_viewed, v_st_km_genai_mra_similar_task, v_st_popular_item,
v_st_recent_item, v_st_sc_cat_item, v_st_sc_catalog, v_st_sc_category,
validator_run_summary, vtb_card_history, wf_log, wf_log0000, wf_log0001,
wf_log0002, wf_log0003, wf_log0004, wf_log0005, wf_log0006, wf_log0007,
spotlight_criteria, pa_snapshots, pa_widget_indicators, pa_widgets,
sn_cim_register, global, gsw_change_log, gsw_content, gsw_content_group,
gsw_content_information, gsw_status_of_content, multi_factor_browser_fingerprint,
multi_factor_criteria, pa_filters, password_policy, plan_execution,
plan_mysql, plan_oracle, plan_postgres, sc_rest_api_without_access_policy,
sn_actsub_activity, sn_actsub_activity_fanout, sn_actsub_activity_stream,
sn_actsub_activity_type, sn_actsub_atype_attributes, sn_actsub_atype_notif_pref,
sn_actsub_module, sn_actsub_notif_object, sn_actsub_subobject_stream,
sn_actsub_subscribable_object, sn_actsub_subscription_notif_pref,
sn_actsub_user_stream, sn_appclient_store_outbound_http_quota,
sn_appcreator_app_template, sn_critical_update, sn_docker_spoke_images,
sn_employee_app, sn_employee_app_access, sn_employee_app_access_criteria,
sn_entitlement_genai_assist_counts, sn_entitlement_genai_creator_user_counts,
sn_entitlement_genai_creator_users, sn_mif_instance, sn_mif_sync_data,
sn_mif_sync_status, sn_mif_table_registration, sn_mif_trust_config,
sn_vsc_best_practice_configurations, sn_vsc_best_practice_goals,
sn_vsc_changed_hardening_settings, sn_vsc_changed_scan_findings,
sn_vsc_check_security_area, sn_vsc_elevation_event, sn_vsc_event,
sn_vsc_export_event, sn_vsc_export_setting, sn_vsc_harc_compliance_status_lookup,
sn_vsc_hardening_compliance_scores, sn_vsc_impersonation_event,
sn_vsc_instance_hardening_settings, sn_vsc_login_event,
sn_vsc_scan_comparisons, sn_vsc_scan_summary, sn_vsc_security_check_categories,
sn_vsc_security_check_configurations, sn_vsc_security_configuration_groups,
sn_vsc_security_privacy_capabilities, sn_vsc_updated_settings,
```

```
sn_vsc_user_comparisons, sys_app_hash_inventory, sys_coalesce_strategy_deferred,
sys_flow_secure_data, sys_formula_function, sys_geocoding_request,
sys_global_file_hash, sys_import_set_row, sys_index, sys_index_explain,
sys_installation_schedule, sys_installation_schedule_item, sys_offline_app,
sys_package, sys_package_dependency_item, sys_package_dependency_m2m,
sys_plugins, sys_querystat, sys_reap_package, sys_scoped_plugin,
sys_stage_storage_alias, sys_storage_alias, sys_storage_table_alias, sys_store_app,
sys_table_partition, sys_upgrade_history_log, sys_user_public_credential,
sys_webauthn_authentication_request, sys_webauthn_registration_request,
syslog_app_scope, syslog_email, syslog_page_timing, ua_instance_state_config,
v_expression_cache, v_private_cache, v_shared_cache, sys_amb_message0002,
sys_amb_message0004, sys_amb_message0005, sys_metadata, v_par_unified_report_viz
```

AWS Glue Data Quality

AWS Glue Data Quality에서는 올바른 비즈니스 결정을 내릴 수 있도록 데이터의 품질을 측정하고 모니터링합니다. 오픈 소스 DeeQu 프레임워크를 기반으로 구축된 AWS Glue Data Quality는 관리형 서버리스 환경을 제공합니다. AWS Glue Data Quality는 데이터 품질 정의 언어(DQDL)를 사용합니다. 이 언어는 데이터 품질에 대한 규칙을 정의하는 데 사용되는 도메인 특정 언어입니다. DQDL 및 지원되는 규칙 유형에 대한 자세한 내용은 데이터 품질 정의 언어(DQDL) 참조 섹션을 참조하세요.

추가 제품 세부 정보 및 요금은 AWS Glue Data Quality의 서비스 페이지를 참조하세요.

이점 및 주요 특징

AWS Glue Data Quality의 이점과 주요 특성은 다음과 같습니다.

- 서버리스 - 설치, 패치 또는 유지 관리가 필요하지 않습니다.
- 빠른 시작 - AWS Glue Data Quality는 데이터를 빠르게 분석하고 사용자를 위해 데이터 품질 규칙을 생성합니다. '데이터 품질 규칙 생성 → 권장 규칙'을 두 번만 클릭하여 시작할 수 있습니다.
- 데이터 품질 문제 탐지 – 기계 학습(ML)을 사용하여 이상과 탐지하기 어려운 데이터 품질 문제를 탐지합니다.
- 규칙 사용자 지정 - 25개 이상의 기본 DQ 규칙부터 시작하여 특정 요구 사항에 맞는 규칙을 생성할 수 있습니다.
- 품질 평가 및 신뢰할 수 있는 비즈니스 의사 결정 지원 - 규칙을 평가한 후에는 데이터 상태에 대한 개요를 제공하는 Data Quality 점수를 제공합니다. Data Quality 점수를 사용하여 신뢰할 수 있는 비즈니스 의사 결정을 지원합니다.
- 잘못된 데이터를 정확하게 식별 - AWS Glue Data Quality를 사용하면 품질 평가 점수를 떨어뜨리는 해당 레코드를 정확히 식별할 수 있습니다. 데이터를 쉽게 식별하고 격리한 후 수정할 수 있습니다.
- 사용한 만큼만 지불 - AWS Glue Data Quality를 사용하는 데 요구되는 연간 라이선스는 없습니다.
- 종속성 없음 - AWS Glue Data Quality는 오픈 소스 DeeQu를 기반으로 구축되었으므로 작성 중인 규칙을 오픈 언어에서 유지할 수 있습니다.
- 데이터 품질 검사 - Data Catalog 및 AWS Glue ETL 파이프라인에서 데이터 품질 검사를 적용하여 저장 중인 데이터 및 전송 중인 데이터의 품질을 관리할 수 있습니다.
- ML 기반 데이터 품질 탐지 – 기계 학습(ML)을 사용하여 이상과 탐지하기 어려운 데이터 품질 문제를 탐지합니다.
- 규칙을 표현할 수 있는 개방형 언어 - 데이터 품질 규칙이 일관되고 간단하게 작성되도록 보장합니다. 비즈니스 사용자는 자신이 이해할 수 있는 간단한 언어로 데이터 품질 규칙을 쉽게 표현할 수 있

습니다. 엔지니어에게 이 언어는 코드를 생성하고, 일관된 버전 제어를 구현하고, 배포를 자동화할 수 있는 유연성을 제공합니다.

작동 방법

AWS Glue Data Quality에는 AWS Glue Data Catalog 및 AWS Glue ETL 작업이라는 두 가지 진입점이 있습니다. 이 섹션에서는 각 진입점에서 지원하는 사용 사례 및 AWS Glue 기능에 대한 개요를 제공합니다.

AWS Glue Data Catalog에 대한 데이터 품질

AWS Glue Data Quality는 AWS Glue Data Catalog에 저장된 객체를 평가합니다. 이를 통해 코딩 작성자가 아닌 사용자도 데이터 품질 규칙을 쉽게 설정할 수 있습니다. 이러한 페르소나로, 데이터 관리자 및 비즈니스 분석가가 해당됩니다.

다음과 같은 사용 사례에서 이 옵션을 선택할 수 있습니다.

- AWS Glue Data Catalog에서 이미 카탈로그화한 데이터 세트에 대해 데이터 품질 작업을 수행하려고 합니다.
- 데이터 거버넌스 관련 업무를 수행하면서 데이터 레이크의 데이터 품질 문제를 지속적으로 식별하거나 평가해야 합니다.

다음 인터페이스를 사용하여 데이터 카탈로그의 데이터 품질을 관리할 수 있습니다.

- AWS Glue 관리 콘솔
- AWS Glue API

AWS Glue Data Catalog에 대해 AWS Glue Data Quality를 시작하려면 Data Catalog에서 AWS Glue Data Quality 시작하기 섹션을 참조하세요.

AWS Glue ETL 작업에 대한 데이터 품질

AWS Glue ETL 작업에 대한 AWS Glue Data Quality를 통해 사전 예방적 데이터 품질 작업을 수행할 수 있습니다. 사전 예방적 작업을 사용하면 데이터 세트를 데이터 레이크에 로드하기 전에 잘못된 데이터를 식별하고 필터링할 수 있습니다.

비디오: Introducing AWS Glue Data Quality for ETL Pipelines

다음 사용 사례에서 ETL 작업에 대한 데이터 품질을 선택할 수 있습니다.

- 데이터 품질 작업을 ETL 작업에 통합하려고 합니다.
- ETL 스크립트에서 데이터 품질 작업을 정의하는 코드를 작성하려고 합니다.
- 시각적 데이터 파이프라인에서 이동하는 데이터의 품질을 관리하려고 합니다.

다음 인터페이스를 사용하여 ETL 작업에 대한 데이터 품질을 관리할 수 있습니다.

- AWS Glue Studio, AWS Glue Studio 노트북 및 AWS Glue 대화형 세션
- ETL 스크립트 작성을 위한 AWS Glue 라이브러리
- AWS Glue API

ETL 작업에 대한 Data Quality를 시작하려면 AWS Glue Studio 사용 설명서의 자습서: Data Quality 시작하기를 참조하세요.

데이터 카탈로그에 대한 데이터 품질과 ETL 작업에 대한 데이터 품질 비교

이 테이블에서는 AWS Glue Data Quality의 각 진입점에서 지원하는 기능에 대한 개요를 제공합니다.

Feature	데이터 카탈로그에 대한 데이터 품질	ETL 작업에 대한 데이터 품질
데이터 소스	Amazon S3, Amazon Redshift, 데이터 카탈로그와 호환되는 JDBC 소스, 그리고 Apache Iceberg, Apache Hudi 및 Delta Lake와 같은 트랜잭션 데이터 레이크 형식. 테이블이 AWS Lake Formation 관리형인 경우 Iceberg, Delta, HUDI 테이블은 지원되지 않습니다. AWS Glue Data Catalog로 카탈로그화된 Amazon Athena 보기는 지원되지 않습니다.	AWS Glue에서 지원하는 모든 데이터 소스(사용자 지정 커넥터 및 서드 파티 커넥터 포함).

Feature	데이터 카탈로그에 대한 데이터 품질	ETL 작업에 대한 데이터 품질
Data Quality 규칙 권장 사항	지원	지원되지 않음
DQDL 규칙 작성 및 실행	지원	지원
Auto Scaling	지원되지 않음	지원
AWS Glue Flex 지원	지원되지 않음	지원
일정 예약	Step Functions를 통해 데이터 품질 규칙을 평가할 때 지원됩니다.	Step Functions와 워크플로를 사용할 때 지원됩니다.
데이터 품질 검사에 실패한 레코드 식별	지원되지 않음	지원
Amazon EventBridge와 통합	지원	지원
AWS Cloudwatch와 통합	지원	지원
Amazon S3에 데이터 품질 결과 작성	지원	지원
증분 데이터 품질	푸시다운 조건자를 통해 지원됨	AWS Glue 북마크를 통해 지원됨
AWS CloudFormation 지원	지원	지원
ML 기반 이상 탐지	지원되지 않음	지원
동적 규칙	지원되지 않음	지원

고려 사항

AWS Glue Data Quality를 사용하기 전에 다음을 고려합니다.

- 데이터 품질 규칙은 중첩된 데이터 소스 또는 목록 유형 데이터 소스를 평가할 수 없습니다. 중첩된 구조체 평면화을(를) 참조하세요.

용어

다음 목록은 AWS Glue 데이터 품질 관련 용어를 정의합니다.

DQDL(데이터 품질 정의 언어)

AWS Glue 데이터 품질 규칙을 작성하는 데 사용할 수 있는 도메인별 언어입니다.

DQDL에 대한 자세한 내용은 데이터 품질 정의 언어(DQDL) 참조 안내서를 참조하세요.

데이터 품질

데이터 세트가 특정 목적에 얼마나 적합한지를 설명합니다. AWS Glue 데이터 품질은 데이터 세트를 기준으로 규칙을 평가하여 데이터 품질을 측정합니다. 각 규칙은 데이터 최신성 또는 무결성과 같은 특정 특성을 확인합니다. 데이터 품질을 정량화하려면 데이터 품질 점수를 사용할 수 있습니다.

데이터 품질 점수

AWS Glue 데이터 품질로 규칙 세트를 평가할 때 통과한 데이터 품질 규칙(true 결과)의 비율입니다.

규칙

데이터에 특정 특성이 있는지 검사하고 부울 값을 반환하는 DQDL 표현식입니다. 자세한 내용은 규칙 구조 섹션을 참조하세요.

분석기

데이터 통계를 수집하는 DQDL 표현식입니다. 분석기는 시간이 지남에 따라 ML 알고리즘이 이상과 탐지하기 어려운 데이터 품질 문제를 탐지하는 데 사용할 수 있는 데이터 통계를 수집합니다.

규칙 세트

일련의 데이터 품질 규칙으로 구성된 AWS Glue 리소스입니다. 규칙 세트는 AWS Glue Data Catalog의 테이블과 연결되어야 합니다. 규칙 세트를 저장할 때 AWS Glue에서는 Amazon 리소스 이름(ARN)을 규칙 세트에 할당합니다.

데이터 품질 점수

AWS Glue 데이터 품질로 규칙 세트를 평가할 때 통과한 데이터 품질 규칙(true 결과)의 비율입니다.

관찰

AWS Glue에서 시간이 지남에 따라 규칙과 분석기로부터 수집된 데이터 통계를 분석하여 생성되는 확인되지 않은 인사이트입니다.

Limits

AWS Glue Data Quality 서비스 한도:

- 규칙 세트에는 2,000개의 규칙을 포함할 수 있습니다. 규칙 세트가 더 크면 여러 규칙 세트로 분할하는 것이 좋습니다.
- 규칙 세트의 크기는 65KB입니다. 규칙 세트가 더 크면 여러 규칙 세트로 분할하는 것이 좋습니다.
- AWS Glue Data Quality는 규칙 또는 분석기를 생성할 때 통계를 수집합니다. 이러한 통계를 저장하는 데는 비용이 들지 않습니다. 단, 계정당 10만 개로 통계 수가 제한되며 이러한 통계는 최대 2년간 보관됩니다.

AWS Glue Data Quality의 릴리즈 정보

이 주제에서는 AWS Glue Data Quality에 도입된 기능에 대해 설명합니다.

정식 출시: 새 기능

AWS Glue Data Quality의 정식 출시를 통해 다음과 같은 새로운 기능을 사용할 수 있습니다.

- 이제 데이터 품질 검사에 실패한 레코드를 식별하는 기능이 AWS Glue Studio에서 지원됩니다.
- 두 데이터 세트 간 데이터의 참조 무결성 검증, 두 데이터 세트 간 데이터 비교, 데이터 형식 검사와 같은 새로운 데이터 품질 규칙 유형
- AWS Glue Data Catalog의 사용자 경험 개선
- Apache Iceberg, Apache Hudi 및 Delta Lake에 대한 지원
- Amazon Redshift에 대한 지원
- Amazon EventBridge를 통한 간소화된 알림
- 규칙 세트 생성을 위한 AWS CloudFormation 지원
- 성능 개선: 데이터 품질 평가 시 더 빠른 성능을 제공하기 위해 ETL 및 AWS Glue Studio의 캐싱 옵션

2023년 11월 27일(미리 보기)

- ML 기반 이상 탐지 기능은 이제 AWS Glue ETL 및 AWS Glue Studio에서 사용할 수 있습니다. 이를 통해 이제 이상과 탐지하기 어려운 데이터 품질 문제를 탐지할 수 있습니다.

- 동적 규칙을 사용하면 동적 임곗값(예: `RowCount > avg(last(10))`)을 제공할 수 있습니다.

2024년 3월 12일

- DQDL 개선 사항
  - NULL, BLANKS, WHITESPACES_ONLY와 같은 키워드에 대한 지원
  - AWS Glue 데이터 품질에서 복합 규칙을 처리하는 방법을 지정하는 옵션
  - ColumnValues 규칙 유형에서는 비교 중에 NULL 값이 전달되는 것을 허용하지 않음
  - DQDL의 NOT 연산자에 대한 지원

2024년 6월 26일

- DQDL 개선 사항
  - DQDL은 이제 where 절을 지원하므로 DQ 규칙을 적용하기 전에 데이터를 필터링할 수 있습니다.

2024년 8월 7일

- 이제 이상 탐지 기능과 동적 규칙을 정식 버전으로 사용할 수 있습니다.

2024년 11월 22일

- 복잡한 복합 규칙을 사용하여 중첩을 지원하는 더 복잡한 비즈니스 규칙을 작성할 수 있음
- 파일의 데이터 품질을 관리하기 위한 새로운 규칙 유형
  - FileFreshness
  - FileSize
  - FileUniqueness
  - FileMatch
- 시각적 ETL 작업의 기본 데이터 품질 검사

2024년 12월 6일

- 이제 AWS Glue Data Quality는 AWS Glue ETL 5.0에서 Amazon SageMaker AI LakeHouse 테이블과 AWS Lake Formation 관리형 Iceberg, Delta 및 HUDI 테이블을 지원합니다.

2025년 6월 30일

- 집계 지표 - 이제 AWS Glue ETL 기반 DQ 작업에 대해 API 수준에서 통과되거나 실패한 레코드 수와 같은 집계 지표를 가져올 수 있습니다.

AWS Glue Data Quality의 이상 탐지

Data Quality의 기계 학습 사례

엔지니어들은 수백 개의 데이터 파이프라인을 동시에 관리합니다. 각 파이프라인은 서로 다른 소스에서 데이터를 추출하여 데이터 레이크에 로드할 수 있습니다. 의사 결정을 위한 고품질 데이터를 제공하기 위해 데이터 품질 규칙을 설정합니다. 이러한 규칙은 현재 비즈니스 상태를 반영하는 고정된 기준에 따라 데이터를 평가합니다. 하지만 비즈니스 환경이 변화하면 데이터 속성이 변화하여 이러한 고정된 기준이 더 이상 맞지 않게 되고 데이터 품질이 저하됩니다.

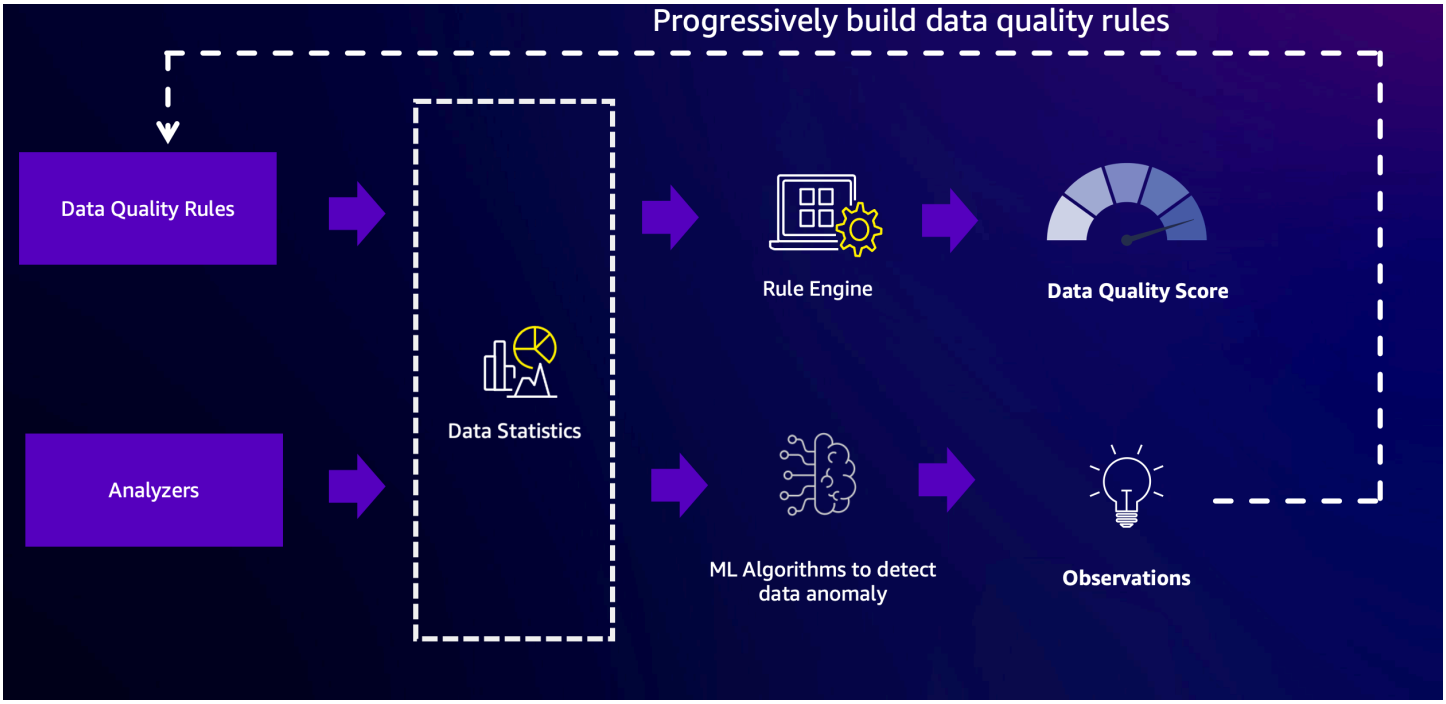
예를 들어 한 소매업체의 데이터 엔지니어는 일일 매출이 임계값 100만 USD를 초과하는지 검증하는 규칙을 설정했습니다. 몇 달 후 일일 매출이 200만 USD를 넘어서면서 이 임계값은 무용지물이 되었습니다. 알림이 부족하고 규칙을 수동으로 분석하고 업데이트하기가 복잡하고 번거로워, 데이터 엔지니어가 최신 임계값을 반영하도록 규칙을 업데이트할 수 없었습니다. 이달 말, 비즈니스 사용자들은 매출이 25%나 감소한 것을 확인했습니다. 몇 시간에 걸친 조사 끝에 데이터 엔지니어들은 일부 매장에서 데이터를 추출하는 ETL 파이프라인이 오류를 발생시키지 않고 실패했다는 사실을 발견했습니다. 임계값이 오래된 규칙은 이 문제를 발견하지 못한 채 계속 정상적으로 작동했습니다.

또는 이러한 이상 현상을 탐지할 수 있는 사전 예방적 알림을 통해 사용자가 이 문제를 탐지할 수 있었을 수도 있습니다. 게다가 비즈니스의 계절성을 추적하면 심각한 데이터 품질 문제를 쉽게 찾아낼 수 있습니다. 예를 들어 소매 판매는 주말과 휴일에 가장 높지만 평일에는 상대적으로 낮을 수 있습니다. 이 패턴과 다를 경우 데이터 품질에 문제가 있거나 비즈니스 상황이 바뀐 것일 수 있습니다. 데이터 품질 규칙으로는 계절적 패턴을 탐지할 수 없습니다. 이를 위해서는 과거 패턴을 학습하여 계절적 변동을 포착함으로써 편차를 탐지할 수 있는 고급 알고리즘이 필요하기 때문입니다.

마지막으로, 규칙 생성 프로세스의 기술적 특성과 규칙 작성에 소요되는 시간 때문에 사용자가 규칙을 생성하고 유지 관리하기가 어렵습니다. 따라서 사용자들은 규칙을 정의하기 전에 먼저 데이터 인사이트를 탐색하는 것을 선호합니다. 고객들은 이상을 쉽게 찾아내 데이터 품질 문제를 사전에 탐지하고 리스크 없이 비즈니스 의사 결정을 내릴 수 있는 기능을 필요로 합니다.

작동 방식

> **Note**
>
> 이상 탐지는 AWS Glue ETL에서만 지원됩니다. 데이터 카탈로그 기반 데이터 품질 솔루션에서는 지원되지 않습니다.



AWS Glue Data Quality는 규칙 기반 데이터 품질의 성능과 이상 탐지 기능을 결합하여 고품질 데이터를 제공합니다. 시작하려면 먼저 규칙과 분석기를 구성한 후 이상 탐지를 활성화해야 합니다.

규칙

규칙 - 규칙은 데이터 품질 정의 언어(DQDL)라는 개방형 언어로 데이터에 대한 기대치를 표현합니다. 아래에 규칙의 예가 나와 있습니다. 이 규칙은 `presger_count` 열에 빈 값이나 NULL 값이 없을 때 통과합니다.

```
Rules = [
    IsComplete "passenger_count"
]
```

분석기

중요 열을 알고 있지만 데이터를 잘 몰라 특정 규칙을 작성할 수 없는 경우, 분석기를 사용하여 해당 열을 모니터링할 수 있습니다. 분석기는 명시적인 규칙을 정의하지 않고도 데이터 통계를 수집할 수 있는 방법입니다. 다음 이미지는 분석기 구성 예를 보여줍니다.

```
Analyzers = [
AllStatistics "fare_amount",
DistinctValuesCount "pulocationid",
RowCount
]
```

이 예에서는 세 개의 분석기가 구성되어 있습니다.

1. 첫 번째 분석기인 `AllStatistics “fare_amount”`는 `fare_amount `필드에 사용 가능한 모든 통계를 캡처합니다.
2. 두 번째 분석기인 `DistinctValuesCount “pulocationid”`는 `pulocationid` 열에 있는 고유 값의 개수를 캡처합니다.
3. 세 번째 분석기인 'RowCount'는 데이터 세트의 총 레코드 수를 캡처합니다.

분석기는 복잡한 규칙을 지정하지 않고도 관련 데이터 통계를 수집할 수 있는 간단한 방법입니다. 이러한 통계를 모니터링하면 데이터 품질에 대한 인사이트를 얻고 추가 조사를 실시하거나 특정 규칙을 생성해야 하는 잠재적 문제 또는 이상을 식별할 수 있습니다.

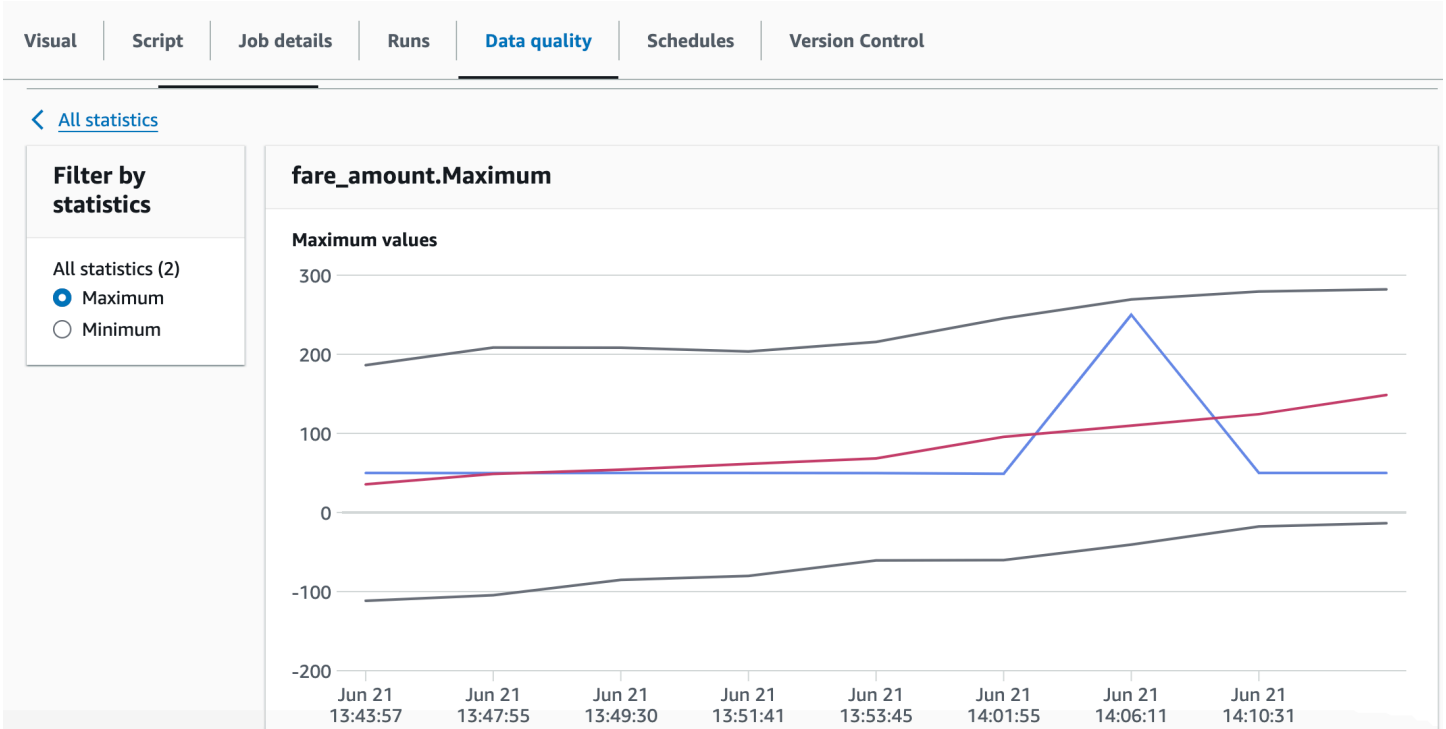
데이터 통계

AWS Glue Data Quality의 분석기와 규칙은 모두 데이터 프로필이라고도 하는 데이터 통계를 수집합니다. 이러한 통계는 데이터의 특성과 품질에 대한 인사이트를 제공합니다. 수집된 통계는 시간이 지남에 따라 AWS Glue 서비스에 저장되므로 데이터 프로필의 변경 사항을 추적하고 분석할 수 있습니다.

적절한 API를 호출하여 이러한 통계를 쉽게 검색하고 추가 분석 또는 장기 보관용으로 Amazon S3에 기록할 수 있습니다. 이 기능을 사용하면 데이터 프로파일링을 데이터 처리 워크플로에 통합하고 수집된 통계를 데이터 품질 모니터링, 이상 탐지 등의 다양한 목적으로 활용할 수 있습니다.

Amazon S3에 데이터 프로필을 저장하면 Amazon 객체 스토리지 서비스의 확장성, 내구성 및 비용 효율성을 활용할 수 있습니다. 또한 다른 AWS 서비스 또는 서드 파티 도구를 활용하여 데이터 프로필을 분석 및 시각화할 수 있으므로, 데이터 품질에 대한 심층적인 인사이트를 얻고 데이터 관리 및 거버넌스와 관련하여 정보에 입각한 결정을 내릴 수 있습니다.

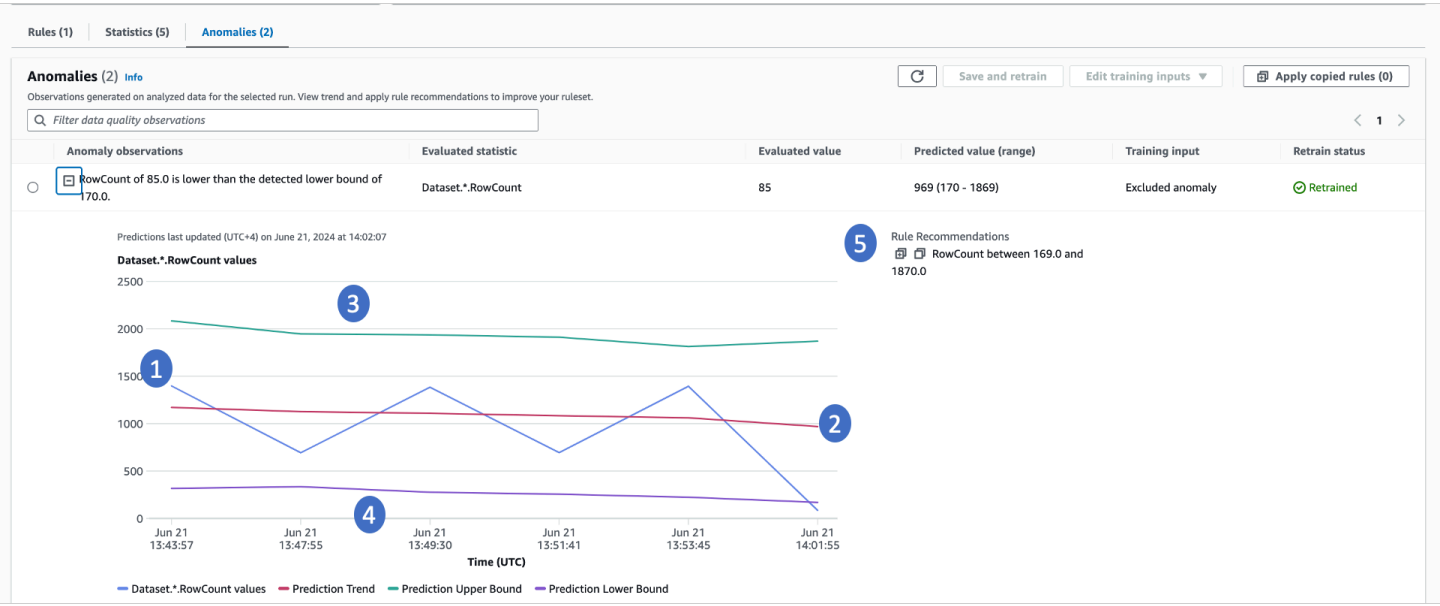
다음은 시간 경과에 따라 저장되는 데이터 통계의 예입니다.



> **Note**
>
> AWS Glue Data Quality는 동일한 열에 규칙과 분석기를 모두 사용하는 경우에도 통계를 한 번만 수집하므로 통계 생성 프로세스가 효율적입니다.

이상 탐지

AWS Glue Data Quality가 이상 현상을 탐지하는 데에는 최소 3개의 데이터 포인트가 필요합니다. 기계 학습 알고리즘을 활용하여 과거 추세를 학습한 다음 미래 값을 예측합니다. 실제 값이 예측 범위 안에 있지 않은 경우 AWS Glue Data Quality는 이상 관찰 항목을 생성합니다. 실제 값과 추세를 시각적으로 보여줍니다. 아래 그래프에는 네 가지 값이 표시됩니다.



1. 실제 통계 및 시간 경과에 따른 추세입니다.
2. 실제 추세를 학습하여 도출된 추세입니다. 추세 방향을 파악하는 데 유용합니다.
3. 통계의 가능한 상한입니다.
4. 통계의 가능한 하한입니다.
5. 향후 이러한 문제를 탐지할 수 있는 권장 데이터 품질 규칙입니다.

이상과 관련하여 유의해야 할 몇 가지 중요한 사항이 있습니다.

- 이상이 생성되어도 데이터 품질 점수는 영향을 받지 않습니다.
- 이상이 탐지되면 후속 실행에서는 정상으로 간주됩니다. 명시적으로 제외하지 않는 한, 기계 학습 알고리즘은 이 이상 값을 입력으로 간주합니다.

재훈련

이상을 정확하게 탐지하려면 이상 탐지 모델을 재훈련하는 것이 중요합니다. 이상이 탐지되면 AWS Glue Data Quality는 모델의 이상 항목을 정상 값으로 포함합니다. 이상 탐지가 정확하게 작동하도록 하려면 이상 항목을 승인하거나 거부하여 피드백을 제공하는 것이 중요합니다. AWS Glue Data Quality는 AWS Glue Studio와 API 모두에서 모델에 피드백을 제공하는 메커니즘을 제공합니다. 자세한 내용은 AWS Glue ETL 파이프라인에서 이상 탐지를 설정하는 방법에 대한 설명서를 참조하세요.

이상 탐지 알고리즘의 세부 정보

- 이상 탐지 알고리즘은 시간 경과에 따른 데이터 통계를 조사합니다. 이 알고리즘은 사용 가능한 모든 데이터 포인트를 고려하고 명시적으로 제외된 통계는 무시합니다.
- 이러한 데이터 통계는 AWS Glue 서비스에 저장되며, 이를 암호화하기 위한 AWS KMS 키를 제공할 수 있습니다. AWS Glue Data Quality 통계를 암호화하기 위한 AWS KMS 키를 제공하는 방법은 보안 가이드를 참조하세요.
- 시간 구성 요소는 이상 탐지 알고리즘에 매우 중요합니다. AWS Glue Data Quality는 과거 값을 기반으로 상한과 하한을 결정합니다. 이 결정 과정에서 시간 구성 요소를 고려합니다. 1분 간격, 1시간 간격 또는 하루 간격으로 값이 같아도 한도가 달라집니다.

계절성 캡처

AWS Glue Data Quality의 이상 탐지 알고리즘은 계절적 패턴을 캡처할 수 있습니다. 예를 들어 평일 패턴이 주말 패턴과 다르다는 것을 이해할 수 있습니다. 이는 AWS Glue Data Quality가 데이터 값의 계절적 추세를 탐지하는 아래 예에서 확인할 수 있습니다. 이 기능을 활성화하는 데 별도의 조치가 필요하지 않습니다. AWS Glue Data Quality는 시간 경과에 따라 계절적 추세를 학습하고 이러한 패턴을 벗어나면 이상을 탐지합니다.

비용

이상을 탐지하는 데 걸리는 시간을 기준으로 요금이 부과됩니다. 모든 통계에는 이상을 탐지하는 데 소요되는 시간을 기준으로 1 DPU의 요금이 부과됩니다. 자세한 예는 AWS Glue 요금을 참조하세요.

주요 고려 사항

통계를 저장하는 데에는 요금이 발생하지 않습니다. 단, 계정당 통계는 10만 건으로 제한됩니다. 이러한 통계는 최대 2년간 저장됩니다.

AWS Glue Data Quality에 대한 IAM 권한 구성

이 주제에서는 IAM 관리자가 AWS Glue Data Quality에 대한 AWS Identity and Access Management(IAM) 정책에서 사용할 수 있는 작업과 리소스를 이해하는 데 도움이 되는 정보를 제공합니다. 또한 AWS Glue 데이터 카탈로그와 함께 AWS Glue Data Quality를 사용하는 데 필요한 최소 권한이 포함된 샘플 IAM 정책도 포함되어 있습니다.

AWS Glue의 보안에 대한 자세한 내용은 AWS Glue의 보안을(를) 참조하세요.

AWS Glue Data Quality에 대한 IAM 권한

다음 테이블에는 특정 AWS Glue Data Quality 작업을 수행하는 데 필요한 권한이 나와 있습니다. AWS Glue Data Quality에 대한 세분화된 권한 부여를 설정하려면 IAM 정책 명령문의 `Action` 요소에서 이러한 작업을 지정할 수 있습니다.

AWS Glue 데이터 품질 작업

작업	설명	리소스 유형
`glue:CreateDataQualityRuleset`	데이터 품질 규칙 세트를 생성할 수 있는 권한을 부여합니다.	`::dataQualityRuleset/<name>`
`glue:DeleteDataQualityRuleset`	데이터 품질 규칙 세트를 삭제할 수 있는 권한을 부여합니다.	`::dataQualityRuleset/<name>`
`glue:GetDataQualityRuleset`	데이터 품질 규칙 세트를 검색할 수 있는 권한을 부여합니다.	`::dataQualityRuleset/<name>`

작업	설명	리소스 유형
glue:ListDataQualityRulesets	모든 데이터 품질 규칙 세트를 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:UpdateDataQualityRuleset	데이터 품질 규칙 세트를 업데이트할 수 있는 권한을 부여합니다.	::dataQualityRuleset/<name>
glue:GetDataQualityResult	데이터 품질 작업 실행 결과를 검색할 수 있는 권한을 부여합니다. 이 IAM 작업은 다음 API에 대한 권한도 제공합니다. • BatchGetDataQualityQualityResult • ListDataQualityStatistics • ListDataQualityStatisticAnnotations	::dataQualityRuleset/<name>
glue:ListDataQualityResults	모든 데이터 품질 작업 실행 결과를 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:CancelDataQualityRuleRecommendationRun	진행 중인 데이터 품질 권장 작업 실행을 중지할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:GetDataQualityRuleRecommendationRun	데이터 품질 권장 작업 실행을 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:ListDataQualityRuleRecommendationRuns	모든 데이터 품질 권장 작업 실행을 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*

작업	설명	리소스 유형
glue:StartDataQualityRuleRecommendationRun	데이터 품질 권장 작업 실행을 시작할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:CancelDataQualityRulesetEvaluationRun	진행 중인 데이터 품질 작업 실행을 중지할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:GetDataQualityRulesetEvaluationRun	데이터 품질 작업 실행을 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:ListDataQualityRulesetEvaluationRuns	모든 데이터 품질 작업 실행을 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/*
glue:StartDataQualityRulesetEvaluationRun	데이터 품질 작업 실행을 시작할 수 있는 권한을 부여합니다.	::dataQualityRuleset/<name>
glue:PublishDataQuality	데이터 품질 결과를 게시할 수 있는 권한을 부여합니다.	::dataQualityRuleset/<name>
glue:GetDataQualityModel	데이터 품질 모델을 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/<name>, ::job/<name>
glue:GetDataQualityModelResult	데이터 품질 모델 결과를 검색할 수 있는 권한을 부여합니다.	::dataQualityRuleset/<name>, ::job/<name>

작업	설명	리소스 유형
glue:PutDataQualityStatisticAnnotation	통계에 주석을 추가할 수 있는 권한을 부여합니다. 이 IAM 작업은 다음 API에 대한 권한도 제공합니다. • BatchPutDataQualityStatisticAnnotation	::dataQualityRuleset/<name>, ::job/<name>
glue:PutDataQualityProfileAnnotation	프로필의 모든 통계에 주석을 달 수 있는 권한을 부여합니다.	::dataQualityRuleset/<name>, ::job/<name>

평가 실행을 예약하는 데 필요한 IAM 설정

IAM 권한

예약된 데이터 품질 평가를 실행하려면 권한 정책에 IAM:PassRole 작업을 추가해야 합니다.

AWS EventBridge 스케줄러 필수 권한

작업	설명	리소스 유형
iam:PassRole	사용자가 승인된 역할을 전달할 수 있도록 IAM에 권한을 부여합니다.	StartDataQualityRulesetEvaluationRun 직접 호출에 사용된 역할의 ARN

이 권한이 없으면 다음과 같은 오류가 발생합니다.

```
"errorCode": "AccessDenied"
"errorMessage": "User: arn:aws:sts::account_id:assumed-role/AWSGlueServiceRole is not
authorized to perform: iam:PassRole on resource: arn:aws:iam::account_id:role/service-
role/AWSGlueServiceRole
because no identity-based policy allows the iam:PassRole action"
```

IAM 신뢰할 수 있는 엔터티

예약된 `StartDataQualityEvaluationRun`을 생성하고 실행하려면 AWS Glue 및 AWS EventBridge 스케줄러 서비스가 신뢰할 수 있는 엔티티에 나열되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

예제 IAM 정책

AWS Glue Data Quality를 대한 IAM 역할에는 다음과 같은 유형의 권한이 필요합니다.

- AWS Glue Data Quality 작업에 대한 권한을 통해 권장 데이터 품질 규칙을 가져오고 AWS Glue 데이터 카탈로그의 테이블에 대해 데이터 품질 작업을 실행할 수 있습니다. 이 섹션의 예제 IAM 정책에는AWS Glue Data Quality 작업에 필요한 최소 권한이 포함되어 있습니다.
- 데이터 카탈로그 테이블 및 기본 데이터에 대한 액세스 권한을 부여하는 권한입니다. 이러한 권한은 사용 사례에 따라 다릅니다. 예를 들어 Amazon S3에서 분류하는 데이터의 경우 권한에 Amazon S3에 대한 액세스가 포함되어야 합니다.

> **Note**
>
> 이 섹션에 설명된 권한과 함께 Amazon S3 권한을 구성해야 합니다.

권장 데이터 품질 규칙에 필요한 최소 권한

이 예제 정책에는 권장 데이터 품질 규칙을 생성하는 데 필요한 권한이 포함되어 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGlueRuleRecommendationRunActions",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRuleRecommendationRun",
        "glue:PublishDataQuality",
        "glue:CreateDataQualityRuleset"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
 {
     "Sid": "AllowCatalogPermissions",
     "Effect": "Allow",
     "Action": [
        "glue:GetPartitions",
        "glue:GetTable"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowS3GetObjectToRunRuleRecommendationTask",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::aws-glue-*"
    },
 {   // Optional for Logs
        "Sid": "AllowPublishingCloudwatchLogs",
        "Effect": "Allow",
        "Action": [
          "logs:CreateLogStream",
          "logs:CreateLogGroup",
          "logs:PutLogEvents"
        ],
```

```
        "Resource": "*"
    },
  ]
}
```

데이터 품질 작업을 실행하기 위한 최소한의 권한

이 예제 정책에는 데이터 품질 평가 작업을 실행하는 데 필요한 권한이 포함되어 있습니다.

다음 정책 설명은 사용 사례에 따라 선택 사항입니다.

- `AllowCloudWatchPutMetricDataToPublishTaskMetrics` - 데이터 품질 실행 지표를 Amazon CloudWatch에 게시하려는 경우 필요합니다.
- `AllowS3PutObjectToWriteTaskResults` - 데이터 품질 실행 결과를 Amazon S3에 기록하려는 경우 필요합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGlueGetDataQualityRuleset",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRuleset"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/<YOUR-
RULESET-NAME>"
    },
    {
      "Sid": "AllowGlueRulesetEvaluationRunActions",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRulesetEvaluationRun",
        "glue:PublishDataQuality"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
 {
      "Sid": "AllowCatalogPermissions",
      "Effect": "Allow",
      "Action": [
        "glue:GetPartitions",
```

```
        "glue:GetTable"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowS3GetObjectForRulesetEvaluationRun",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::aws-glue-*"
    },
    {
      "Sid": "AllowCloudWatchPutMetricDataToPublishTaskMetrics",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": "Glue Data Quality"
        }
      }
    },
    {
      "Sid": "AllowS3PutObjectToWriteTaskResults",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject*"
      ],
      "Resource": "arn:aws:s3:::<YOUR-BUCKET-NAME>/*"
    }
  ]
}
```

데이터 품질 ETL 작업을 실행하기 위한 최소 권한

이 예제 정책에는 데이터 품질 ETL 작업을 실행하는 데 필요한 권한이 포함되어 있습니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGluePublishDataQualityResult",
      "Effect": "Allow",
      "Action": [
        "glue:PublishDataQuality"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
    //Optional to retrieve results, observation generation,
    //dynamic rules and DetectAnomalies
    {
      "Sid": "AllowGlueGetDataQualityResult",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityResult"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
    //Optional to allow annotating statistics
    {
      "Sid": "AllowGlueDataQualityStatisticAnnotation",
      "Effect": "Allow",
      "Action": [
        "glue:PutDataQualityStatisticAnnotation"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*",
        "arn:aws:glue:us-east-1:111122223333::job/{JobName}"
      ]
    },
    //Optional to allow annotating all statistics in a profile
    {
      "Sid": "AllowGlueDataQualityProfileAnnotation",
      "Effect": "Allow",
      "Action": [
        "glue:PutDataQualityProfileAnnotation"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*",
        "arn:aws:glue:us-east-1:111122223333::job/{JobName}"
      ]
    }
```

```
}
```

Data Catalog에서 AWS Glue Data Quality 시작하기

이 시작하기 섹션에서는 AWS Glue 콘솔에서 AWS Glue Data Quality를 시작하는 데 도움이 되는 지침을 제공합니다. 데이터 품질 규칙 권장 사항 생성, 데이터에 대한 규칙 세트 평가와 같은 필수 작업을 완료하는 방법을 알아봅니다.

주제

- 사전 조건
- 단계별 예제
- 규칙 권장 사항 생성
- 모니터링 규칙 권장 사항
- 권장 규칙 세트 편집
- 새 규칙 세트 생성
- 규칙 세트를 실행하여 데이터 품질 평가
- 데이터 품질 점수 및 결과 보기
- 관련 주제

사전 조건

AWS Glue Data Quality를 사용하기 전에 AWS Glue에서 Data Catalog 및 크롤러를 사용하는 방법을 숙지해야 합니다. AWS Glue Data Quality를 사용하여 Data Catalog 데이터베이스의 테이블 품질을 평가할 수 있습니다. 또한 다음 항목이 필요합니다.

- 데이터 품질 규칙 세트를 평가하기 위한 Data Catalog의 테이블.
- 규칙 권장 사항을 생성하거나 데이터 품질 작업을 실행할 때 제공하는 AWS Glue의 IAM 역할. 이 역할에는 다양한 AWS Glue Data Quality 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한이 있어야 합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch가 포함됩니다. AWS Glue Data Quality에 대한 최소 권한이 포함된 정책 예제를 보려면 예제 IAM 정책 섹션을 참조하세요.

  AWS Glue의 IAM 역할에 대한 자세한 내용은 AWS Glue 서비스를 위한 IAM 정책 생성 및 AWS Glue 서비스를 위한 IAM 역할 생성을 참조하세요. AWS Glue Data Quality 작업에 대한 권한에서 데이터 품질과 관련된 모든 AWS Glue 권한 목록을 볼 수도 있습니다.

- 다양한 데이터를 포함하는 테이블이 하나 이상 있는 데이터베이스. 이 자습서에서 사용되는 테이블의 이름은 `yyz-tickets`이며, `tickets` 테이블도 있습니다. 이 데이터는 토론토시가 주차 위반 고지서에서 공개적으로 이용할 수 있는 정보의 컬렉션입니다. 테이블을 직접 생성하는 경우 가장 적합한 권장 규칙 세트를 가져올 수 있도록 테이블에 다양한 유효한 데이터를 입력해야 합니다.

단계별 예제

샘플 데이터 세트가 포함된 단계별 예제는 AWS Glue Data Quality 블로그 게시물을 참조하세요.

규칙 권장 사항 생성

규칙 권장을 사용하면 코드를 작성하지 않고도 데이터 품질을 쉽게 시작할 수 있습니다. AWS Glue Data Quality를 사용하면 데이터를 분석하고, 규칙을 식별하며, 데이터 품질 작업에서 평가할 수 있는 규칙 세트를 생성할 수 있습니다. 권장 사항 실행은 90일 후에 자동으로 삭제됩니다.

데이터 품질 규칙 권장 사항을 생성하려면

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 Tables(테이블)을 선택합니다. 그런 다음 데이터 품질 규칙 권장 사항을 생성할 테이블을 선택합니다.
3. 테이블 세부 정보 페이지에서 데이터 품질 탭을 선택하여 테이블의 AWS Glue Data Quality 규칙 및 설정에 액세스합니다.
4. 데이터 품질 탭에서 규칙 추가 및 데이터 품질 모니터링을 선택합니다.
5. 규칙 권장 사항이 실행되지 않는 경우 규칙 세트 빌더 페이지의 페이지 상단에 권장 작업을 시작하라는 경보가 표시됩니다.
6. 권장 규칙을 선택하여 모달을 열고 권장 작업의 파라미터를 입력합니다.
7. AWS Glue에 액세스할 수 있는 IAM 역할을 선택합니다. 이 역할에는 다양한 AWS Glue 데이터 품질 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여해야 합니다.
8. 기본 설정에 따라 필드를 작성한 후 권장 규칙을 선택하여 권장 작업 실행을 시작합니다. 권장 실행이 진행 중이거나 완료된 경우 이 알림에서 실행을 관리할 수 있습니다. 상태 변경을 보려면 알림을 새로 고쳐야 할 수 있습니다. 완료된 권장 작업 실행과 진행 중인 권장 작업 실행은 지난 90일 동안의 권장 사항 실행을 모두 나열하는 실행 기록 페이지에 표시됩니다.

권장 규칙의 의미

AWS Glue Data Quality는 입력 테이블의 각 열에 있는 데이터를 기반으로 규칙을 생성합니다. 그리고 규칙을 사용하여 품질 요구 사항을 유지하기 위해 데이터를 필터링할 수 있는 잠재적 경계를 식별합니다. 생성된 규칙의 다음 목록에는 규칙의 의미와 데이터에 적용할 때 규칙이 어떤 역할을 할 수 있는지 이해하는 데 유용한 예제가 포함되어 있습니다.

생성된 데이터 품질 정의 언어(DQDL) 규칙 유형의 전체 목록은 DQDL 규칙 유형 참조를 참조하세요.

- `IsComplete "SET_FINE_AMOUNT"` - `IsComplete` 규칙은 해당 행에 열이 채워졌는지 확인합니다. 이 규칙을 사용하여 데이터에서 열에 선택 사항이 아닌 항목으로 태그를 지정합니다.
- `Uniqueness "TICKET_NUMBER" > 0.95` - `Uniqueness` 규칙은 열 내 데이터가 고유성 임곗값을 충족하는지 확인합니다. 이 예제에서는 "TICKET_NUMBER"에 대해 해당 행을 채우는 데이터가 나머지 모든 행의 콘텐츠와 최대 95% 동일하다고 확인되었으며, 이에 따라 이 규칙을 제안합니다.
- `ColumnValues "PROVINCE" in ["ON", "QC", "AB", "NY",...]` - `ColumnValues` 규칙은 기존 열 콘텐츠를 기반으로 열의 유효한 값을 정의합니다. 이 예제에서 각 행의 데이터는 state 또는 province에 대한 2자리 라이선스 코드입니다.
- `ColumnLength "INFRACTION_DESCRIPTION" between 15 and 31` - 이 `ColumnLength` 규칙은 열 데이터에 길이 제한을 적용합니다. 이 규칙은 문자열 열에 기록된 최소 및 최대 길이를 기반으로 샘플 데이터에서 생성됩니다.

모니터링 규칙 권장 사항

데이터 품질 규칙 권장 작업이 실행 중이면 규칙 추가 및 데이터 품질 모니터링 페이지에 상단 표시줄에서 사용할 수 있는 정보와 추가 작업이 표시됩니다.

규칙 권장 작업이 진행 중인 경우 권장 작업이 완료되기 전에 실행 중지를 선택할 수 있습니다. 작업이 진행 중인 동안에는 진행 중 상태가 나타나고, 실행이 시작된 날짜 및 시간이 표시됩니다.

규칙 권장 작업이 완료되면 규칙 권장 사항 표시줄에 권장된 규칙 수, 마지막 권장 실행 상태, 완료 날짜 및 타임스탬프가 표시됩니다.

규칙 권장 사항 삽입을 선택하여 권장 규칙을 추가할 수 있습니다. 이전의 권장 규칙을 보려면 특정 날짜를 선택합니다. 새 권장 작업을 실행하려면 추가 작업을 선택하고 권장 규칙을 선택합니다.

사용자 설정 관리를 선택하여 기본 설정을 지정합니다. Amazon S3의 기본 경로를 설정하여 규칙 세트를 저장하거나 데이터 카탈로그를 실행하기 위한 기본 역할을 설정할 수 있습니다.

권장 규칙 세트 편집

AWS Glue Data Quality는 사용 가능한 기존 데이터를 기반으로 규칙을 생성하므로 자동 제안에서 예상치 못한 규칙 또는 바람직하지 않은 규칙이 표시될 수 있습니다. 권장 규칙 세트를 최대한 활용하려면 규칙 세트를 평가하고 수정해야 합니다. 자습서의 이 단계에서는 이전 단계에서 생성한 규칙을 가져와 이를 조정하여 일부 데이터에 더 제한적인 품질을 적용합니다. 또한 나중에 정확하고 고유한 데이터를 추가할 수 있도록 다른 규칙을 완화할 수도 있습니다.

제안된 규칙 세트 편집

1. AWS Glue 콘솔에서 데이터 카탈로그를 선택하고 탐색 창에서 데이터베이스 테이블을 선택합니다. `tickets` 테이블을 선택합니다.
2. 테이블 세부 정보 페이지에서 데이터 품질 탭을 선택하여 테이블의 AWS Glue Data Quality 규칙 및 설정에 액세스합니다.
3. 규칙 세트 섹션에서는 규칙 권장 사항 생성에서 생성된 규칙 세트를 선택합니다.
4. 작업을 선택한 다음 콘솔 창에서 편집을 선택합니다. 규칙 세트 편집기가 콘솔에 로드됩니다. 여기에는 규칙을 위한 편집 창과 DQDL에 대한 빠른 참조가 포함되어 있습니다.
5. 스크립트의 2번째 줄을 제거합니다. 이렇게 하면 데이터베이스 크기를 특정 행 수로 제한해야 하는 요구 사항이 완화됩니다. 편집 후에는 파일의 1~3번째 줄에 다음을 포함해야 합니다.

```
Rules = [
    IsComplete "TAG_NUMBER_MASKED",
    ColumnLength "TAG_NUMBER_MASKED" between 6 and 9,
```

6. 스크립트의 25번째 줄을 제거합니다. 이렇게 하면 기록된 province의 96%가 ON이어야 한다는 요구 사항이 완화됩니다. 편집 후에는 파일의 24번째 줄부터 규칙 세트 끝까지 다음을 포함해야 합니다.

```
ColumnValues "PROVINCE" in ["ON", "QC", "AB", "NY", "AZ", "NS", "BC", "MI", "PQ",
 "MB", "PA", "FL", "SK", "NJ", "OH", "NB", "IL", "MA", "CA",
    "VA", "TX", "NF", "MD", "PE", "CT", "NC", "GA", "IN", "OR", "MN", "TN", "WI",
 "KY", "MO", "WA", "NH", "SC", "CO", "OK", "VT", "RI", "ME", "AL",
    "YT", "IA", "DE", "AR", "LA", "XX", "WV", "MT", "KS", "NT", "DC", "NV", "NE",
 "UT", "MS", "NM", "ID", "SD", "ND", "AK", "NU", "GO", "WY", "HI"],
ColumnLength "PROVINCE" = 2
]
```

7. 14번째 줄을 다음과 같이 변경합니다.

```
IsComplete "TIME_OF_INFRACTION",
```

이렇게 하면 데이터베이스를 기록된 위반 시간이 포함된 티켓으로만 제한함으로써 열의 요구 사항이 강화됩니다. 기록된 위반 시간이 없는 티켓은 항상 이 데이터 세트에서 유효하지 않은 데이터로 간주해야 합니다. 추가 데이터를 사용하거나 검사하여 품질 규칙을 결정하려고 할 때 파티셔닝 또는 변환이 더 적합한 경우와는 상황이 다릅니다.

8. 콘솔 페이지 하단에서 규칙 세트 업데이트를 선택합니다.

새 규칙 세트 생성

규칙 세트는 데이터를 기준으로 평가하는 데이터 품질 규칙의 그룹입니다. AWS Glue 콘솔에서 데이터 품질 정의 언어(DQDL)를 사용하여 사용자 지정 규칙 세트를 작성할 수 있습니다.

데이터 품질 규칙 세트를 생성하려면

1. AWS Glue 콘솔에서 데이터 카탈로그를 선택하고 데이터베이스를 선택한 다음 탐색 창에서 테이블을 선택합니다. `tickets` 테이블을 선택합니다.
2. Data quality(데이터 품질) 탭을 엽니다.
3. 규칙 섹션에서 규칙 생성을 선택합니다. DQDL 편집기가 콘솔에서 실행됩니다. 여기에는 직접 편집할 수 있는 텍스트 영역과 DQDL 규칙 및 테이블 스키마에 대한 빠른 참조가 포함되어 있습니다.
4. DQDL 편집기의 텍스트 영역에 규칙을 추가하기 시작합니다. 이 자습서에서 직접 규칙을 작성하거나 데이터 품질 규칙 편집기의 DQDL 규칙 빌더 기능을 사용할 수 있습니다.

> **Note**
>
> DQDL 규칙 빌더를 사용하는 방법
>
> 1. 목록에서 규칙 유형을 선택하고 더하기 기호를 선택하여 편집기 창에 예제 구문을 삽입합니다.
> 2. 자리 표시자 열 이름을 자신의 열 이름으로 변경합니다. 테이블의 열 이름은 스키마 탭에서 사용할 수 있습니다.
> 3. 상황에 맞게 표현식 파라미터를 업데이트합니다. DQDL이 지원하는 전체 표현식 목록은 Expressions 섹션을 참조하세요.

예를 들어 다음 규칙은 `tickets` 테이블의 `ticket_number` 열에 대한 데이터 검증과 관련된 제약 조건입니다. 다음 규칙을 추가하려면 DQDL 규칙 빌더를 사용하거나 규칙 세트를 직접 편집합니다.

```
IsComplete "ticket_number",
IsUnique "ticket_number",
ColumnValues "ticket_number" > 9000000000
```

5. 규칙 세트 이름 필드에 새 규칙 세트의 이름을 제공합니다.
6. 규칙 세트 저장을 선택합니다.

여러 데이터 세트에서 데이터 품질 평가

ReferentialIntegrity 및 DatasetMatch 규칙 세트를 사용하여 여러 데이터 세트에서 데이터 품질 규칙을 설정할 수 있습니다. ReferentialIntegrity는 기본 데이터 세트의 데이터가 다른 데이터 세트에 있는지 검사합니다.

참조 데이터 세트를 추가하려면 스키마 탭을 선택한 다음 참조 테이블 업데이트를 선택합니다. 데이터베이스와 테이블을 선택하라는 메시지가 표시됩니다. 테이블을 추가한 다음 데이터 품질 규칙을 설정할 수 있습니다. AggregateMatch, RowCountMatch, ReferentialIntegrity, SchemaMatch 및 DatasetMatch와 같은 규칙 유형은 여러 데이터 세트에서 데이터 품질 검사를 수행하는 기능을 지원합니다.

규칙 세트를 실행하여 데이터 품질 평가

데이터 품질 작업을 실행할 때 AWS Glue 데이터 품질은 데이터를 기준으로 규칙 세트를 평가하고 데이터 품질 점수를 계산합니다. 이 점수는 입력에 대해 통과한 데이터 품질 규칙의 백분율을 나타냅니다.

데이터 품질 작업을 실행하려면

1. AWS Glue 콘솔에서 데이터 카탈로그를 선택하고 데이터베이스를 선택한 다음 탐색 창에서 테이블을 선택합니다. `tickets` 테이블을 선택합니다.
2. 데이터 품질 탭을 선택합니다.
3. 규칙 세트 목록에서 테이블을 기준으로 평가하려는 규칙 세트를 선택합니다. 이 단계에서는 생성된 규칙 대신 이미 작성했거나 수정한 규칙 세트를 사용하는 것이 좋습니다. Run(실행)을 선택합니다.

4. 모달에서 IAM 역할을 선택합니다. 이 역할에는 다양한 AWS Glue 데이터 품질 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여해야 합니다. IAM 역할을 기본값으로 저장하거나 기본 설정 페이지로 이동하여 수정할 수 있습니다.
5. Data quality actions(데이터 품질 작업)에서 Publish metrics to Amazon CloudWatch(Amazon CloudWatch에 지표 게시) 여부를 선택합니다. 이 옵션을 선택하면 AWS Glue Data Quality는 통과한 규칙 수와 실패한 규칙 수를 나타내는 지표를 게시합니다. 이러한 방식으로 저장된 지표에 대해 작업을 수행하기 위해 CloudWatch 경보를 사용할 수 있습니다. 알림을 설정할 수 있도록 주요 지표도 Amazon EventBridge에 게시됩니다. 자세한 내용은 알림, 배포 및 예약 설정을 참조하세요.
6. 실행 빈도에서 온디맨드로 실행을 선택하거나 규칙 세트를 예약합니다. 규칙 세트를 예약하면 작업 이름을 입력하라는 메시지가 표시됩니다. 예약은 Amazon EventBridge에서 생성됩니다. Amazon EventBridge에서 예약을 편집할 수 있습니다.
7. Amazon S3에 데이터 품질 결과를 저장하려면 데이터 품질 결과 위치를 선택합니다. 이 작업을 위해 이전에 선택한 IAM 역할에는 선택한 위치에 대한 쓰기 권한이 있어야 합니다.
8. 추가 구성 아래에 AWS Glue에서 데이터 품질 작업에 할당할 요청된 작업자 수를 입력합니다.
9. 선택적으로 데이터 소스에서 필터를 설정할 수 있습니다. 이렇게 하면 읽는 데이터가 줄어듭니다. 필터를 사용하여 파티션 정보를 선택하고 API 직접 호출을 통해 해당 정보를 파라미터로 전달하여 증분 검증을 실행할 수도 있습니다. 성능 개선을 위해 파티션 조건자를 제공할 수 있습니다.
10. 실행을 선택합니다. Data quality task runs(데이터 품질 작업 실행) 목록에 새 작업이 표시됩니다. 작업의 실행 상태 열이 완료됨으로 표시되면 품질 점수 결과를 볼 수 있습니다. 상태를 올바르게 업데이트하려면 콘솔 창을 새로 고쳐야 할 수도 있습니다.
11. 데이터 품질 결과 세부 정보 열을 보려면 '+' 아이콘을 선택하여 규칙 세트를 확장합니다. 결과에는 평가에서 통과한 규칙과 실패한 규칙, 그리고 규칙 실패의 원인이 표시됩니다.

데이터 품질 점수 및 결과 보기

생성된 모든 규칙 세트의 최신 실행을 보려면

1. AWS Glue 콘솔의 탐색 창에서 Tables(테이블)을 선택합니다. 그런 다음 데이터 품질 작업을 실행할 테이블을 선택합니다.
2. 데이터 품질 탭을 선택합니다.
3. 데이터 품질 스냅샷에서는 시간 경과에 따른 일반적인 실행 추세를 보여줍니다. 기본적으로 모든 규칙 세트에 대한 마지막 10개 실행이 표시됩니다. 규칙 세트별로 필터링하려면 드롭다운 목록에서 원하는 항목을 선택합니다. 실행이 10개 미만인 경우 사용 가능한 모든 완료된 실행이 표시됩니다.

4. 데이터 품질 테이블에는 최근 실행 기록이 있는 각 규칙 세트(있는 경우)가 점수와 함께 표시됩니다. 규칙 세트를 확장하면 해당 규칙 세트에 있는 규칙과 해당 실행의 규칙 결과가 함께 표시됩니다.

특정 규칙 세트의 최신 실행을 보려면

1. AWS Glue 콘솔의 탐색 창에서 Tables(테이블)을 선택합니다. 그런 다음 데이터 품질 작업을 실행할 테이블을 선택합니다.
2. 데이터 품질 탭을 선택합니다.
3. 데이터 품질 테이블에서 특정 규칙 세트를 선택합니다.
4. 규칙 세트 세부정보 페이지에서 실행 기록 탭을 선택합니다.

   이 특정 규칙 세트에 대한 모든 평가 실행이 이 탭의 테이블에 나열됩니다. 점수 기록과 실행 상태를 볼 수 있습니다.
5. 특정 실행에 대한 자세한 내용을 보려면 실행 ID를 선택하여 평가 실행 세부 정보 페이지로 이동합니다. 이 페이지에서는 실행에 대한 세부 정보와 개별 규칙 결과의 상태에 대한 자세한 내용을 볼 수 있습니다.

관련 주제

- DQDL 규칙 유형 참조
- 데이터 품질 정의 언어(DQDL) 참조

AWS Glue Studio에서 데이터 품질 평가

AWS Glue 데이터 품질은 정의된 규칙에 따라 데이터의 품질을 평가하고 모니터링합니다. 그러면 조치가 필요한 데이터를 쉽게 식별할 수 있습니다. AWS Glue Studio에서는 시각적 작업에 데이터 품질 노드를 추가하여 데이터 카탈로그의 테이블에서 데이터 품질 규칙을 생성할 수 있습니다. 그런 다음 시간 경과에 따라 진화하는 데이터 세트의 변경 사항을 모니터링하고 평가할 수 있습니다. AWS Glue Studio에서 AWS Glue Data Quality를 사용하는 방법에 대한 개요는 다음 비디오를 참조하세요.

다음은 AWS Glue 데이터 품질을 사용하는 방법에 대한 주요 단계입니다.

1. Create data quality rules(데이터 품질 규칙 생성) - 구성된 기본 제공 규칙 세트를 선택하여 DQDL 빌더를 통해 데이터 품질 규칙 세트를 구축합니다.

2. Configure a data quality job(데이터 품질 작업 구성) - 데이터 품질 결과 및 출력 옵션을 기반으로 작업을 정의합니다.
3. 데이터 품질 작업 저장 및 실행 - 작업을 생성하고 실행합니다. 작업을 저장하면 해당 작업에 대해 생성한 규칙 세트가 저장됩니다.
4. Monitor and review the data quality results(데이터 품질 결과 모니터링 및 검토) - 작업 실행이 완료된 후 데이터 품질 결과를 검토합니다. 원하는 경우 작업을 미래 날짜로 예약할 수 있습니다.

이점

데이터 분석가, 데이터 엔지니어 및 데이터 사이언티스트는 AWS Glue Studio의 데이터 품질 평가 노드를 사용하여 시각적 작업 편집기에서 데이터 품질을 분석, 구성, 모니터링 및 개선할 수 있습니다. 데이터 품질 노드를 사용하면 다음과 같은 이점이 있습니다.

- 데이터 품질 문제 감지 가능 - 데이터 세트의 특성을 확인하는 규칙을 생성하여 문제를 확인할 수 있습니다.
- 간편한 시작 - 미리 구축된 규칙 및 작업으로 시작할 수 있습니다.
- 긴밀한 통합 - AWS Glue Data Quality는 AWS Glue 데이터 카탈로그를 기반으로 실행되므로 AWS Glue Studio에서 데이터 품질 노드를 사용할 수 있습니다.

AWS Glue Studio에서 ETL 작업에 대한 데이터 품질 평가

이 자습서에서는 AWS Glue Studio에서 AWS Glue Data Quality를 시작합니다. 다음 작업을 수행하는 방법에 대해 알아봅니다.

- 데이터 품질 정의 언어(DQDL) 규칙 작성기를 사용하여 규칙을 생성합니다.
- 데이터 품질 조치, 출력할 데이터 및 데이터 품질 결과의 출력 위치를 지정하는 방법
- 데이터 품질 결과를 검토하는 방법

예제를 사용해 연습하려면 블로그 게시물, ETL 파이프라인용 AWS Glue Data Quality 시작하기를 검토하세요.

1단계: 시각적 작업에 데이터 품질 평가 변환 노드 추가

이 단계에서는 시각적 작업 편집기에 데이터 품질 평가 노드를 추가합니다.

데이터 품질 노드를 추가하려면

1. AWS Glue Studio 콘솔의 작업 생성 섹션에서 소스 및 대상이 있는 시각적 객체를 선택하고 생성을 선택합니다.
2. 데이터 품질 변환을 적용할 노드를 선택합니다. 일반적으로 변환 노드 또는 데이터 소스입니다.
3. '+' 아이콘을 선택하여 왼쪽의 리소스 패널을 엽니다. 검색 표시줄에서 데이터 품질 평가를 검색하고 검색 결과에서 데이터 품질 평가를 선택합니다.
4. 시각적 작업 편집기에는 선택한 노드에서 분기되는 데이터 품질 평가 변환 노드가 표시됩니다. 콘솔 오른쪽에 Transform(변환) 탭이 자동으로 열립니다. 상위 노드를 변경해야 하는 경우 노드 속성 탭을 선택한 다음 드롭다운 메뉴에서 노드 상위 항목을 선택합니다.

   새 노드 상위 항목을 선택하면 상위 노드와 Evaluate Data Quality(데이터 품질 평가) 노드 사이에 새 연결이 설정됩니다. 원치 않는 상위 노드를 제거합니다. 각 Evaluate Data Quality(데이터 품질 평가) 노드에는 상위 노드를 하나만 연결할 수 있습니다.
5. 데이터 품질 평가 변환은 여러 상위 항목을 지원하므로 여러 데이터 세트에서 데이터 품질 규칙을 검증할 수 있습니다. 여러 데이터 세트를 지원하는 규칙으로는 ReferentialIntegrity, DatasetMatch, SchemaMatch, RowCountMatch, AggregateMatch가 있습니다.

   데이터 품질 평가 변환에 여러 입력을 추가할 때는 '기본' 입력을 선택해야 합니다. 기본 입력은 데이터 품질을 검증하려는 데이터 세트입니다. 다른 모든 노드 또는 입력은 참조로 처리됩니다.

   데이터 품질 평가 변환을 사용하여 데이터 품질 검사에 실패한 특정 레코드를 식별할 수 있습니다. 잘못된 레코드에 플래그를 지정하는 새 열이 기본 데이터 세트에 추가되므로 기본 데이터 세트를 선택하는 것이 좋습니다.
6. 입력 데이터 소스의 별칭을 지정할 수 있습니다. 별칭은 ReferentialIntegrity 규칙을 사용할 때 입력 소스를 참조하는 또 다른 방법을 제공합니다. 하나의 데이터 소스만 기본 소스로 지정할 수 있으므로 사용자가 추가하는 각 추가 데이터 소스에는 별칭이 필요합니다.

   다음 예제에서 ReferentialIntegrity 규칙은 별칭 이름으로 입력 데이터 소스를 지정하고 기본 데이터 소스에 대한 일대일 비교를 수행합니다.

```
Rules = [
 ReferentialIntegrity “Aliasname.name” = 1
]
```

2단계: DQDL을 사용하여 규칙 생성

이 단계에서는 DQDL을 사용하여 규칙을 생성합니다. 이 자습서에서는 완전성 규칙 유형을 사용하여 단일 규칙을 생성합니다. 이 규칙 유형은 지정된 표현식을 기준으로 열에서 전체(null이 아닌) 값의 백분율을 검사합니다. DQDL 사용에 대한 자세한 내용은 DQDL을 참조하세요.

1. 변환 탭에서 삽입 버튼을 선택하여 규칙 유형을 추가합니다. 그러면 규칙 편집기에 규칙 유형이 추가되어 규칙에 대한 파라미터를 입력할 수 있습니다.

   > **Note**
   >
   > 규칙을 편집할 경우 규칙이 대괄호 안에 있고 쉼표로 구분되었는지 확인합니다. 예를 들어 전체 규칙 표현식은 다음과 비슷합니다.
   >
   > ```
   > Rules= [
   >     Completeness "year">0.8, Completeness "month">0.8
   > ]
   > ```
   >
   > 이 예제에서는 이름이 'year' 및 'month'인 열의 완전성 파라미터를 지정합니다. 규칙이 통과되려면 이러한 열의 'complete' 비율이 80%를 초과하거나 각 열에 대해 80%가 넘는 인스턴스에 데이터가 있어야 합니다.

   이 예에서는 Completeness(완전성) 규칙 유형을 검색하여 삽입합니다. 그러면 규칙 유형이 규칙 편집기에 추가됩니다. 이 규칙 유형의 구문은 다음과 같습니다. `Completeness <COL_NAME> <EXPRESSION>`.

   대부분의 규칙 유형에서는 부울 응답을 생성하려면 표현식을 파라미터로 제공해야 합니다. 지원되는 DQDL 표현식에 대한 자세한 내용은 DQDL 표현식을 참조하세요. 이제 열 이름을 추가합니다.

2. DQDL 규칙 작성기에서 스키마 탭을 선택합니다. 검색 표시줄을 사용하여 입력 스키마에서 열 이름을 찾습니다. 입력 스키마는 열 이름과 데이터 유형을 표시합니다.

3. 규칙 편집기에서 규칙 유형의 오른쪽을 클릭하여 열을 삽입할 위치에 커서를 삽입합니다. 또는 규칙에 열 이름을 입력할 수 있습니다.

   예를 들어 입력 스키마 목록의 열 목록에서 열(이 예제의 경우 year) 옆에 있는 삽입 버튼을 클릭합니다. 그러면 해당 열이 규칙에 추가됩니다.

4. 그런 다음 규칙 편집기에서 규칙을 평가하는 표현식을 추가합니다. 완전성 규칙 유형은 지정된 표현식을 기준으로 열에서 전체(null이 아닌) 값의 백분율을 검사하므로 `> 0.8`과 같은 표현식을 입력합니다. 이 규칙은 전체(null이 아닌) 값이 80%를 초과하는 경우에 열을 검사합니다.

3단계: 데이터 품질 출력 구성

데이터 품질 규칙을 생성한 후 추가 옵션을 선택하여 데이터 품질 노드 출력을 지정할 수 있습니다.

1. 데이터 품질 변환 출력에서 다음 옵션 중 하나를 선택합니다.

   - 원래 데이터 - 원래 입력 데이터를 출력하려면 선택합니다. 이 옵션을 선택하면 새 하위 노드, 'rowLevelOutcomes'가 작업에 추가됩니다. 스키마는 변환에 대한 입력으로 전달된 기본 데이터 세트의 스키마와 일치합니다. 이 옵션은 데이터를 전달할 때 품질 문제가 있으면 작업을 실패시키려는 경우에 유용합니다.

     또 다른 사용 사례로, 데이터 품질 검사에 실패한 잘못된 레코드를 감지하려는 경우가 있습니다. 잘못된 레코드를 감지하려면 데이터 품질 오류를 표시하도록 새 열 추가 옵션을 선택합니다. 이 작업을 수행하면 'rowLevelOutcomes' 변환의 스키마에 4개의 새 열이 추가됩니다.
     - DataQualityRulesPass(문자열 배열) – 데이터 품질 검사에 통과한 규칙 배열을 제공합니다.
     - DataQualityRulesFail(문자열 배열) – 데이터 품질 검사에 실패한 규칙 배열을 제공합니다.
     - DataQualityRulesSkip(문자열 배열) – 건너뛴 규칙 배열을 제공합니다. 다음 규칙은 데이터 세트 수준에서 적용되므로 오류 레코드를 식별할 수 없습니다.
       - AggregateMatch
       - ColumnCount
       - ColumnExists
       - ColumnNamesMatchPattern
       - CustomSql
       - RowCount
       - RowCountMatch
       - StandardDeviation
       - 평균
       - ColumnCorrelation
     - DataQualityEvaluationResult – 행 수준에서 '통과' 또는 '실패' 상태를 제공합니다. 전체 결과는 실패일 수 있지만 특정 레코드는 통과할 수 있습니다. 예를 들어 RowCount 규칙은 실패해도 다른 모든 규칙이 성공했을 수 있습니다. 이 경우 이 필드 상태는 '통과'입니다.

2. 데이터 품질 결과 - 구성된 규칙과 규칙의 통과 또는 실패 상태를 출력하려면 선택합니다. 이 옵션은 결과를 Amazon S3 또는 다른 데이터베이스에 기록하려는 경우 유용합니다.
3. 데이터 품질 출력 설정(선택 사항) - 데이터 품질 출력 설정을 클릭하여 데이터 품질 결과 위치 필드를 표시합니다. 그런 다음 찾아보기를 클릭하여 데이터 품질 출력 대상으로 설정할 Amazon S3 위치를 검색합니다.

4단계. 데이터 품질 작업 구성

작업을 사용하여 지표를 CloudWatch에 게시하거나 특정 기준에 따라 작업을 중지할 수 있습니다. 작업은 규칙을 생성한 경우에만 사용할 수 있습니다. 이 옵션을 선택하면 동일한 지표도 Amazon EventBridge에 게시됩니다. 이러한 옵션을 사용하여 알림용 경보를 생성할 수 있습니다.

- 규칙 세트 실패 시 - 작업 실행 중에 규칙 세트에서 실패한 경우 수행할 조치를 선택할 수 있습니다. 데이터 품질에서 실패한 경우 작업이 실패하게 하려면 다음 옵션 중 하나를 선택하여 작업이 실패해야 하는 시점을 선택합니다. 기본적으로 이 작업은 선택되지 않으며 데이터 품질 규칙에서 실패하더라도 작업 실행은 완료됩니다.
  - 없음 – 없음(기본값)을 선택하면 규칙 세트에서 실패했어도 작업은 실패하지 않고 계속 실행됩니다.
  - 대상에 데이터를 로드한 후 작업 실패 - 작업이 실패하고 데이터는 저장되지 않습니다. 결과를 저장하려면 데이터 품질 결과를 저장할 Amazon S3 위치를 선택합니다.
  - 대상 데이터를 로드하지 않고 작업 실패 - 이 옵션은 데이터 품질 오류가 발생하는 경우 즉시 작업이 실패하게 합니다. 데이터 품질 변환의 결과를 포함하여 어떠한 데이터 대상도 로드되지 않습니다.

5단계: 데이터 품질 결과 보기

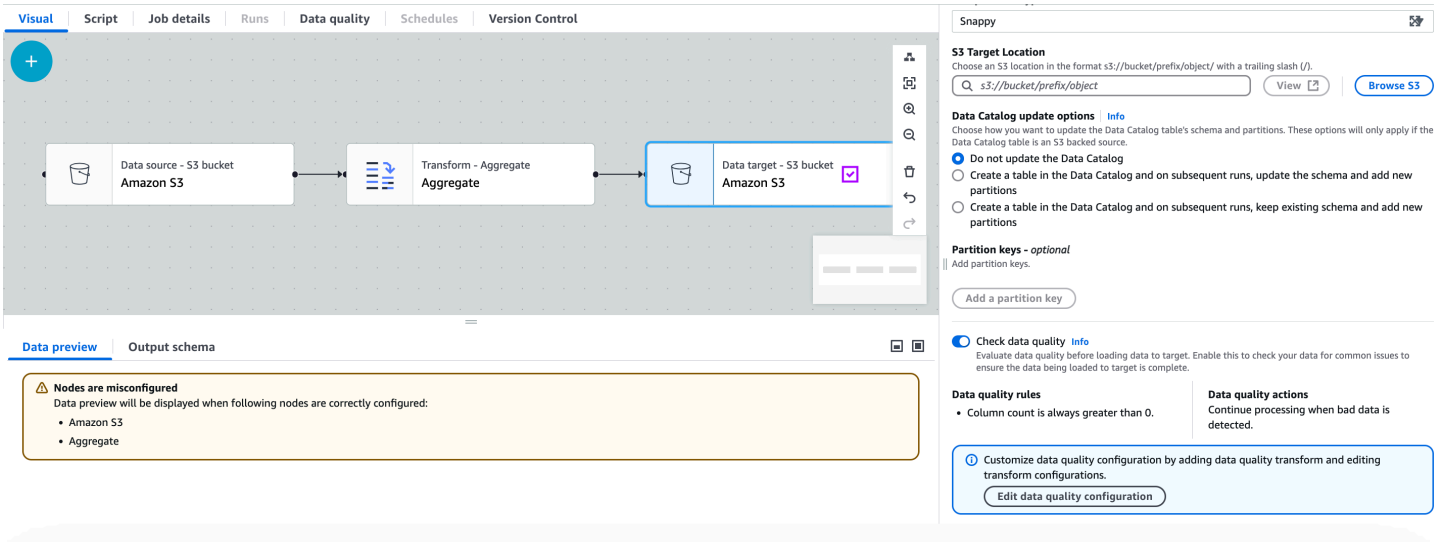
작업을 실행한 후 데이터 품질 탭을 클릭하여 데이터 품질 결과를 확인합니다.

1. 각 작업 실행에 대해 데이터 품질 결과를 확인합니다. 각 노드에는 데이터 품질 상태와 상태 세부 정보가 표시됩니다. 노드를 선택하면 모든 규칙과 각 규칙의 상태가 표시됩니다.
2. 결과 다운로드를 선택하여 작업 실행 및 데이터 품질 결과에 대한 정보가 포함된 CSV 파일을 다운로드합니다.
3. 데이터 품질 결과를 포함하는 작업 실행이 두 개 이상인 경우 날짜 및 시간 범위별로 결과를 필터링할 수 있습니다. 날짜 및 시간 범위별 필터링을 선택하여 필터 기간을 확장합니다.

4. Relative range(상대 범위) 또는 Absolute range(절대 범위)를 선택합니다. 절대 범위의 경우 달력을 사용하여 날짜를 선택하고 시작 시간 및 종료 시간 값을 입력합니다. 완료했으면 적용을 선택합니다.

자동 데이터 품질

Amazon S3를 대상으로 하는 AWS Glue ETL 작업을 생성할 경우 AWS Glue ETL은 로드되는 데이터에 열이 하나 이상 있는지 확인하는 데이터 품질 규칙을 자동으로 활성화합니다. 이 규칙은 로드되는 데이터가 비어 있거나 손상되지 않았는지 확인하도록 설계되었습니다. 그러나 이 규칙이 실패하면 작업이 실패하는 것이 아니라 데이터 품질 점수가 감소합니다. 또한 이상 탐지가 기본적으로 활성화되어 데이터의 열 수를 모니터링합니다. 열 수에 변형이나 이상이 있는 경우 AWS Glue ETL은 이러한 이상에 대해 알려줍니다. 이 기능은 데이터의 잠재적 문제를 식별하고 적절한 조치를 취하는 데 도움이 됩니다. 데이터 품질 규칙 및 해당 구성을 보려면 AWS Glue ETL 작업에서 Amazon S3 대상을 클릭하면 됩니다. 제공된 스크린샷에서 볼 수 있는 것처럼, 규칙의 구성이 표시됩니다.



데이터 품질 구성 편집을 선택하여 추가적인 데이터 품질 규칙을 추가할 수 있습니다.

집계 지표

대시보드를 구축하려면 규칙 수준이나 규칙 세트 수준에서 통과되거나 실패하거나 건너뛴 레코드 수와 같은 집계 지표가 필요할 수 있습니다. 각 규칙에 대한 집계 지표와 규칙 지표를 가져오려면 먼저 `EvaluateDataQuality` 함수에 `publishAggregatedMetrics` 옵션을 추가하여 집계 지표를 사용하도록 설정해야 합니다.

`additional_options` `publishAggregatedMetrics`에 사용할 수 있는 옵션은 `ENABLED` 및 `DISABLED`입니다. 예를 들면 다음과 같습니다.

```
EvaluateDataQualityMultiframe = EvaluateDataQuality().process_rows(
    frame=medicare_dyf,
    ruleset=EvaluateDataQuality_ruleset,
    publishing_options={
        "dataQualityEvaluationContext": "EvaluateDataQualityMultiframe",
        "enableDataQualityCloudWatchMetrics": False,
        "enableDataQualityResultsPublishing": False,
    },
    additional_options={"publishAggregatedMetrics.status": "ENABLED"},
)
```

지정하지 않으면 publishAggregatedMetrics.status는 기본적으로 DISABLED이고 이제 ruleMetrics 및 집계 지표가 계산됩니다. 이 기능은 현재 AWS Glue 대화형 세션 및 Glue ETL 작업에서 지원됩니다. 이 기능은 Glue 카탈로그 데이터 품질 API에서 지원되지 않습니다.

집계 지표 결과 검색

additionalOptions "publishAggregatedMetrics.status": "ENABLED"인 경우 다음 두 장소에서 결과를 가져올 수 있습니다.

1. AggregatedMetrics 및 RuleMetrics는 resultId를 제공할 때 GetDataQualityResult()를 통해 반환됩니다. 이 때 AggregatedMetrics 및 RuleMetrics에는 다음이 포함됩니다.

   집계 지표:

   - 처리된 총 행 수
   - 통과된 총 행
   - 실패한 총 행
   - 처리된 총 규칙
   - 통과된 총 규칙
   - 실패한 총 규칙

```
[redacted] % aws --region us-east-1 glue get-data-quality-result --result-id "dqresult-09655f0234b3e3d129381bd162d0347ed25d1c05"
{
    "ResultId": "dqresult-09655f0234b3e3d129381bd162d0347ed25d1c05",
    "ProfileId": "dqprofile-[redacted]",
    "Score": 0.6666666666666666,
    "RulesetName": "EvaluateDataQuality_node1741974822533",
    "EvaluationContext": "EvaluateDataQuality_node1741974822533",
    "StartedOn": 1750687515.374,
    "CompletedOn": 1750687537.485,
    "JobName": "[redacted]_aggregatedmetrics_test",
    "JobRunId": "jr_[redacted]",
    "RuleResults": [
        {
            "Name": "Rule_1",
            "Description": "IsUnique \"customer_identifier\"",
            "Result": "PASS",
            "EvaluatedMetrics": {
                "Column.customer_identifier.Uniqueness": 1.0
            },
            "EvaluatedRule": "IsUnique \"customer_identifier\"",
            "RuleMetrics": {
                "rows_passed": 10.0,
                "rows_failed": 0.0,
                "rows_skipped": 0.0,
                "total_rows": 10.0
            }
        },
        {
            "Name": "Rule_2",
            "Description": "RowCount > 10",
            "EvaluationMessage": "Value: 10 does not meet the constraint requirement!",
            "Result": "FAIL",
            "EvaluatedMetrics": {
                "Dataset.*.RowCount": 10.0
            },
            "EvaluatedRule": "RowCount > 10",
            "RuleMetrics": {
                "rows_passed": 0.0,
                "rows_failed": 0.0,
                "rows_skipped": 10.0,
                "total_rows": 10.0
            }
        },
        {
            "Name": "Rule_3",
            "Description": "Completeness \"customer_identifier\" > 0.5",
            "Result": "PASS",
            "EvaluatedMetrics": {
                "Column.customer_identifier.Completeness": 1.0
            },
            "EvaluatedRule": "Completeness \"customer_identifier\" > 0.5",
            "RuleMetrics": {
                "rows_passed": 10.0,
                "rows_failed": 0.0,
                "rows_skipped": 0.0,
                "total_rows": 10.0
            }
        }
    ],
    "AggregatedMetrics": {
        "TotalRowsProcessed": 10.0,
        "TotalRowsPassed": 10.0,
        "TotalRowsFailed": 0.0,
        "TotalRulesProcessed": 3.0,
        "TotalRulesPassed": 2.0,
        "TotalRulesFailed": 1.0
    }
}
```

또한 규칙 수준에서는 다음과 같은 지표가 제공됩니다.

규칙 지표:

- 통과된 행
- 실패한 행
- 건너뛴 행
- 처리된 총 행

2. AggregatedMetrics는 추가 데이터 프레임으로 반환되고, RuleOutcomes 데이터 프레임은 RuleMetrics를 포함하도록 보강됩니다.

예제 구현

다음 예제는 Scala에서 집계 지표를 구현하는 방법을 보여줍니다.

```
// Script generated for node Evaluate Data Quality
val EvaluateDataQuality_node1741974822533_ruleset = """
  # Example rules: Completeness "colA" between 0.4 and 0.8, ColumnCount > 10
  Rules = [
      IsUnique "customer_identifier",
      RowCount > 10,
      Completeness "customer_identifier" > 0.5
  ]
"""

val EvaluateDataQuality_node1741974822533 =
 EvaluateDataQuality.processRows(frame=ChangeSchema_node1742850392012,
 ruleset=EvaluateDataQuality_node1741974822533_ruleset,
 publishingOptions=JsonOptions("""{"dataQualityEvaluationContext":
 "EvaluateDataQuality_node1741974822533", "enableDataQualityCloudWatchMetrics":
 "true", "enableDataQualityResultsPublishing": "true"}"""),
 additionalOptions=JsonOptions("""{"compositeRuleEvaluation.method":"ROW","observations.scope":
 "publishAggregatedMetrics.status": "ENABLED"}""" ))

println("---------------------------------ROW LEVEL
 OUTCOMES---------------------------------")
val rowLevelOutcomes_node = EvaluateDataQuality_node1741974822533("rowLevelOutcomes")

rowLevelOutcomes_node.show(10)

 println("---------------------------------RULE LEVEL
 OUTCOMES---------------------------------")

val ruleOutcomes_node = EvaluateDataQuality_node1741974822533("ruleOutcomes")

ruleOutcomes_node.show()

 println("---------------------------------AGGREGATED
 METRICS---------------------------------")

val aggregatedMetrics_node = EvaluateDataQuality_node1741974822533("aggregatedMetrics")
```

```
aggregatedMetrics_node.show()
```

샘플 결과

결과는 다음과 같이 반환됩니다.


```
{
    "Rule": "IsUnique \"customer_identifier\"",
    "Outcome": "Passed",
    "FailureReason": null,
    "EvaluatedMetrics": {
        "Column.customer_identifier.Uniqueness": 1
    },
    "EvaluatedRule": "IsUnique \"customer_identifier\"",
    "PassedCount": 10,
    "FailedCount": 0,
    "SkippedCount": 0,
    "TotalCount": 10
}
{
    "Rule": "RowCount > 10",
    "Outcome": "Failed",
    "FailureReason": "Value: 10 does not meet the constraint requirement!",
    "EvaluatedMetrics": {
        "Dataset.*.RowCount": 10
    },
    "EvaluatedRule": "RowCount > 10",
    "PassedCount": 0,
    "FailedCount": 0,
    "SkippedCount": 10,
    "TotalCount": 10
}
{
    "Rule": "Completeness \"customer_identifier\" > 0.5",
    "Outcome": "Passed",
    "FailureReason": null,
    "EvaluatedMetrics": {
        "Column.customer_identifier.Completeness": 1
    },
    "EvaluatedRule": "Completeness \"customer_identifier\" > 0.5",
    "PassedCount": 10,
    "FailedCount": 0,
    "SkippedCount": 0,
```

```
    "TotalCount": 10
}
```

집계 지표는 다음과 같습니다.

```
{ "TotalRowsProcessed": 10, "PassedRows": 10, "FailedRows": 0, "TotalRulesProcessed":
 3, "RulesPassed": 2, "RulesFailed": 1 }
```

데이터 품질 규칙 작성기

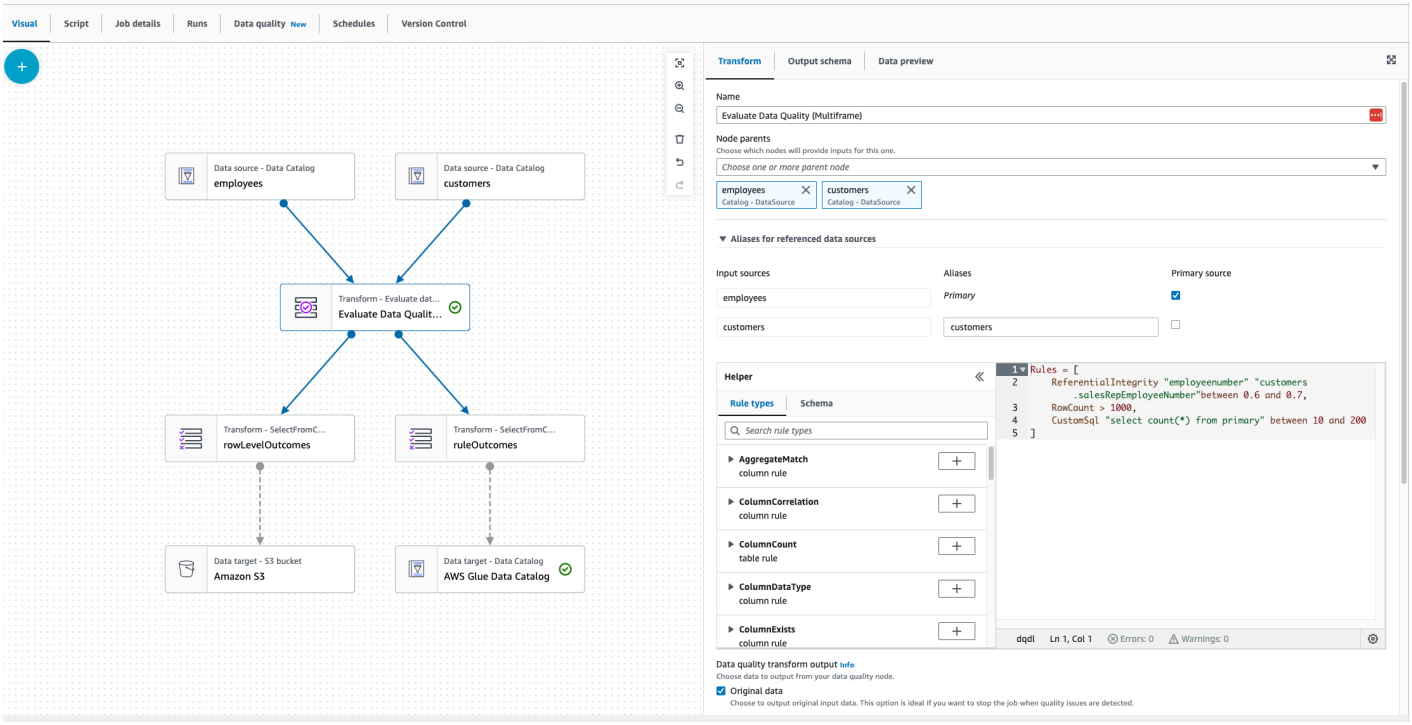
데이터 품질 정의 언어(DQDL) 규칙 작성기를 사용하면 데이터 품질 규칙을 생성하여 데이터를 평가할 수 있습니다. 먼저 규칙 유형을 선택하고 규칙 편집기에서 파라미터를 지정합니다. 규칙 편집기에는 규칙을 생성할 때 발생하는 오류 및 경고도 표시됩니다.

DQDL 안내서에서는 DQDL 구문, 기본 제공 규칙 유형 및 예제를 사용하여 규칙을 구성하는 방법에 대한 포괄적인 문서를 제공합니다.

데이터 품질 평가 노드

데이터 품질 평가 변환 노드 및 DQDL 규칙 작성기를 사용하여 작업 공간을 확장할 수 있습니다.

- 변환 탭을 확장하여 전체 화면을 채우려면 노드 세부 정보 패널의 오른쪽 상단에 있는 확장 아이콘을 선택합니다.
- DQDL 규칙 편집기를 확장하려면 << 아이콘을 선택하여 규칙 편집기를 확장하고 규칙 유형 및 스키마 탭을 축소합니다.



구성 요소

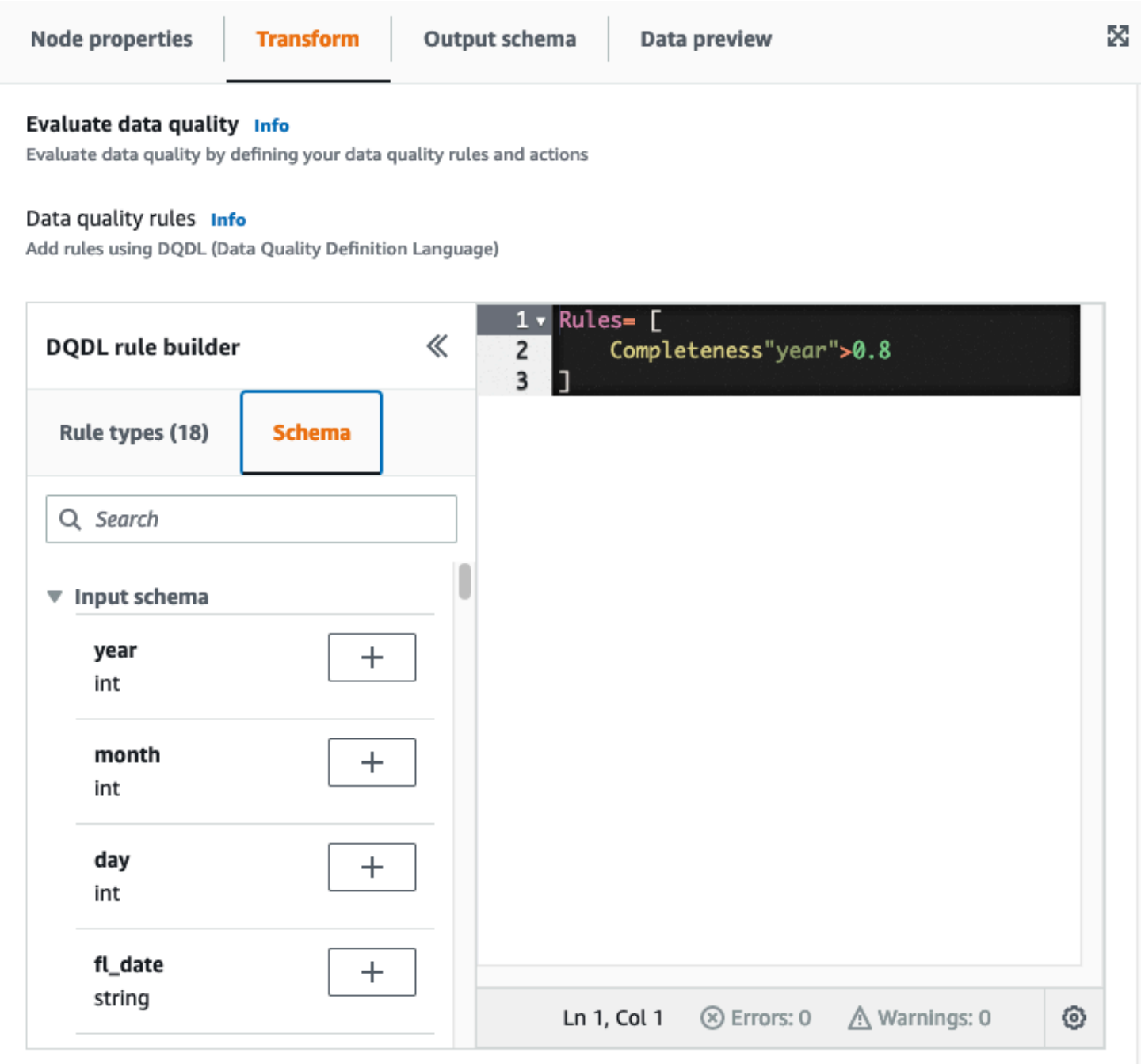
26개의 규칙 유형이 AWS Glue Studio에 기본 제공됩니다. 각 규칙 유형에는 사용 방법에 대한 설명과 예제가 있습니다.

데이터 품질 규칙 유형

AWS Glue Studio에서는 규칙을 쉽게 만들 수 있도록 기본 제공 규칙 유형을 제공합니다. 규칙 유형에 대한 자세한 내용은 DQDL 규칙 유형 참조를 참조하세요.

스키마

Schema(스키마) 탭에는 상위 노드의 열 이름과 데이터 유형이 표시됩니다. 여러 노드의 스키마가 표시됩니다. 입력 스키마를 보고, 열 이름을 기준으로 검색하고, 해당 열을 규칙 편집기에 삽입할 수 있습니다.



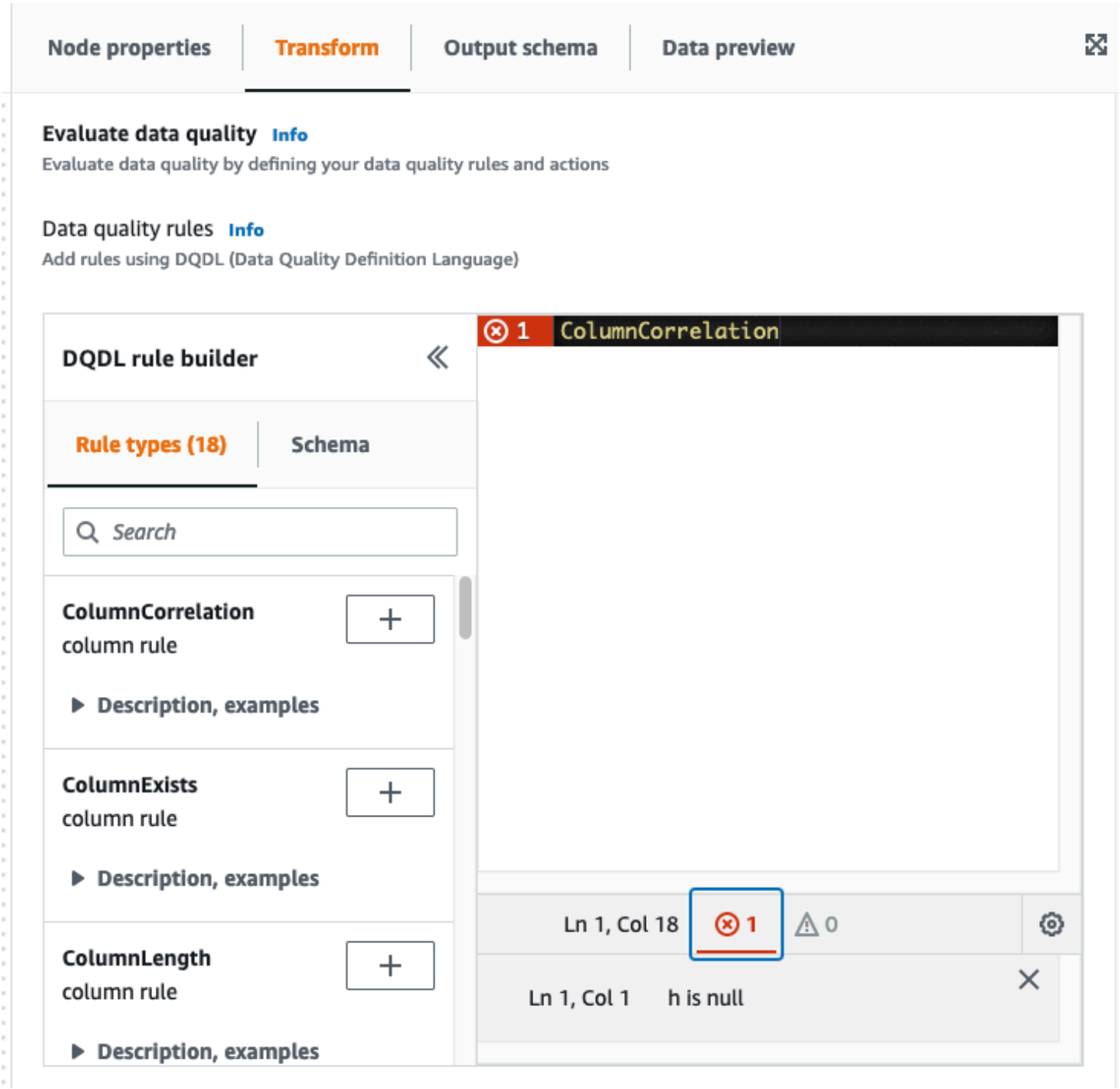
규칙 편집기

규칙 편집기는 규칙을 작성하고 편집할 수 있는 텍스트 편집기입니다. DQDL 규칙 작성기에서 규칙 유형을 선택하면 규칙 유형이 규칙 편집기에 추가됩니다. 그런 다음 텍스트를 수정하여 필요에 따라 파라미터를 지정하고 규칙을 추가하며 규칙을 편집할 수 있습니다. AWS Glue Studio에서는 규칙 편집기에서 규칙을 검증하고 오류 및 경고가 있을 경우 이를 표시합니다.

Errors and warnings(오류 및 경고)

규칙이 DQDL 규칙 구문을 따르지 않는 경우 규칙 편집기에 오류가 있음을 나타내는 몇 가지 시각적 표시기가 나타납니다.

- 규칙 편집기는 오류가 있는 행을 오류 아이콘과 함께 빨간색으로 표시합니다.
- 규칙 편집기의 빨간색 오류 아이콘 옆에 오류 수가 표시됩니다.
- 오류가 있는 행을 선택하면 오류 설명과 위치(행 및 열)가 규칙 편집기 아래쪽에 표시됩니다.



데이터 품질 작업

기본적으로 이 작업은 선택되지 않으며 데이터 품질 규칙이 실패하더라도 작업 실행이 완료됩니다.

다음 작업 중에서 선택합니다. 작업을 사용하여 결과를 CloudWatch에 게시하거나 특정 기준에 따라 작업을 중지할 수 있습니다. 작업은 규칙을 생성한 경우에만 사용할 수 있습니다.

- CloudWatch에 결과 게시 - 작업을 실행할 때 결과를 CloudWatch에 추가합니다.
- 데이터 품질이 실패하면 작업 실패 - 데이터 품질 규칙이 실패하면 결과적으로 작업도 실패합니다.

데이터 품질 변환 출력

- 원래 데이터 - 원래 입력 데이터를 출력하려면 선택합니다. 이 옵션은 품질 문제가 감지되었을 때 작업을 중지하려는 경우에 적합합니다.
- 데이터 품질 지표 - 구성된 규칙과 규칙의 통과 또는 실패 상태를 출력하려면 선택합니다. 이 옵션은 사용자 지정 작업을 수행하려는 경우에 유용합니다.

데이터 품질 출력 설정

Amazon S3 위치를 데이터 품질 출력 대상으로 지정하여 데이터 품질 결과 위치를 설정합니다.

AWS Glue ETL 작업에서 이상 탐지 구성

AWS Glue Studio에서 이상 탐지를 시작하려면 AWS Glue Studio 작업을 열고 데이터 품질 평가 변환을 클릭합니다.

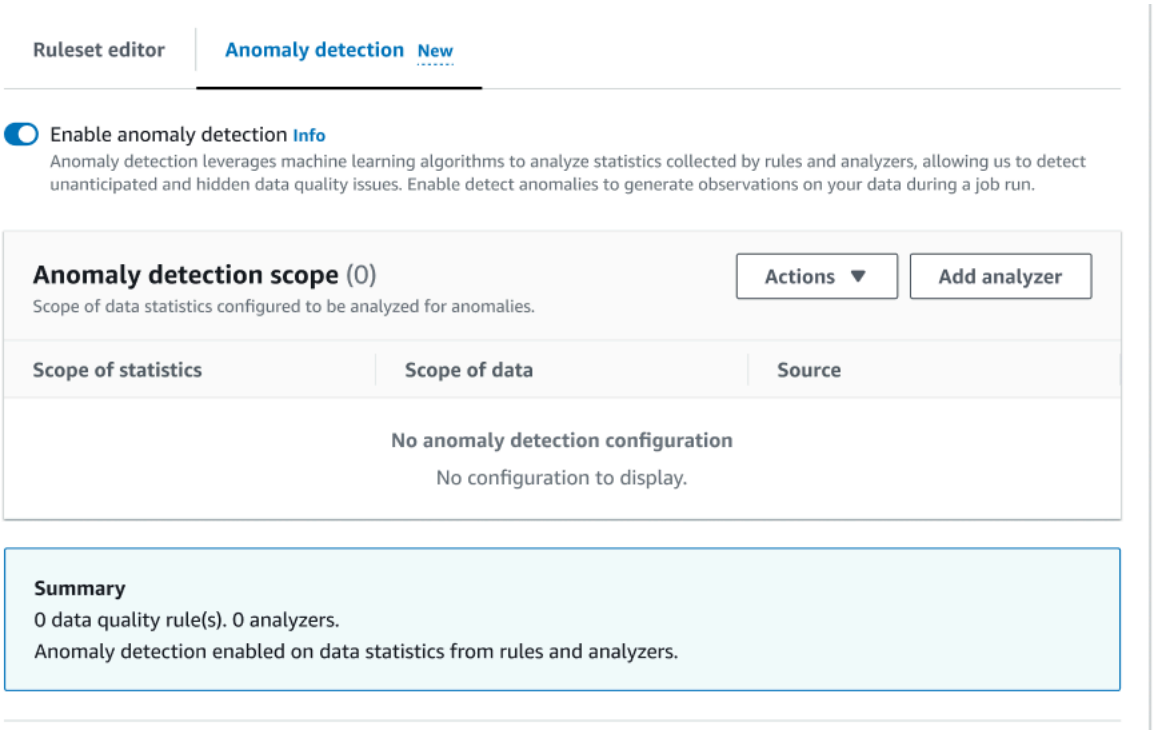
이 기능을 활성화하면 AWS Glue Data Quality가 시간 경과에 따라 데이터를 분석하여 이상을 탐지합니다. 데이터에 대한 중요한 데이터 통계 및 관찰 결과를 제공하므로 식별된 이상에 대해 조치를 취할 수 있습니다.

이 기능의 내부 작동 방식을 이해하려면 이상 탐지 설명서를 참조하세요.

이상 탐지 활성화

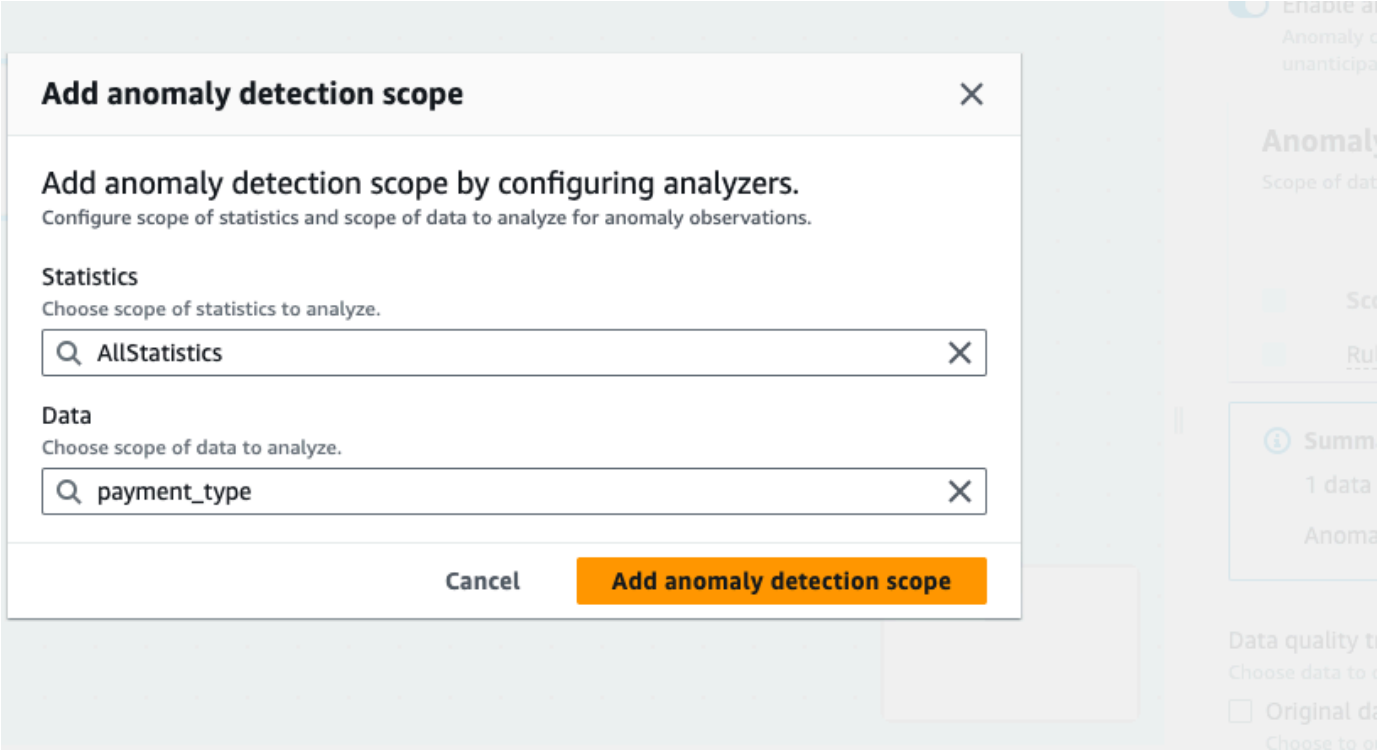
AWS Glue Studio에서 이상 탐지를 활성화하려면 다음을 수행합니다.

1. 작업에서 Data Quality 노드를 선택한 다음 이상 탐지 탭을 선택합니다. 토글하여 이상 탐지 활성화를 켭니다.



2. 분석기 추가를 선택하여 이상을 모니터링할 데이터를 정의합니다. 입력할 수 있는 두 가지 필드는 통계와 데이터입니다.

   - 통계는 데이터의 형태 및 기타 속성에 대한 정보입니다. 한 번에 하나 이상의 통계를 선택하거나 모든 통계를 선택할 수 있습니다. 통계에는 완전성, 고유성, 평균, 합계, 표준편차, 엔트로피, 개별 값 수, 고유값 비율 등이 포함됩니다. 자세한 내용은 분석기 설명서를 참조하세요.
   - 데이터는 데이터 세트의 열입니다. 모든 열 또는 개별 열을 선택할 수 있습니다.



3. 이상 탐지 범위 추가를 선택하여 변경 사항을 저장합니다. 분석기를 추가하고 나면 이상 탐지 범위 섹션에서 해당 분석기를 확인할 수 있습니다.

   작업 메뉴를 사용하여 분석기를 편집하거나 규칙 세트 편집기 탭을 선택하고 규칙 세트 편집기 메모장에서 직접 분석기를 편집할 수도 있습니다. 생성한 규칙 바로 아래에 저장한 분석기가 표시됩니다.

```
Rules = [

]

Analyzers = [
    Completeness “id”
]
```

업데이트된 규칙 세트와 분석기가 구성되면 AWS Glue Data Quality가 들어오는 데이터 스트림을 지속적으로 모니터링합니다. 설정에 따라 알림 또는 작업 중지를 통해 잠재적 이상을 알릴 수 있습니다. 이러한 사전 예방적 모니터링은 데이터 파이프라인 전반에서 데이터 품질과 무결성을 보장하는 데 도움이 됩니다.

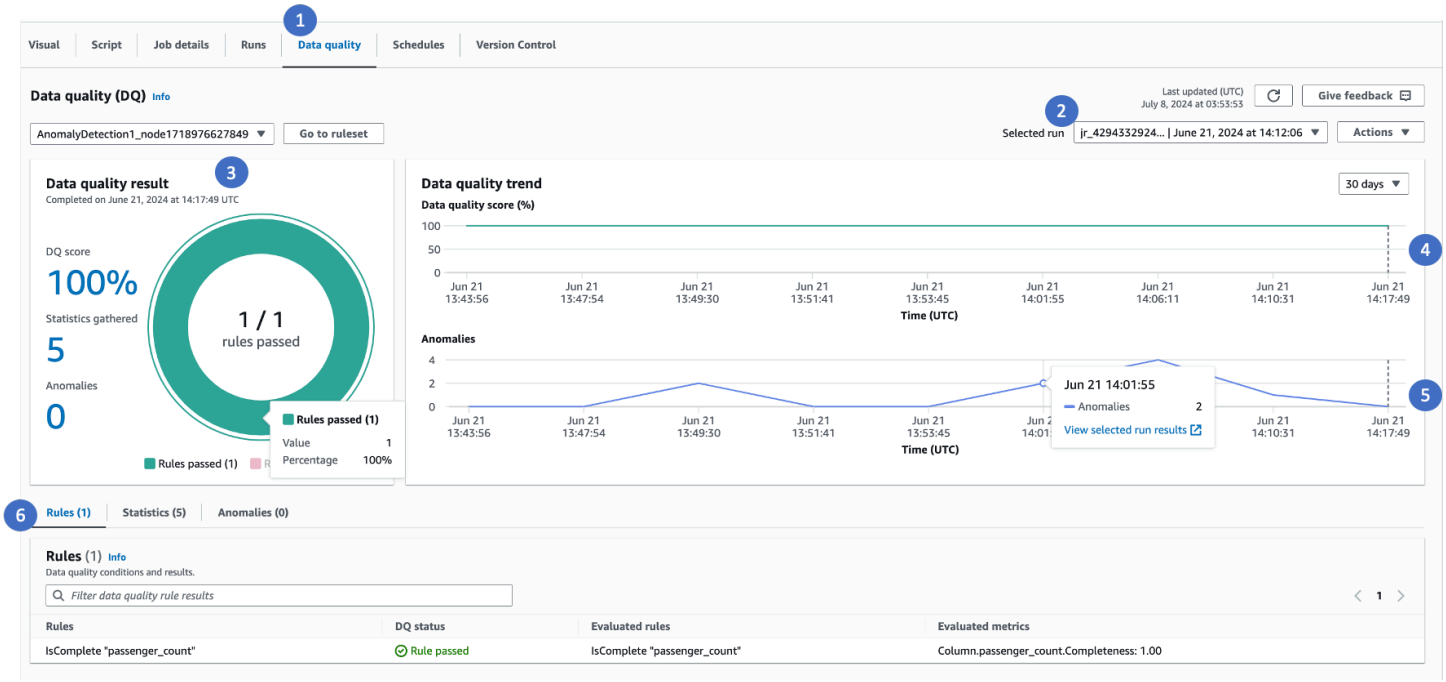
다음 섹션에서는 시스템에서 식별되는 이상을 효과적으로 모니터링하는 방법을 알아봅니다. AWS Glue Data Quality에서 수집한 데이터 통계를 보고 분석하는 방법도 알아봅니다. 아울러 이상 탐지 기능을 지원하는 기계 학습 모델에 피드백을 제공하는 방법도 이해하게 됩니다. 이 피드백 루프는 모델의 정확도를 높이고 특정 비즈니스 요구 사항 및 데이터 패턴에 부합하는 이상을 효과적으로 탐지할 수 있도록 하는 데 매우 중요합니다.

데이터 품질 점수 및 이상 보기

이 섹션에서는 데이터 품질 대시보드와 이 대시보드에서 제공하는 다양한 기능을 살펴보겠습니다.

높은 수준의 데이터 품질 지표와 추세를 시각화하고 이해합니다.

작업이 성공하면 데이터 품질 탭을 선택하여 데이터 품질 점수와 이상을 확인하세요.

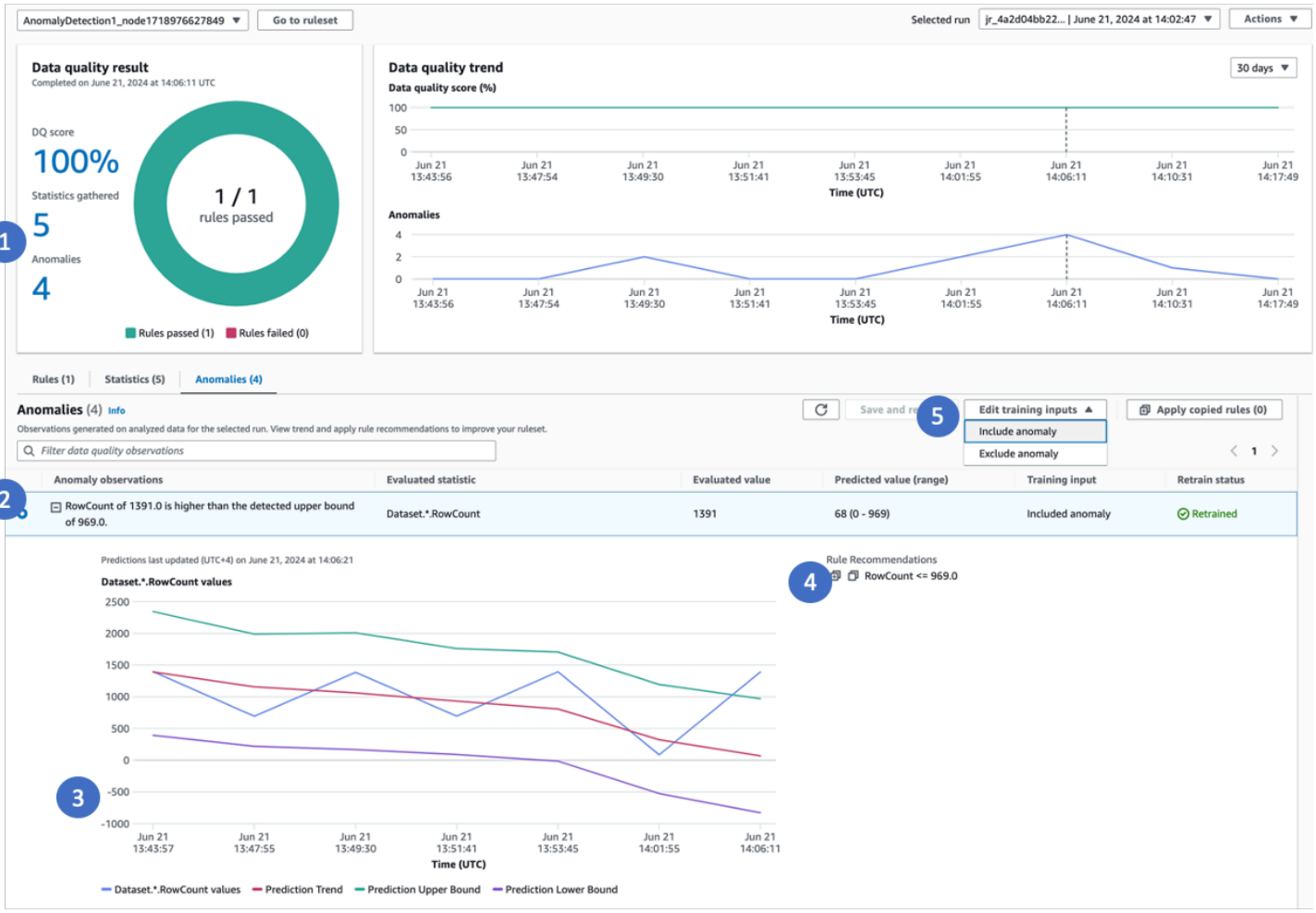


데이터 품질 탭의 다음 구성 요소에서는 유용한 정보를 확인할 수 있습니다.

1. 데이터 품질 지표를 보려면 데이터 품질 탭을 선택합니다.
2. 데이터 품질 점수를 보려면 특정 작업 실행 ID를 선택합니다.
3. 이 창에는 세 가지 중요한 정보가 표시됩니다. 각각을 선택하여 특정 테이블로 이동하면 이상, 데이터 통계 또는 규칙을 볼 수 있습니다.
   - 규칙 구성 시 데이터 품질 점수
   - 규칙 및 분석기에서 수집한 통계 수
   - 탐지된 총 이상 수

4. 이 추세 차트는 시간 경과에 따른 데이터 품질 추세를 보여줍니다. 추세를 마우스로 가리키면 데이터 품질 점수가 하락한 특정 시점으로 이동할 수 있습니다.
5. 시간 경과에 따른 이상 추세는 시간 경과에 따라 탐지된 이상의 수를 보여줍니다.
6. 탭:
   - 규칙 탭은 모든 규칙 및 상태 목록을 보여주는 기본 탭입니다. 평가된 규칙은 동적 규칙의 경우 규칙을 평가한 실제 결과 값을 보는 데 유용합니다.
   - 통계 탭에는 모든 통계가 나열되므로 시간 경과에 따른 지표와 추세를 볼 수 있습니다.
   - 이상 탭에는 탐지된 이상 목록이 표시됩니다.

이상 보기 및 이상 탐지 알고리즘 훈련



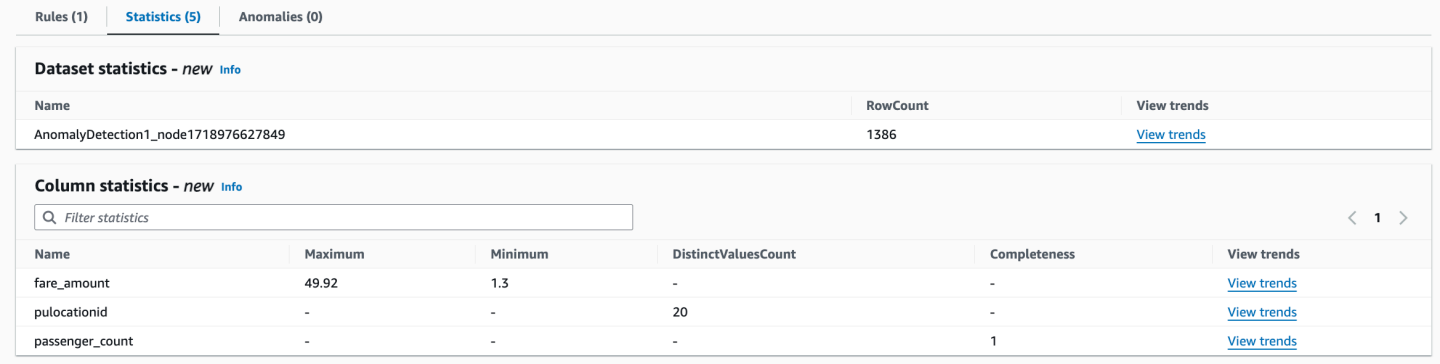
위의 이미지에 대한 콜아웃:

1. 이상이 탐지되면 이상을 클릭하거나 이상 탭을 선택합니다.
2. AWS Glue Data Quality가 이상, 실제 값, 예측 범위에 대한 자세한 설명을 제공합니다.

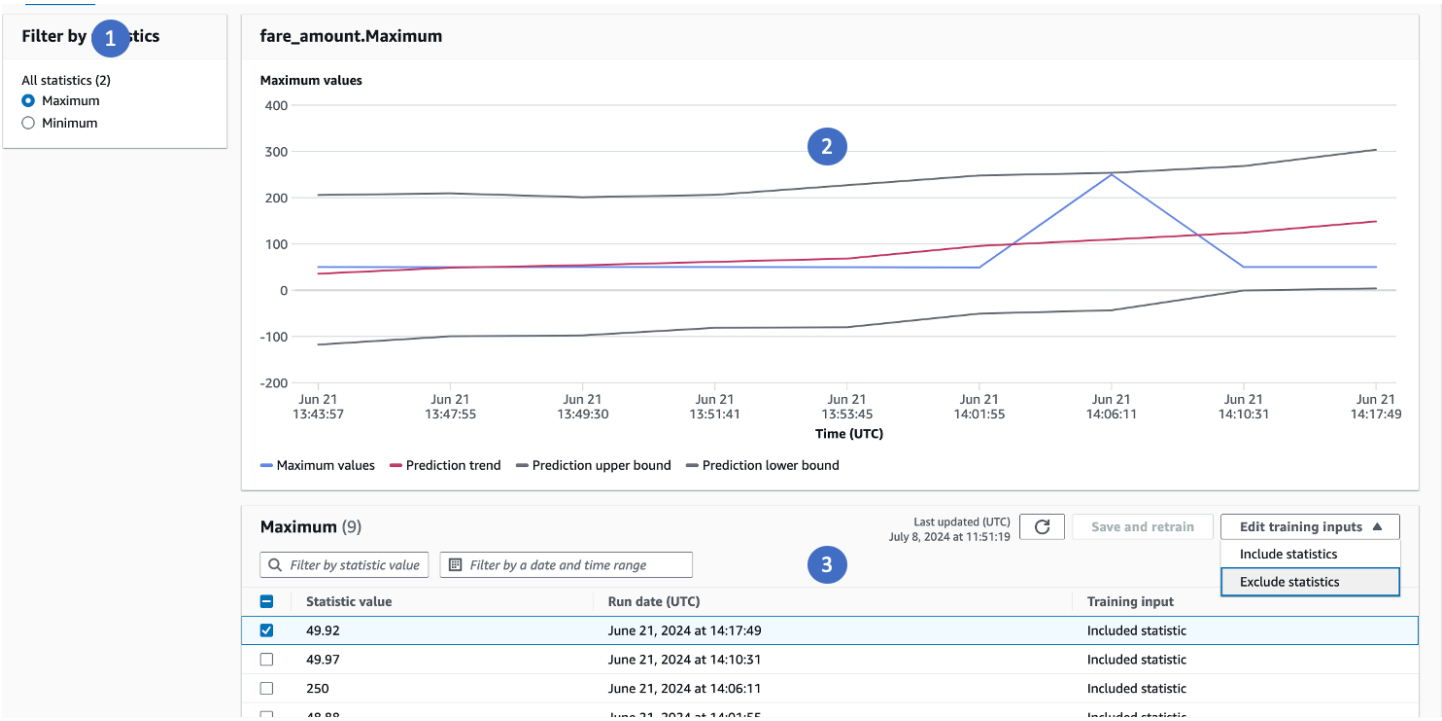
3. AWS Glue Data Quality이 추세선을 보여줍니다. 여기에는 실제 값, 실제 값을 기반으로 도출된 추세(빨간색 선), 상한 및 하한이 있습니다.
4. AWS Glue Data Quality가 미래의 패턴을 캡처하는 데 사용할 수 있는 데이터 품질 규칙을 추천합니다. 권장되는 모든 규칙을 복사하고 데이터 품질 노드에 적용하여 이러한 패턴을 효과적으로 캡처할 수 있습니다.
5. 기계 학습(ML) 모델에 입력을 제공하여 이상 값을 제외함으로써 향후 실행에서 이상을 정확하게 탐지하도록 할 수 있습니다. 이상을 명시적으로 제외하지 않으면 AWS Glue Data Quality가 자동으로 향후 예측 시에 이를 모델의 일부로 간주합니다. 입력한 모델 입력은 최신 실행에만 반영된다는 점에 유의해야 합니다. 예를 들어 이전 실행으로 돌아가서 몇 번의 실행에서 이상 포인트를 제외한 경우, 최근 실행에서 모델 입력을 확인하고 업데이트하지 않는 한 모델에 해당 변경 사항이 반영되지 않습니다. 가장 최근 실행에서 필요한 조정을 할 때까지 모델은 이전에 제공된 입력을 계속 사용합니다. 이상 값 제외를 적극적으로 관리하면 특정 데이터 패턴 및 요구 사항의 이상을 구성하는 요소에 대한 ML 모델의 이해도를 높여, 시간이 지남에 따라 이상을 더 정확하게 탐지하도록 할 수 있습니다.

시간 경과에 따른 데이터 통계 보기 및 훈련 입력 제공

데이터 통계 또는 데이터 프로필을 보고 시간 경과에 따라 어떻게 변화하는지 확인해야 할 경우도 있습니다. 이렇게 하려면 통계를 선택하거나 통계 탭을 엽니다. 그러면 AWS Glue Data Quality에서 수집한 최신 데이터 통계를 볼 수 있습니다.



추세 보기를 클릭하면 시간 경과에 따른 각 통계의 변화를 확인할 수 있습니다.



1. 특정 열의 통계를 선택할 수 있습니다.
2. 추세가 어떻게 변화하는지 확인할 수 있습니다.
3. 이상 값을 선택하고 제외하거나 포함하도록 선택할 수 있습니다. 이 피드백을 제공하면 알고리즘이 식별된 이상 데이터 포인트를 제외하거나 포함시키고 모델을 재훈련합니다. 사용자가 제공한 피드백을 통해 어떤 값을 이상으로 간주해야 하는지 아닌지를 모델이 학습하므로, 이 재훈련 프로세스를 거치면서 이상을 정확하게 탐지할 수 있게 됩니다.

   이 피드백 루프를 통해 특정 데이터 패턴 및 비즈니스 요구 사항에 맞는 이상을 구성하는 요소에 대한 알고리즘의 이해도를 높일 수 있습니다. 이상으로 표시해서는 안 되는 값을 제외하거나 누락된 값을 포함함으로써 재훈련된 모델은 예상 데이터 포인트와 실제 이상 데이터 포인트를 더 잘 구분하게 됩니다.

AWS Glue Studio 노트북에서 ETL 작업에 대한 데이터 품질

이 자습서에서는 AWS Glue Studio 노트북에서 추출, 전환, 적재(ETL) 작업에 대해 AWS Glue Data Quality를 사용하는 방법을 알아봅니다.

AWS Glue Studio에서 노트북을 사용하여 전체 작업을 실행하지 않고도 작업 스크립트를 편집하고 출력을 볼 수 있습니다. 마크다운을 추가하고 노트북을 `.ipynb` 파일 및 작업 스크립트로 저장할 수도 있습니다. 소프트웨어를 로컬로 설치하거나 서버를 관리하지 않고도 노트북을 시작할 수 있습니다. 코드

가 만족스러우면 AWS Glue Studio를 사용하여 노트북을 AWS Glue 작업으로 쉽게 전환할 수 있습니다.

이 예제에서 사용된 데이터 세트는 두 Data.CMS.gov 데이터 세트('Inpatient Prospective Payment System Provider Summary for the Top 100 Diagnosis-Related Groups - FY2011' 및 'Inpatient Charge Data FY 2011')에서 다운로드한 Medicare Provider 지불 데이터로 구성됩니다.

데이터 다운로드 후 데이터 집합을 수정하여 파일 끝에 몇 가지 잘못된 기록을 소개합니다. 이 수정된 파일은 `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv`의 퍼블릭 Amazon S3 버킷에 있습니다.

사전 조건

- 대상 Amazon S3 버킷에 대한 Amazon S3 쓰기 권한이 있는 AWS Glue 역할
- 새 노트북(AWS Glue Studio에서 노트북 시작하기 참조)

AWS Glue Studio에서 ETL 작업 생성

ETL 작업을 생성하려면

1. 세션 버전을 AWS Glue 3.0으로 변경합니다.

   이렇게 하려면 다음과 같은 매직으로 모든 표준 문안 코드 셀을 제거하고 셀을 실행합니다. 이 표준 문안 코드는 새 노트북이 생성되면 첫 번째 셀에 자동으로 제공됩니다.

   ```
   %glue_version 3.0
   ```

2. 다음 코드를 복사하여 셀에서 실행합니다.

   ```
   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

   sc = SparkContext.getOrCreate()
   glueContext = GlueContext(sc)
   spark = glueContext.spark_session
   job = Job(glueContext)
   ```

3. 다음 셀에서는 AWS Glue Data Quality를 평가하는 `EvaluateDataQuality` 클래스를 가져옵니다.

```
from awsgluedq.transforms import EvaluateDataQuality
```

4. 다음 셀에서는 퍼블릭 Amazon S3 버킷에 저장된.csv 파일을 사용하여 소스 데이터를 읽습니다.

```
medicare = spark.read.format(
"csv").option(
"header", "true").option(
"inferSchema", "true").load(
's3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv')
medicare.printSchema()
```

5. 데이터를 AWS Glue DynamicFrame으로 변환합니다.

```
from awsglue.dynamicframe import DynamicFrame
medicare_dyf = DynamicFrame.fromDF(medicare,glueContext,"medicare_dyf")
```

6. 데이터 품질 정의 언어(DQDL)를 사용하여 규칙 세트를 생성합니다.

```
EvaluateDataQuality_ruleset = """
    Rules = [
        ColumnExists "Provider Id",
        IsComplete "Provider Id",
        ColumnValues  " Total Discharges " > 15
]
    ]
"""
```

7. 규칙 세트를 기준으로 데이터 세트를 검증합니다.

```
EvaluateDataQualityMultiframe = EvaluateDataQuality().process_rows(
    frame=medicare_dyf,
    ruleset=EvaluateDataQuality_ruleset,
    publishing_options={
        "dataQualityEvaluationContext": "EvaluateDataQualityMultiframe",
```

```
        "enableDataQualityCloudWatchMetrics": False,
        "enableDataQualityResultsPublishing": False,
    },
    additional_options={"performanceTuning.caching": "CACHE_NOTHING"},
)
```

8. 결과를 검토합니다.

```
ruleOutcomes = SelectFromCollection.apply(
    dfc=EvaluateDataQualityMultiframe,
    key="ruleOutcomes",
    transformation_ctx="ruleOutcomes",
)

ruleOutcomes.toDF().show(truncate=False)
```

출력:

```
-------------------------------------+-------
+-----------------------------------------------------
+-----------------------------------------+
|Rule                                 |Outcome|FailureReason
                  |EvaluatedMetrics                           |
+-------------------------------------+-------
+-----------------------------------------------------
+-----------------------------------------+
|ColumnExists "Provider Id"           |Passed |null
                  |{}                                         |
|IsComplete "Provider Id"             |Passed |null
                  |{Column.Provider Id.Completeness -> 1.0}   |
|ColumnValues " Total Discharges " > 15|Failed |Value: 11.0 does not meet the
 constraint requirement!|{Column. Total Discharges .Minimum -> 11.0}|
+-------------------------------------+-------
+-----------------------------------------------------
+-----------------------------------------+
```

9. Data Quality 행 수준 결과에서 통과된 행을 필터링하고 실패한 행을 검토합니다.

```
owLevelOutcomes = SelectFromCollection.apply(
```

```
dfc=EvaluateDataQualityMultiframe,
key="rowLevelOutcomes",
transformation_ctx="rowLevelOutcomes",
)

rowLevelOutcomes_df = rowLevelOutcomes.toDF() # Convert Glue DynamicFrame to
 SparkSQL DataFrame
rowLevelOutcomes_df_passed =
 rowLevelOutcomes_df.filter(rowLevelOutcomes_df.DataQualityEvaluationResult ==
 "Passed") # Filter only the Passed records.
rowLevelOutcomes_df.filter(rowLevelOutcomes_df.DataQualityEvaluationResult ==
 "Failed").show(5, truncate=False) # Review the Failed records
```

출력:

```
+----------------------------------------+-----------
+-----------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+----------------------
+--------------------------+-------------------------
+----------------------------------------+--------------------------
+---------------------------+
|DRG Definition                           |Provider Id|Provider Name
           |Provider Street Address   |Provider City|Provider State|Provider Zip
 Code|Hospital Referral Region Description| Total Discharges | Average Covered
 Charges | Average Total Payments |Average Medicare Payments|DataQualityRulesPass
      |DataQualityRulesFail                    |DataQualityRulesSkip        |
DataQualityEvaluationResult|
+----------------------------------------+-----------
+-----------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+----------------------
+--------------------------+-------------------------
+----------------------------------------+--------------------------
+---------------------------+
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10005      |MARSHALL MEDICAL CENTER SOUTH
        |2505 U S HIGHWAY 431 NORTH|BOAZ         |AL            |35957
 |AL - Birmingham                      |14               |$15131.85
  |$5787.57               |$4976.71                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
```

```
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10046      |RIVERVIEW REGIONAL MEDICAL
 CENTER    |600 SOUTH THIRD STREET    |GADSDEN      |AL            |35901
   |AL - Birmingham                      |14               |$67327.92
   |$5461.57               |$4493.57                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10083      |SOUTH BALDWIN REGIONAL
 MEDICAL CENTER|1613 NORTH MCKENZIE STREET|FOLEY        |AL            |36535
        |AL - Mobile                          |15               |$25411.33
         |$5282.93               |$4383.73                |[IsComplete "Provider
 Id"]|[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
                    |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|30002      |BANNER GOOD SAMARITAN MEDICAL
 CENTER |1111 EAST MCDOWELL ROAD   |PHOENIX      |AZ            |85006
 |AZ - Phoenix                         |11               |$34803.81
  |$7768.90               |$6951.45                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|30010      |CARONDELET ST  MARYS HOSPITAL
        |1601 WEST ST MARY'S ROAD  |TUCSON       |AZ            |85745
 |AZ - Tucson                          |12               |$35968.50
  |$6506.50               |$5379.83                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
+---------------------------------------+-----------
+-------------------------------------+-------------------------+-------------
+-------------+----------------+-----------------------------------
+-----------------+-------------------------+----------------------
+-------------------------+-------------------------
+---------------------------------------+---------------------------
+--------------------------+
only showing top 5 rows
```

AWS Glue Data Quality에서는 새 열 네 개(DataQualityRulesPass, DataQualityRulesFail, DataQualityRulesSkip, DataQualityEvaluationResult)를 추가했습니다. 이는 통과한 레코드, 실패한 레코드, 행 수준 평가에서 건너뛴 규칙, 전체 행 수준 결과를 나타냅니다.

10. 출력을 Amazon S3 버킷에 기록하여 데이터를 분석하고 결과를 시각화합니다.

```
#Write the Passed records to the destination.

glueContext.write_dynamic_frame.from_options(
```

```
        frame = rowLevelOutcomes_df_passed,
        connection_type = "s3",
        connection_options = {"path": "s3://glue-sample-target/output-dir/
medicare_parquet"},
        format = "parquet")
```

데이터 품질 정의 언어(DQDL) 참조

DQDL(데이터 품질 정의 언어)은 AWS Glue 데이터 품질에 대한 규칙을 정의하는 데 사용되는 도메인 별 언어입니다.

이 안내서에서는 언어를 이해하는 데 도움이 되는 주요 DQDL 개념을 소개합니다. 또한 구문 및 예제와 함께 DQDL 규칙 유형에 대한 참조를 제공합니다. 이 안내서를 사용하기 전에 AWS Glue 데이터 품질을 숙지하는 것이 좋습니다. 자세한 내용은 AWS Glue Data Quality 단원을 참조하십시오.

> **Note**
>
> DynamicRule은 AWS Glue ETL에서만 지원됩니다.

목차

- DQDL 구문
  - 규칙 구조
  - 복합 규칙
    - 복합 규칙의 작동 방식
  - Expressions
    - NULL, EMPTY, WHITESPACES_ONLY에 대한 키워드
    - Where 절을 사용한 필터링
  - 동적 규칙
  - 분석기
  - 설명
- DQDL 규칙 유형 참조
  - AggregateMatch
  - ColumnCorrelation

- ColumnCount
- ColumnDataType
- ColumnExists
- ColumnLength
- ColumnNamesMatchPattern
- ColumnValues
- 완전성
- CustomSQL
- DataFreshness
- DatasetMatch
- DistinctValuesCount
- Entropy
- IsComplete
- IsPrimaryKey
- IsUnique
- Mean
- ReferentialIntegrity
- RowCount
- RowCountMatch
- StandardDeviation
- Sum
- SchemaMatch
- Uniqueness
- UniqueValueRatio
- DetectAnomalies
- FileFreshness
- FileMatch
- FileUniqueness
- FileSize

DQDL 구문

DQDL 문서는 대/소문자를 구분하며 개별 데이터 품질 규칙을 그룹화하는 규칙 세트를 포함합니다. 규칙 세트를 구성하려면 한 쌍의 대괄호로 구분된 Rules(대문자) 목록을 생성해야 합니다. 목록에는 다음 예와 같이 쉼표로 구분된 DQDL 규칙이 하나 이상 포함되어야 합니다.

```
Rules = [
   IsComplete "order-id",
   IsUnique "order-id"
]
```

규칙 구조

DQDL 규칙의 구조는 규칙 유형에 따라 달라집니다. 하지만 DQDL 규칙은 일반적으로 다음 형식에 적합합니다.

```
<RuleType> <Parameter> <Parameter> <Expression>
```

RuleType은 구성하려는 규칙 유형의 이름(대/소문자 구분)입니다. 예: IsComplete, IsUnique 또는 CustomSql. 규칙 파라미터는 규칙 유형마다 다릅니다. DQDL 규칙 유형 및 파라미터에 대한 전체 참조는 DQDL 규칙 유형 참조 섹션을 참조하세요.

복합 규칙

DQDL은 규칙을 결합하는 데 사용될 수 있는 다음과 같은 논리 연산자를 지원합니다. 이러한 규칙을 복합 규칙이라고 합니다.

and

논리 and 연산자는 연결하는 규칙이 true인 경우에만 true 결과를 얻습니다. 그렇지 않으면 결합된 규칙에 따라 false 결과를 얻습니다. and 연산자와 연결하는 각 규칙은 괄호로 묶어야 합니다.

다음 예제에서는 and 연산자를 사용하여 두 DQDL 규칙을 결합합니다.

```
(IsComplete "id") and (IsUnique "id")
```

or

논리 or 연산자는 연결하는 규칙 중 하나 이상이 true인 경우에만 true 결과를 얻습니다. or 연산자와 연결하는 각 규칙은 괄호로 묶어야 합니다.

다음 예제에서는 or 연산자를 사용하여 두 DQDL 규칙을 결합합니다.

```
(RowCount "id" > 100) or (IsPrimaryKey "id")
```

동일한 연산자를 사용하여 여러 규칙을 연결할 수 있으므로 다음과 같은 규칙 조합이 허용됩니다.

```
(Mean "Star_Rating" > 3) and (Mean "Order_Total" > 500) and (IsComplete "Order_Id")
```

논리 연산자를 단일 표현식으로 결합할 수 없습니다. 예제:

```
(Mean "Star_Rating" > 3) and ((Mean "Order_Total" > 500) or (IsComplete "Order_Id"))
```

더 복잡하고 중첩된 규칙을 작성할 수도 있습니다.

```
(RowCount > 0) or ((IsComplete "colA") and (IsUnique "colA"))
```

복합 규칙의 작동 방식

기본적으로 복합 규칙은 전체 데이터세트 또는 테이블에서 개별 규칙으로 평가된 다음 결과가 결합됩니다. 즉, 전체 열을 먼저 평가한 다음 연산자를 적용합니다. 이 기본 동작은 예제와 함께 아래에 설명되어 있습니다.

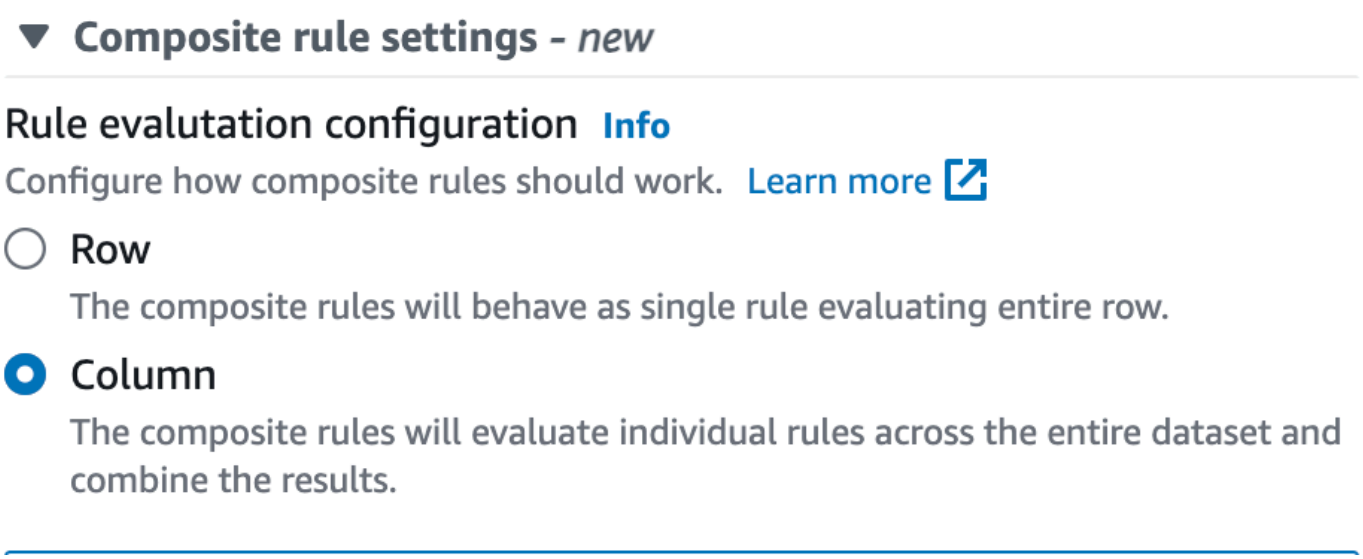
```
# Dataset

+------+------+
|myCol1|myCol2|
+------+------+
|     2|     1|
|     0|     3|
+------+------+

# Overall outcome

+-------------------------------------------------------------+-------+
|Rule                                                         |Outcome|
+-------------------------------------------------------------+-------+
|(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)|Failed |
+-------------------------------------------------------------+-------+
```

위의 예제에서 AWS Glue Data Quality는 먼저 (ColumnValues "myCol1" > 1)을 평가하여 실패하게 됩니다. 그런 다음 (ColumnValues "myCol2" > 2)를 평가하며 역시 실패하게 됩니다. 두 결과의 조합은 FAILED로 표시됩니다.

그러나 전체 행을 평가해야 하는 SQL과 같은 동작을 선호하는 경우 아래 코드 스니펫의 additionalOptions에 표시된 것처럼 ruleEvaluation.scope 파라미터를 명시적으로 설정해야 합니다.

```
object GlueApp {
  val datasource = glueContext.getCatalogSource(
    database="<db>",
    tableName="<table>",
    transformationContext="datasource"
  ).getDynamicFrame()

  val ruleset = """
    Rules = [
        (ColumnValues "age" >= 26) OR (ColumnLength "name" >= 4)
    ]
  """

  val dq_results = EvaluateDataQuality.processRows(
    frame=datasource,
    ruleset=ruleset,
    additionalOptions=JsonOptions("""
        {
          "compositeRuleEvaluation.method":"ROW"
        }
      """
    )
  )
}
```

AWS Glue Data Catalog에서는 아래와 같이 사용자 인터페이스에서 이 옵션을 쉽게 구성할 수 있습니다.



일단 설정되면 복합 규칙은 전체 행을 평가하는 단일 규칙으로 작동합니다. 다음 예제는 이 동작을 보여 줍니다.

```
# Row Level outcome

+------+------+-------------------------------------------------------------------
+---------------------------+
|myCol1|myCol2|DataQualityRulesPass                                                |
DataQualityEvaluationResult|
+------+------+-------------------------------------------------------------------
+---------------------------+
|2     |1     |[(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)]|Passed
                |
|0     |3     |[(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)]|Passed
                |
+------+------+-------------------------------------------------------------------
+---------------------------+
```

일부 규칙은 임계값이나 비율에 따라 전체 결과가 달라지므로 이 기능에서 지원되지 않습니다. 이는 아래에 나열되어 있습니다.

비율에 따른 규칙:

- 완전성

- DatasetMatch
- ReferentialIntegrity
- Uniqueness

임계값에 따른 규칙:

다음 규칙에 with threshold가 포함된 경우 해당 규칙은 지원되지 않습니다. 그러나 `with threshold`를 포함하지 않는 규칙은 계속 지원됩니다.

- ColumnDataType
- ColumnValues
- CustomSQL

Expressions

규칙 유형이 부울 응답을 생성하지 않는 경우 부울 응답을 생성하려면 표현식을 파라미터로 제공해야 합니다. 예를 들어 다음 규칙은 열에 있는 모든 값의 중앙값(평균)을 표현식과 비교하여 true 또는 false 결과를 반환합니다.

```
Mean "colA" between 80 and 100
```

`IsUnique` 및 `IsComplete`와 같은 일부 규칙 유형은 이미 부울 응답을 반환합니다.

다음 표는 DQDL 규칙에서 사용할 수 있는 표현식을 나열합니다.

지원되는 DQDL 표현식

표현식	설명	예제
=*x*	규칙 유형 응답이 *x*와 같은 경우 `true`로 확인됩니다.	`Completeness "colA" = "1.0", ColumnValues "colA" = "2022-06-30"`
!=*x*	규칙 유형 응답이 *x*와 같지 않으면 x는 true로 확인됩니다.	`ColumnValues "colA" != "a", ColumnValues "colA" != "2022-06-30"`

표현식	설명	예제
> *x*	규칙 유형 응답이 *x*보다 큰 경우 true로 확인됩니다.	`ColumnValues "colA" > 10`
< *x*	규칙 유형 응답이 *x*보다 작은 경우 true로 확인됩니다.	`ColumnValues "colA" < 1000, ColumnValues "colA" < "2022-06-30"`
>= *x*	규칙 유형 응답이 *x*보다 크거나 같은 경우 true로 확인됩니다.	`ColumnValues "colA" >= 10`
<= *x*	규칙 유형 응답이 *x*보다 작거나 같은 경우 true로 확인됩니다.	`ColumnValues "colA" <= 1000`
between *x* and *y*	규칙 유형 응답이 지정된 범위(제외)에 속하는 경우 true로 확인됩니다. 이 표현식 유형은 숫자 및 날짜 유형에만 사용해야 합니다.	`Mean "colA" between 8 and 100, ColumnValues "colA" between "2022-05-31" and "2022-06-30"`
x 및 *y* 사이가 아님	규칙 유형 응답이 지정된 범위(포함)에 속하지 않으면 true로 확인됩니다. 이 표현식 유형은 숫자 및 날짜 유형에만 사용해야 합니다.	`ColumnValues "colA" not between "2022-05-31" and "2022-06-30"`
in [*a*, *b*, *c*, ...]	규칙 유형 응답이 지정된 세트 내에 있는 경우 true로 확인됩니다.	`ColumnValues "colA" in [ 1, 2, 3 ], ColumnValues "colA" in [ "a", "b", "c" ]`

표현식	설명	예제
not in [*a, b, c, ...*]	규칙 유형 응답이 지정된 세트에 없으면 `true`로 확인됩니다.	`ColumnValues "colA" not in [ 1, 2, 3 ], ColumnValues "colA" not in [ "a", "b", "c" ]`
matches */ab+c/i*	규칙 유형 응답이 정규 표현식과 일치하는 경우 `true`로 확인됩니다.	`ColumnValues "colA" matches "[a-zA-Z]*"`
*/ab+c/i*와 일치하지 않음	규칙 유형 응답이 정규식과 일치하지 않는 경우 `true`로 확인됩니다.	`ColumnValues "colA" not matches "[a-zA-Z]*"`
`now()`	`ColumnValues` 규칙 유형에서만 작동하여 날짜 표현식을 생성합니다.	`ColumnValues "load_date" > (now() - 3 days)`
matches/in […]/not matches/not in [...] `with threshold`	규칙 조건과 일치하는 값의 백분율을 지정합니다. `ColumnValues` , `ColumnDataType` , `CustomSQL` 규칙 유형에만 적용됩니다.	`ColumnValues "colA" in ["A", "B"] with threshold > 0.8, ColumnValues "colA" matches "[a-zA-Z]*" with threshold between 0.2 and 0.9 ColumnDataType "colA" = "Timestamp" with threshold > 0.9`

NULL, EMPTY, WHITESPACES_ONLY에 대한 키워드

문자열 열에 null, 비어 있음 또는 공백만 있는 문자열이 있는지 확인하려면 다음 키워드를 사용할 수 있습니다.

- NULL/null - 이 키워드는 문자열 열의 `null` 값에 대해 true로 확인됩니다.

  `ColumnValues "colA" != NULL with threshold > 0.5`는 데이터의 50% 이상에 null 값이 없는 경우 true를 반환합니다.

`(ColumnValues "colA" = NULL) or (ColumnLength "colA" > 5)`는 null 값이 있거나 길이가 5를 초과하는 모든 행에 대해 true를 반환합니다. 이 경우 "compositeRuleEvaluation.method" = "ROW" 옵션을 사용해야 한다는 점에 유의하세요.

- EMPTY/empty - 이 키워드는 문자열 열의 빈 문자열("") 값에 대해 true로 확인됩니다. 일부 데이터 형식은 문자열 열의 null을 빈 문자열로 변환합니다. 이 키워드는 데이터에서 빈 문자열을 필터링하는 데 도움이 됩니다.

  `(ColumnValues "colA" = EMPTY) or (ColumnValues "colA" in ["a", "b"])`는 행이 비어 있거나 "a" 또는 "b"인 경우 true를 반환합니다. 이 경우 "compositeRuleEvaluation.method" = "ROW" 옵션을 사용해야 한다는 점에 유의하세요.

- WHITESPACES_ONLY/whitespaces_only - 이 키워드는 문자열 열에 공백(" ") 값만 있는 문자열에 대해 true로 확인됩니다.

  `ColumnValues "colA" not in ["a", "b", WHITESPACES_ONLY]`는 'a', 'b' 또는 공백만 있는 행이 아닌 경우 true를 반환합니다.

  지원되는 규칙:

  - ColumnValues

숫자 또는 날짜 기반 표현식에서 열에 null이 있는지 확인하려는 경우 다음 키워드를 사용할 수 있습니다.

- NULL/null - 이 키워드는 문자열 열의 null 값에 대해 true로 확인됩니다.

  `ColumnValues "colA" in [NULL, "2023-01-01"]`은 열의 날짜가 `2023-01-01`이거나 null인 경우 true를 반환합니다.

  `(ColumnValues "colA" = NULL) or (ColumnValues "colA" between 1 and 9)`는 null 값이 있거나 1에서 9 사이의 값을 갖는 모든 행에 대해 true를 반환합니다. 이 경우 "compositeRuleEvaluation.method" = "ROW" 옵션을 사용해야 한다는 점에 유의하세요.

  지원되는 규칙:

  - ColumnValues

Where 절을 사용한 필터링

> **Note**
>
> 여기에서 조항은 AWS Glue 4.0에서만 지원됩니다.

규칙을 작성할 때 데이터를 필터링할 수 있습니다. 이는 조건부 규칙을 적용하려는 경우에 유용합니다.

```
<DQDL Rule> where "<valid SparkSQL where clause> "
```

필터는 where 키워드와 따옴표(""))로 묶인 유효한 SparkSQL 문을 사용하여 지정해야 합니다.

임계값이 있는 규칙에 where 절을 추가하려면 임계값 조건보다 먼저 where 절을 지정해야 합니다.

```
<DQDL Rule> where "valid SparkSQL statement>" with threshold <threshold condition>
```

이 구문을 사용하면 다음과 같은 규칙을 작성할 수 있습니다.

```
Completeness "colA" > 0.5 where "colB = 10"
ColumnValues "colB" in ["A", "B"] where "colC is not null" with threshold > 0.9
ColumnLength "colC" > 10 where "colD != Concat(colE, colF)"
```

제공된 SparkSQL 문이 유효한지 검증해보겠습니다. 유효하지 않은 경우 규칙 평가가 실패하고 다음 형식의 `IllegalArgumentException`이 실행됩니다.

```
Rule <DQDL Rule> where "<invalid SparkSQL>" has provided an invalid where clause :
<SparkSQL Error>
```

행 수준 오류 레코드 식별 기능이 켜져 있을 때의 Where 절 동작

AWS Glue Data Quality를 사용하면 실패한 특정 레코드를 식별할 수 있습니다. 행 수준 결과를 지원하는 규칙에 where 절을 적용할 때는 where 절로 필터링된 행에 `Passed`라는 레이블을 지정합니다.

필터링된 행에 별도로 `SKIPPED`라는 레이블을 지정하려는 경우 ETL 작업에 대해 다음 `additionalOptions`를 설정할 수 있습니다.

```
object GlueApp {
  val datasource = glueContext.getCatalogSource(
```

```
    database="<db>",
    tableName="<table>",
    transformationContext="datasource"
  ).getDynamicFrame()

  val ruleset = """
    Rules = [
       IsComplete "att2" where "att1 = 'a'"
    ]
  """

  val dq_results = EvaluateDataQuality.processRows(
    frame=datasource,
    ruleset=ruleset,
    additionalOptions=JsonOptions("""
        {
          "rowLevelConfiguration.filteredRowLabel":"SKIPPED"
        }
      """
    )
  )
}
```

다음 규칙 및 데이터 프레임을 예시로 참조하세요.

```
IsComplete att2 where "att1 = 'a'"
```

id	att1	att2	행 수준 결과(기본값)	행 수준 결과(Skipped 옵션)	설명
1	a	f	통과	통과	
2	b	d	통과	SKIPPED	att1이 "a"가 아니므로 행이 필터링됩니다.
3	a	null	FAILED	FAILED	
4	a	f	통과	통과	

id	att1	att2	행 수준 결과 (기본값)	행 수준 결과 (Skipped 옵션)	설명
5	b	null	통과	SKIPPED	att1이 "a"가 아니므로 행이 필터링됩니다.
6	a	f	통과	통과	

동적 규칙

> **Note**
>
> 동적 규칙은 AWS Glue ETL에서만 지원되며 AWS Glue Data Catalog에서 지원되지 않습니다.

이제 동적 규칙을 작성하여 규칙에 의해 생성된 현재 지표를 기록 값과 비교할 수 있습니다. 이러한 기록 비교는 표현식에 `last()` 연산자를 사용하여 사용할 수 있습니다. 예를 들어 현재 실행 중인 행 수가 동일한 데이터 세트에 대한 가장 최근의 이전 행 수보다 많을 때 규칙 `RowCount > last()`가 성공합니다. `last()`는 고려할 기록 지표의 개수를 나타내는 선택적 자연수 인수를 취하고, `last(k)`는 마지막 `k` 지표를 참조하는 `k >= 1`을 취합니다.

- 사용 가능한 데이터 포인트가 없는 경우 `last(k)`는 기본값 0.0을 반환합니다.
- 사용할 수 있는 지표가 `k`개 미만인 경우 `last(k)`는 이전 지표를 모두 반환합니다.

유효한 표현식을 만들려면 `last(k)`를 사용합니다. 여기서 `k > 1`에는 여러 개의 기록 결과를 하나의 숫자로 줄이는 집계 함수가 필요합니다. 예를 들어 `RowCount > avg(last(5))`는 현재 데이터 세트의 행 수가 동일한 데이터 세트의 최근 5개 행 수의 평균보다 확실하게 큰지 확인합니다. `RowCount > last(5)`는 현재 데이터 세트 행 수를 목록과 유의미하게 비교할 수 없기 때문에 오류를 생성합니다.

지원되는 집계 함수:

- `avg`
- `median`
- `max`

- min
- sum
- std(표준 편차)
- abs(절대값)
- index(last(k), i)를 사용하면 최근 k개 중 가장 최근인 i번째 값을 선택할 수 있습니다. i는 0으로 인덱싱되므로 index(last(3), 0)은 가장 최근의 데이터 포인트를 반환하고, index(last(3), 3)은 데이터 포인트가 3개뿐이지만 4번째 가장 최근의 데이터 포인트를 인덱싱하려고 시도하기 때문에 오류가 발생합니다.

샘플 표현식

ColumnCorrelation

- ColumnCorrelation "colA" "colB" < avg(last(10))

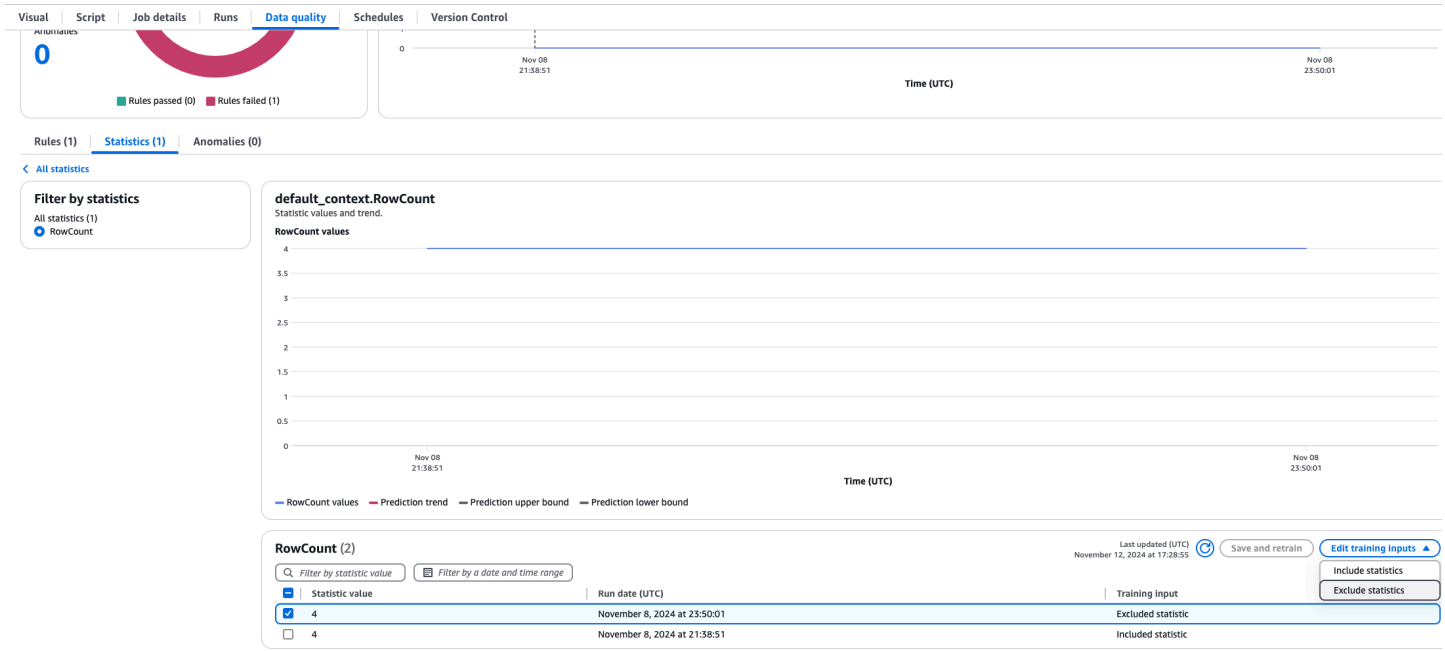
DistinctValuesCount

- DistinctValuesCount "colA" between min(last(10))-1 and max(last(10))+1

숫자 조건 또는 임곗값이 있는 대부분의 규칙 유형은 동적 규칙을 지원합니다. 제공된 테이블인 분석기 및 규칙을 참조하여 해당 규칙 유형에 동적 규칙이 지원되는지 확인하세요.

동적 규칙에서 통계 제외

때때로 동적 규칙 계산에서 데이터 통계를 제외해야 합니다. 과거 데이터 로드를 수행했지만 평균에 영향을 주지 않기를 원한다고 가정합니다. 이를 위해 AWS Glue ETL에서 작업을 열고 데이터 품질 탭을 선택한 다음, 통계를 선택하고 제외하려는 통계를 선택합니다. 통계 테이블과 함께 추세 차트를 볼 수 있습니다. 제외하려는 값을 선택하고 통계 제외를 선택합니다. 이제 제외된 통계는 동적 규칙 계산에 포함되지 않습니다.



분석기

> **Note**
>
> 분석기는 AWS Glue 데이터 카탈로그에서 지원되지 않습니다.

DQDL 규칙은 분석기라는 함수를 사용하여 데이터에 대한 정보를 수집합니다. 이 정보는 규칙의 부울 표현식에 의해 사용되어 규칙의 성공 또는 실패 여부를 결정합니다. 예를 들어 RowCount 규칙 `RowCount > 5` 는 행 수 분석기를 사용하여 데이터 세트의 행 수를 검색하고, 그 수를 표현식 `> 5`와 비교하여 현재 데이터 세트에 5개 이상의 행이 존재하는지 확인합니다.

때로는 규칙을 작성하는 대신 분석기를 생성한 다음 이상을 탐지하는 데 사용할 수 있는 통계를 생성하는 것을 권장합니다. 이러한 경우에는 분석기를 생성할 수 있습니다. 분석기는 다음과 같은 점에서 규칙과 다릅니다.

기능	분석기	규칙
규칙 세트의 일부	예	예
통계 생성	예	예
관찰 결과 생성	예	예

기능	분석기	규칙
조건 평가 및 어설션 가능	아니요	예
실패 시 작업 중지, 작업 계속 처리 등의 작업을 구성할 수 있음	아니요	예

분석기는 규칙 없이 독립적으로 존재할 수 있으므로 신속하게 구성하고 데이터 품질 규칙을 점진적으로 구축할 수 있습니다.

일부 규칙 유형은 규칙 세트의 `Analyzers` 블록에 입력하여 분석기에 필요한 규칙을 실행하고 조건에 대한 검사를 적용하지 않고 정보를 수집할 수 있습니다. 일부 분석기는 규칙과 연결되지 않으며 `Analyzers` 블록에만 입력할 수 있습니다. 다음 테이블에는 각 항목이 규칙 또는 독립 실행형 분석기로 지원되는지 여부와 각 규칙 유형에 대한 추가 세부 정보가 나와 있습니다.

분석기를 사용한 규칙 세트 예제

다음 규칙 세트는 다음을 사용합니다.

- 데이터 세트가 지난 세 번의 작업 실행에 대한 후행 평균보다 증가하고 있는지 확인하는 동적 규칙
- 데이터 세트의 `Name` 열에 고유한 값의 수를 기록하는 `DistinctValuesCount` 분석기
- 시간 경과에 따른 최소 및 최대 `Name` 크기를 추적하는 `ColumnLength` 분석기

분석기 메트릭 결과는 작업 실행의 Data Quality 탭에서 확인할 수 있습니다.

```
Rules = [
   RowCount > avg(last(3))
]
Analyzers = [
   DistinctValuesCount "Name",
   ColumnLength "Name"
]
```

AWS Glue Data Quality는 다음 분석기를 지원합니다.

분석기 이름	기능
RowCount	데이터 세트의 행 수를 계산합니다.
Completeness	열의 완전성 백분율을 계산합니다.
Uniqueness	열의 고유성 비율을 계산합니다.
Mean	숫자 열의 평균을 계산합니다.
Sum	숫자 열의 합계를 계산합니다.
StandardDeviation	숫자 열의 표준 편차를 계산합니다.
Entropy	숫자 열의 엔트로피를 계산합니다.
DistinctValuesCount	열의 고유 값 수를 계산합니다.
UniqueValueRatio	열의 고유 값 비율을 계산합니다.
ColumnCount	데이터 세트의 열 수를 계산합니다.
ColumnLength	열의 길이를 계산합니다.
ColumnValues	숫자 열의 최소값, 최대값을 계산합니다. 숫자가 아닌 열의 최소 열 길이 및 최대 열 길이를 계산합니다.
ColumnCorrelation	지정된 열의 열 상관 관계를 계산합니다.
CustomSql	CustomSQL에서 반환된 통계를 계산합니다.
AllStatistics	다음 통계를 계산합니다. • RowCount, ColumnCount • 모든 열: 완전성, 고유성 • 숫자 열: 최소, 최대, 엔트로피, 평균, 표준 편차, 합계 • 문자열: 최소 길이, 최대 길이

설명

'#' 문자를 사용하여 DQDL 문서에 설명을 추가할 수 있습니다. '#' 문자 이후부터 줄 끝까지의 모든 내용은 DQDL에서 무시됩니다.

```
Rules = [
    # More items should generally mean a higher price, so correlation should be
 positive
    ColumnCorrelation "price" "num_items" > 0
]
```

DQDL 규칙 유형 참조

이 섹션에서는 AWS Glue 데이터 품질에서 지원하는 각 규칙 유형에 대한 참조를 제공합니다.

> **Note**
>
> - DQDL은 현재 중첩형 또는 목록형 열 데이터를 지원하지 않습니다.
> - 아래 테이블의 괄호로 묶인 값은 규칙 인수에 제공된 정보로 대체됩니다.
> - 규칙에는 일반적으로 표현식에 대한 추가 인수가 필요합니다.

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
AggregateMatch	총 판매액과 같은 요약 지표를 비교하여 두 데이터 세트	하나 이상의 집계	첫 번째 및 두 번째 집계 열 이름이 일치하는 경우:	예	아니요	아니요	아니요	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?

	가 일치하는지 확인합니다. 금융 기관에서 모든 데이터가 소스 시스템에서 수집되었는지 비교할 때 유용합니다.		`Column.[Column].gregatetch`
			
			첫 번째 및 두 번째 집계 열 이름이 다른 경우:
			
			`Column.[Column1,lumn2].gregatetch`						

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?

AllStatistics	데이터 세트의 제공된 열에 대한 여러 지표를 수집하는 독립 실행형 분석기	단일 열 이름	모든 유형의 열에 적용됩니다.
			`Dataset.RowCou`
			`Column.[Column].mpleters`
			`Column.[Column].iquenes`
			문자열 값 열에 대한 추가 지표:
			`ColumnLgth metrics`
				아니요	예	아니요	아니요	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
			숫자 값 열에 대한 추가 지표: Column\ues metrics						
ColumnCorrelation	두 열의 상관 관계 수준을 확인합니다.	정확히 두 개의 열 이름	Multicomn.[Column1,Colu2].ColuCorrelaon	예	예	아니요	예	아니요	예
ColumnCnt	삭제된 열이 있는지 확인합니다.	None	Dataset.Columnunt	예	예	아니요	예	예	아니요
ColumnDaType	열이 데이터 형식을 준수하는지 확인합니다.	정확히 하나의 열 이름	Column.[Column].lumnDatype.Coniance	예	아니요	아니요	예, 행 수준 임곗값 표현식에서	아니요	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
ColumnExists	데이터 세트에 열이 있는지 확인합니다. 이를 통해 고객은 특정 열을 사용할 수 있도록 셀프 서비스 데이터 플랫폼을 구축할 수 있습니다.	정확히 하나의 열 이름	N/A	예	아니요	아니요	아니요	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?

ColumnLength	데이터 길이가 일관되는지 확인합니다.	정확히 하나의 열 이름	`Column.[Column].MaximumLength`
			
			`Column.[Column].MinimumLength`
			
			행 수준 임곗값이 제공된 경우 추가 지표:
			
			`Column.[Column].ColumnValues.Compliance`
				예	예	예, 행 수준 임곗값이 제공된 경우	아니요	예 최소 및 최대 길이를 분석하여 관찰 결과만 생성합니다.	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
ColumnN esMatchI ttern	열 이름이 정의된 패턴과 일치하는지 확인합니다. 거버넌스 팀이 열 이름의 일관성을 적용하는 데 유용합니다.	열 이름에 대한 정규식	Dataset .Columr mesPatt nMatchR io	예	아니요	아니요	아니요	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?

ColumnValues	데이터가 정의된 값에 따라 일관되는지 확인합니다. 이 규칙은 정규식을 지원합니다.	정확히 하나의 열 이름	`Column.[Column].Maximum`
			
			`Column.[Column].Minimum`
			
			행 수준 임곗값이 제공된 경우 추가 지표:
			
			`Column.[Column].ColumnValues.Compliance`
				예	예	예, 행 수준 임곗값이 제공된 경우	아니요	예 최솟값 및 최댓값을 분석하여 관찰 결과만 생성합니다.	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
완전성	데이터에 공백 또는 NULL이 있는지 확인합니다.	정확히 하나의 열 이름	`Column.[Column].mpleters`	예	예	예	예	예	예

CustomS	고객은 거의 모든 유형의 데이터 품질 검사를 SQL로 구현할 수 있습니다.	SQL 문
		
		(선택 사항) 행 수준 임곗값
			`Dataset.CustomL`
			
			행 수준 임곗값이 제공된 경우 추가 지표:
			
			`Dataset.CustomL.Compl nce`
				예	아니요	예, 행 수준 임곗값이 제공된 경우	예	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
DataFres ness	데이터가 최신인지 확인합니다.	정확히 하나의 열 이름	Column. [C olumn]. taFresh ss.Comp ance	예	아니요	예	아니요	아니요	예
DatasetM tch	두 데이터 세트를 비교하여 두 데이터 세트가 동기화되었는지 확인합니다.	참조 데이터 세트 이름 열 매핑 (선택 사항) 일치하는 항목이 있는지 확인하는 열	Dataset [Referer Dataset ias].Da setMatc	예	아니요	예	예	아니요	아니요
DistinctV aluesCou t	중복된 값이 있는지 확인합니다.	정확히 하나의 열 이름	Column. [C olumn]. stinctV uesCour	예	예	예	예	예	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
DetectAnomalies	다른 규칙 유형의 보고된 지표에서 이상이 있는지 확인합니다.	규칙 유형	규칙 유형 인수로 보고된 지표	예	아니요	아니요	아니요	아니요	아니요
Entropy	데이터의 엔트로피를 확인합니다.	정확히 하나의 열 이름	`Column.[Column].tropy`	예	예	아니요	예	아니요	예
IsComplete	데이터가 100% 완전한지 확인합니다.	정확히 하나의 열 이름	`Column.[Column].mpleteness`	예	아니요	예	아니요	아니요	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?

IsPrimaryKey	열이 프라이머리 키(NULL이 아니고 고유함)인지 확인합니다.	정확히 하나의 열 이름	단일 열의 경우:
			`Column.[Column].iquenes`
			여러 열의 경우:
			`Multicolumn.[Comma Delimit Columns Uniquers`
				예	아니요	예	아니요	아니요	예

IsUnique	데이터가 100% 고유한지 확인합니다.	정확히 하나의 열 이름	`Column.[Column].iquenes`	예	아니요	예	아니요	아니요	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
평균	평균이 설정된 임곗값과 일치하는지 확인합니다.	정확히 하나의 열 이름	`Column.[Column].an`	예	예	예	예	아니요	예
ReferentialIntegrity	두 데이터 세트에 참조 무결성이 있는지 확인합니다.	데이터 세트에서 하나 이상의 열 이름 참조 데이터 세트에서 가져온 하나 이상의 열 이름	`Column.[Reference[DatasetAlias].ReferentialIntegrity`	예	아니요	예	예	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
RowCou	레코드 수가 임곗값과 일치하는지 확인합니다.	None	Dataset .RowCou	예	예	아니요	예	예	예
RowCou atch	두 데이터 세트 사이에서 레코드 수가 일치하는지 확인합니다.	참조 데이터 세트 별칭	Dataset [Referer Dataset ias].Rc ountMat	예	아니요	아니요	예	아니요	아니요
Standarc eviation	표준 편차가 임곗값과 일치하는지 확인합니다.	정확히 하나의 열 이름	Column [C olumn]. andardD iation	예	예	예	예	아니요	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
SchemaMatch	두 데이터 세트 사이에서 스키마가 일치하는지 확인합니다.	참조 데이터 세트 별칭	Dataset[Referer Dataset ias].SchemaMatch	예	아니요	아니요	예	아니요	아니요
Sum	합계가 설정된 임곗값과 일치하는지 확인합니다.	정확히 하나의 열 이름	Column.[Column].m	예	예	아니요	예	아니요	예
Uniqueness	데이터 세트의 고유성이 임곗값과 일치하는지 확인합니다.	정확히 하나의 열 이름	Column.[Column].iquenes	예	예	예	예	아니요	예

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
UniqueV ueRatio	고유 값 비율이 임곗값과 일치하는지 확인합니다.	정확히 하나의 열 이름	Column [C olumn]. iqueVa Ratio	예	예	예	예	아니요	예
FileFresh ness	Amazon S3의 파일이 새 파일인지 확인합니다.	파일 또는 폴더 경로와 임계치.	Dataset .FileFr hness.C pliance Dataset .FileCo t	예	아니요	아니요	아니요	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?
FileMatch	파일 내용이 체크섬 또는 다른 파일과 일치하는지 확인합니다. 이 규칙은 체크섬을 사용하여 두 파일이 동일한지 검증합니다.	소스 파일 또는 폴더 경로와 대상 파일 또는 폴더 경로.	통계가 생성되지 않습니다.	예	아니요	아니요	아니요	아니요	아니요

Ruletype	설명	인수	보고된 지표	규칙으로 지원되나요?	분석기로 지원되나요?	행 수준 결과를 반환하나요?	동적 규칙이 지원되나요?	관찰 결과 생성	Where 절 구문을 지원합니까?

FileSize	파일 크기가 지정된 조건과 일치하는지 확인합니다.	파일 또는 폴더 경로와 임계치.	`Dataset.FileSi.Compli ce`
			`Dataset.FileCo t`
			`Dataset.Maximu ileSize`
			`Dataset.Minimu ileSize`
				예	아니요	아니요	아니요	아니요	아니요

FileUniqu eness	체크섬을 사용하여 파일이 고유한지 확인합니다.	파일 또는 폴더 경로와 임계치.	`Dataset.FileUn uenessR io` `Dataset.FileCo t`	예	아니요	아니요	아니요	아니요	아니요

주제

- AggregateMatch
- ColumnCorrelation
- ColumnCount

- ColumnDataType
- ColumnExists
- ColumnLength
- ColumnNamesMatchPattern
- ColumnValues
- 완전성
- CustomSQL
- DataFreshness
- DatasetMatch
- DistinctValuesCount
- Entropy
- IsComplete
- IsPrimaryKey
- IsUnique
- Mean
- ReferentialIntegrity
- RowCount
- RowCountMatch
- StandardDeviation
- Sum
- SchemaMatch
- Uniqueness
- UniqueValueRatio
- DetectAnomalies
- FileFreshness
- FileMatch
- FileUniqueness
- FileSize

AggregateMatch

지정된 표현식에서 두 열의 집계 비율을 확인합니다. 이 규칙 유형은 여러 데이터 세트에 적용됩니다. 두 열의 집계를 평가하고 첫 번째 열 집계 결과를 두 번째 열 집계 결과로 나누어 비율을 생성합니다. 제공된 표현식에서 비율을 확인하여 부울 응답을 생성합니다.

구문

열 집계

```
AggregateMatch <AGG_OPERATION> (<OPTIONAL_REFERENCE_ALIAS>.<COL_NAME>)
```

- AGG_OPERATION – 집계에 사용할 연산입니다. 현재 `sum` 및 `avg`가 지원됩니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- OPTIONAL_REFERENCE_ALIAS – 기본 데이터 세트가 아닌 참조 데이터 세트에서 열을 가져온 경우 이 파라미터를 제공해야 합니다. AWS Glue 데이터 카탈로그에서 이 규칙을 사용하는 경우 참조 별칭은 '<database_name>.<table_name>.<column_name>' 형식을 따라야 합니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- COL_NAME - 집계할 열의 이름입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short

예: 평균

```
"avg(rating)"
```

예: 합계

```
"sum(amount)"
```

예: 참조 데이터 세트의 열 평균

```
"avg(reference.rating)"
```

규칙

```
AggregateMatch <AGG_EXP_1> <AGG_EXP_2> <EXPRESSION>
```

- AGG_EXP_1 - 첫 번째 열 집계입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- AGG_EXP_2 - 두 번째 열 집계입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 합계를 사용한 일치 항목 집계

다음 예제 규칙은 amount 열에서 값의 합계가 total_amount 열에서 값의 합계와 정확히 같은지 여부를 확인합니다.

```
AggregateMatch "sum(amount)" "sum(total_amount)" = 1.0
```

예: 평균을 사용한 일치 항목 집계

다음 예제 규칙은 ratings 열에서 값의 평균이 reference 데이터 세트 내 ratings 열에서 값의 평균 중 90% 이상인지를 확인합니다. 참조 데이터 세트는 ETL 또는 데이터 카탈로그 환경에서 추가 데이터 소스로 제공됩니다.

AWS Glue ETL에서 다음을 사용할 수 있습니다.

```
AggregateMatch "avg(ratings)" "avg(reference.ratings)" >= 0.9
```

AWS Glue 데이터 카탈로그에서 다음을 사용할 수 있습니다.

```
AggregateMatch "avg(ratings)" "avg(database_name.tablename.ratings)" >= 0.9
```

Null 동작

AggregateMatch 규칙은 집계 방법(합계/평균)의 계산에서 NULL 값이 있는 행을 무시합니다. 예:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

units열의 평균은 (0 + 20 + 40)/3 = 20이 됩니다. 101행과 103행은 이 계산에서 고려되지 않습니다.

ColumnCorrelation

주어진 표현식을 기준으로 두 열 간의 상관 관계를 확인합니다. AWS Glue 데이터 품질에서는 피어슨 상관 계수를 사용하여 두 열 간의 선형 상관 관계를 측정합니다. 결과는 관계의 강도와 방향을 측정하는 -1과 1 사이의 숫자입니다.

구문

```
ColumnCorrelation <COL_1_NAME> <COL_2_NAME> <EXPRESSION>
```

- COL_1_NAME - 데이터 품질 규칙을 평가할 첫 번째 열의 이름입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- COL_2_NAME - 데이터 품질 규칙을 평가할 첫 번째 열의 이름입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 열 상관 관계

다음 예제 규칙은 열 height 및 weight 간의 상관 계수가 강한 양의 상관 관계(0.8보다 큰 계수 값)를 갖는지 여부를 확인합니다.

```
ColumnCorrelation "height" "weight" > 0.8
```

```
ColumnCorrelation "weightinkgs" "Salary" > 0.8 where "weightinkgs > 40"
```

샘플 동적 규칙

- `ColumnCorrelation "colA" "colB" between min(last(10)) and max(last(10))`
- `ColumnCorrelation "colA" "colB" < avg(last(5)) + std(last(5))`

Null 동작

`ColumnCorrelation` 규칙은 상관 관계 계산에서 `NULL` 값이 있는 행을 무시합니다. 예:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

101행과 103행은 무시되고 `ColumnCorrelation`이 1.0이 됩니다.

ColumnCount

지정된 표현식을 기준으로 기본 데이터 세트의 열 수를 확인합니다. 표현식에서 > 및 < 같은 연산자를 사용하여 열 수 또는 열 범위를 지정할 수 있습니다.

구문

```
ColumnCount <EXPRESSION>
```

- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 열 수 숫자 검사

다음 예제 규칙은 열 수가 지정된 범위 내에 있는지 확인합니다.

```
ColumnCount between 10 and 20
```

샘플 동적 규칙

- `ColumnCount >= avg(last(10))`
- `ColumnCount between min(last(10))-1 and max(last(10))+1`

ColumnDataType

지정된 열의 값을 Apache Spark에서 제공된 유형으로 캐스팅할 수 있는지 여부를 확인합니다. `with threshold` 식을 수락하여 열에 있는 값의 하위 세트를 확인합니다.

구문

```
ColumnDataType <COL_NAME> = <EXPECTED_TYPE>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 문자열 유형

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPECTED_TYPE - 열에 있는 값의 예상 유형입니다.

  지원되는 값: Boolean, Date, Timestamp, Integer, Double, Float, Long

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 예상 유형이어야 하는 값의 백분율을 지정하는 선택적 표현식입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short

예: 열 데이터 형식 정수 역할의 문자열

다음 예제 규칙은 문자열 유형인 지정된 열의 값을 정수로 캐스팅할 수 있는지 여부를 확인합니다.

```
ColumnDataType "colA" = "INTEGER"
```

예: 열 데이터 형식 정수 역할의 문자열에서 값의 하위 세트 확인

다음 예제 규칙은 문자열 유형인 지정된 열의 값 중 90%를 정수로 캐스팅할 수 있는지 여부를 확인합니다.

```
ColumnDataType "colA" = "INTEGER" with threshold > 0.9
```

ColumnExists

열이 존재하는지 확인합니다.

구문

```
ColumnExists <COL_NAME>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형

예: 열이 존재함

다음 예제 규칙은 `Middle_Name` 열이 존재하는지 여부를 확인합니다.

```
ColumnExists "Middle_Name"
```

ColumnLength

열에 있는 각 행의 길이가 지정된 표현식을 준수하는지 확인합니다.

구문

```
ColumnLength <COL_NAME><EXPRESSION>
```

- COL_NAME – 데이터 품질 규칙을 평가하려는 열의 이름입니다.

  지원되는 열 유형: 문자열
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 열 행 길이

다음 예제 규칙은 `Postal_Code` 열의 각 행에 있는 값의 길이가 5자인지 여부를 확인합니다.

```
ColumnLength "Postal_Code" = 5
```

```
ColumnLength "weightinkgs" = 2 where "weightinkgs > 10"
```

Null 동작

ColumnLength 규칙은 NULL을 0 길이 문자열로 취급합니다. NULL 행의 경우:

```
ColumnLength "Postal_Code" > 4 # this will fail
```

```
ColumnLength "Postal_Code" < 6 # this will succeed
```

다음 예시 복합 규칙은 NULL 값을 명시적으로 실패하도록 하는 방법을 제공합니다.

```
(ColumnLength "Postal_Code" > 4) AND (ColumnValues "Postal_Code" != NULL)
```

ColumnNamesMatchPattern

기본 데이터 세트의 모든 열 이름이 지정된 정규식과 일치하는지 확인합니다.

구문

```
ColumnNamesMatchPattern <PATTERN>
```

- PATTERN – 데이터 품질 규칙을 평가하는 데 사용할 패턴입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short

예: 열 이름 일치 패턴

다음 예제 규칙은 모든 열이 접두사 'aws_'로 시작하는지 여부를 확인합니다.

```
ColumnNamesMatchPattern "aws_.*"
ColumnNamesMatchPattern "aws_.*" where "weightinkgs > 10"
```

ColumnValues

열의 값에 대해 표현식을 실행합니다.

구문

```
ColumnValues <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 허용된 값

다음 예제 규칙은 지정된 열의 각 값이 허용되는 값 세트(null, 비어 있음, 공백만 있는 문자열 포함)에 있는지 확인합니다.

```
ColumnValues "Country" in [ "US", "CA", "UK", NULL, EMPTY, WHITESPACES_ONLY ]
ColumnValues "gender" in ["F", "M"] where "weightinkgs < 10"
```

예: 정규 표현식

다음 예제 규칙은 열의 값을 정규 표현식과 비교하여 검사합니다.

```
ColumnValues "First_Name" matches "[a-zA-Z]*"
```

예: 날짜 값

다음 예제 규칙은 날짜 열의 값을 날짜 표현식과 비교하여 검사합니다.

```
ColumnValues "Load_Date" > (now() - 3 days)
```

예: 숫자 값

다음 예제 규칙은 열 값이 특정 숫자 제약 조건과 일치하는지 여부를 확인합니다.

```
ColumnValues "Customer_ID" between 1 and 2000
```

Null 동작

모든 `ColumnValues` 규칙(`!=` 및 `NOT IN` 제외)의 경우 `NULL` 행이 규칙에 실패합니다. null 값으로 인해 규칙이 실패하면 실패 이유가 다음과 같이 표시됩니다.

```
Value: NULL does not meet the constraint requirement!
```

다음 예제 복합 규칙은 NULL 값을 명시적으로 허용하는 방법을 제공합니다.

```
(ColumnValues "Age" > 21) OR (ColumnValues "Age" = NULL)
```

!= 및 not in 구문을 사용하는 부정 ColumnValues 규칙은 NULL 행에 대해 성공합니다. 예:

```
ColumnValues "Age" != 21
```

```
ColumnValues "Age" not in [21, 22, 23]
```

다음 예제는 NULL 값을 명시적으로 실패하도록 하는 방법을 제공합니다.

```
(ColumnValues "Age" != 21) AND (ColumnValues "Age" != NULL)
```

```
ColumnValues "Age" not in [21, 22, 23, NULL]
```

완전성

지정된 표현식을 기준으로 열에서 전체(null이 아닌) 값의 백분율을 검사합니다.

구문

```
Completeness <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: Null 값 백분율

다음 예제 규칙은 열에 있는 값의 95% 이상이 완전한지를 확인합니다.

```
Completeness "First_Name" > 0.95
```

```
Completeness "First_Name" > 0.95 where "weightinkgs > 10"
```

샘플 동적 규칙

- `Completeness "colA" between min(last(5)) - 1 and max(last(5)) + 1`
- `Completeness "colA" <= avg(last(10))`

Null 동작

CSV 데이터 형식에 대한 참고 사항: CSV 열의 빈 행은 여러 동작을 표시할 수 있습니다.

- 열의 유형이 `String`인 경우 빈 행은 빈 문자열로 인식되어 `Completeness` 규칙에 실패하지 않습니다.
- 열이 `Int`와 같은 다른 데이터 유형인 경우 빈 행이 `NULL`로 인식되어 `Completeness` 규칙에 실패합니다.

CustomSQL

이 규칙 유형은 다음과 같은 두 가지 사용 사례를 지원하도록 확장되었습니다.

- 데이터 세트에 대해 사용자 지정 SQL 문을 실행하고 지정된 표현식을 기준으로 반환 값을 확인합니다.
- 일부 조건과 비교하여 행 수준 결과를 얻는 SELECT 문에 열 이름을 지정하는 사용자 지정 SQL 문을 실행합니다.

구문

```
CustomSql <SQL_STATEMENT> <EXPRESSION>
```

- SQL_STATEMENT - 큰따옴표로 묶인 단일 숫자 값을 반환하는 SQL 문입니다.
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 전체 규칙 결과를 검색하기 위한 사용자 지정 SQL

이 예제 규칙에서는 SQL 문을 사용하여 데이터 세트의 레코드 수를 검색합니다. 그런 다음 레코드 수가 10~20 사이인지 확인합니다.

```
CustomSql "select count(*) from primary" between 10 and 20
```

예: 행 수준 결과를 검색하기 위한 사용자 지정 SQL

이 예제 규칙은 일부 조건과 비교하여 행 수준 결과를 얻는 SELECT 문에 열 이름을 지정하는 SQL 문을 사용합니다. 임곗값 조건 표현식에서는 전체 규칙이 실패하기 위해 실패해야 하는 레코드 수의 임곗값을 정의합니다. 규칙에는 조건과 키워드가 모두 포함되지 않을 수도 있습니다.

```
CustomSql "select Name from primary where Age  > 18"
```

or

```
CustomSql "select Name from primary where Age > 18" with threshold  > 3
```

> **Important**
>
> `primary` 별칭은 평가하려는 데이터 세트의 이름을 나타냅니다. 콘솔에서 시각적 ETL 작업을 수행할 때 `primary`에서는 항상 `EvaluateDataQuality.apply()` 변환에 전달 중인 `DynamicFrame`을 나타냅니다. AWS Glue 데이터 카탈로그를 사용하여 테이블에 대해 데이터 품질 작업을 실행하는 경우 `primary`는 테이블을 나타냅니다.

AWS Glue 데이터 카탈로그에 있는 경우 실제 테이블 이름을 사용할 수도 있습니다.

```
CustomSql "select count(*) from database.table" between 10 and 20
```

여러 테이블을 조인하여 서로 다른 데이터 요소를 비교할 수도 있습니다.

```
CustomSql "select count(*) from database.table inner join database.table2 on id1 = id2"
 between 10 and 20
```

AWS Glue ETL에서 CustomSQL은 데이터 품질 검사에 실패한 레코드를 식별할 수 있습니다. 이 작업을 수행하려면 데이터 품질을 평가하려는 기본 테이블의 일부인 레코드를 반환해야 합니다. 쿼리의 일부로 반환된 레코드는 성공으로 간주되고 반환되지 않은 레코드는 실패로 간주됩니다. 이는 CustomSQL 쿼리의 결과를 원래 데이터세트와 조인하는 방식으로 작동합니다. SQL 쿼리의 복잡성에 따라 성능에 영향을 미칠 수 있습니다.

방법:

- 기본 테이블에서 열을 하나 이상 선택해야 합니다.
  - `select count(*) from primary`는 OVERALL CustomSQL DQ 규칙에 대한 유효한 쿼리이지만 행 수준 사용자 지정 SQL에는 유효하지 않습니다.
  - 이 규칙은 평가 중에 오류를 발생시킵니다. `The output from CustomSQL must contain at least one column that matches the input dataset for AWS Glue Data Quality to provide row level results. The SQL query is a valid query but the columns from the SQL result are not present in the Input Dataset. Ensure that matching columns are returned from the SQL.`
- SQL 쿼리에서는 테이블에서 '프라이머리 키'를 선택하거나 복합 키를 구성하는 열 세트를 선택합니다. 이렇게 하지 않으면 중복된 행이 일치되어 결과가 일관되지 않고 성능이 저하될 수 있습니다.
- 기본 테이블에서만 키를 선택하고 참조 테이블에서는 선택하지 않습니다.

다음 규칙에서는 age가 100 미만인 레코드는 성공으로, 그 이상인 레코드는 실패로 표시됩니다.

```
CustomSql "select id from primary where age < 100"
```

레코드 중 50%에서 age가 10을 초과하는 경우 이 CustomSQL 규칙은 통과되며 실패한 레코드도 식별합니다. 이 CustomSQL에서 반환된 레코드는 통과로 간주되고 반환되지 않은 레코드는 실패로 간주됩니다.

```
CustomSQL "select ID, CustomerID from primary where age > 10" with threshold > 0.5
```

참고: 데이터 세트에서 사용할 수 없는 레코드를 반환하면 CustomSQL 규칙이 실패합니다.

DataFreshness

현재 시간과 날짜 열 값 간의 차이를 평가하여 열에 있는 데이터의 최신성을 검사합니다. 이 규칙 유형에 시간 기반 표현식을 지정하여 열 값이 최신 상태인지 확인할 수 있습니다.

구문

```
DataFreshness <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 날짜

- 표현식 - 시간 또는 일 단위의 숫자 표현식입니다. 표현식에 시간 단위를 지정해야 합니다.

예: 데이터 최신성

다음 예제 규칙은 데이터 최신성을 확인합니다.

```
DataFreshness "Order_Date" <= 24 hours
DataFreshness "Order_Date" between 2 days and 5 days
```

Null 동작

DataFreshness 규칙은 NULL 값이 있는 행에 대해 실패합니다. null 값으로 인해 규칙이 실패하면 실패 이유가 다음과 같이 표시됩니다.

```
80.00 % of rows passed the threshold
```

여기서 실패한 행의 20%에는 NULL이 있는 행이 포함됩니다.

다음 예제 복합 규칙은 NULL 값을 명시적으로 허용하는 방법을 제공합니다.

```
(DataFreshness "Order_Date" <= 24 hours) OR (ColumnValues "Order_Date" = NULL)
```

Amazon S3 객체의 데이터 최신성

Amazon S3 파일 생성 시간을 기준으로 데이터의 최신성을 검증해야 하는 경우가 있습니다. 이렇게 하려면 다음 코드를 사용하여 타임스탬프를 가져와 데이터 프레임에 추가한 다음 데이터 최신성 검사를 적용할 수 있습니다.

```
df = glueContext.create_data_frame.from_catalog(database = "default", table_name =
 "mytable")
df = df.withColumn("file_ts", df["_metadata.file_modification_time"])

Rules = [
 DataFreshness "file_ts" < 24 hours
]
```

DatasetMatch

기본 데이터 세트의 데이터가 참조 데이터 세트의 데이터와 일치하는지 확인합니다. 제공된 키 열 매핑을 사용하여 두 데이터 세트를 조인합니다. 해당 열에서만 데이터의 관계(equality)를 확인하려는 경

우 추가 열 매핑을 제공할 수 있습니다. DataSetMatch를 사용하려면 조인 키가 고유해야 하고 NULL이 아니어야 합니다(프라이머리 키여야 함). 이러한 조건을 충족하지 않으면 'Provided key map not suitable for given data frames' 오류 메시지가 표시됩니다. 고유한 조인 키를 사용할 수 없는 경우에는 AggregateMatch와 같은 다른 규칙 유형을 사용하여 요약 데이터와 일치시키는 것이 좋습니다.

구문

```
DatasetMatch <REFERENCE_DATASET_ALIAS> <JOIN CONDITION WITH
 MAPPING> <OPTIONAL_MATCH_COLUMN_MAPPINGS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS – 기본 데이터 세트의 데이터를 비교하는 참조 데이터 세트의 별칭입니다.
- KEY_COLUMN_MAPPINGS – 데이터 세트의 키를 구성하는 쉼표로 구분된 열 이름 목록입니다. 두 데이터 세트의 열 이름이 동일하지 않은 경우 ->로 구분해야 합니다.
- OPTIONAL_MATCH_COLUMN_MAPPINGS - 특정 열의 데이터만 일치하는지 확인하려는 경우 이 파라미터를 제공할 수 있습니다. 키 열 매핑과 동일한 구문을 사용합니다. 이 파라미터를 제공하지 않으면 나머지 모든 열의 데이터를 일치시킵니다. 키가 아닌 나머지 열은 두 데이터 세트에서 이름이 같아야 합니다.
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: ID 열을 사용하여 설정된 데이터 세트 일치

다음 예제 규칙은 두 데이터 세트를 조인하기 위해 'ID' 열을 사용하여 기본 데이터 세트 중 90%가 넘는 항목이 참조 데이터 세트와 일치하는지 확인합니다. 이 경우 모든 열을 비교합니다.

```
DatasetMatch "reference" "ID" >= 0.9
```

예: 여러 키 열을 사용하는 설정된 데이터 세트 일치

다음 예제에서는 기본 데이터 세트와 참조 데이터 세트의 키 열 이름이 다릅니다. `ID_1` 및 `ID_2`는 기본 데이터 세트에서 복합 키를 함께 형성합니다. `ID_ref1` 및 `ID_ref2`는 참조 데이터 세트에 복합 키를 함께 형성합니다. 이 시나리오에서는 특수 구문을 사용하여 열 이름을 제공할 수 있습니다.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_ref2->ID_ref2" >= 0.9
```

예: 여러 키 열을 사용하여 설정된 데이터 세트 일치 및 특정 열이 일치하는지 확인

이 예는 이전 예를 기반으로 구축되었습니다. 금액이 포함된 열만 일치하는지 확인하려고 합니다. 이 열의 이름은 기본 데이터 세트에서 Amount1이고, 참조 데이터 세트에서 Amount2입니다. 정확한 일치시키려고 합니다.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_2->ID_ref2" "Amount1->Amount2" >= 0.9
```

DistinctValuesCount

지정된 표현식을 기준으로 열의 고유 값 수를 확인합니다.

구문

```
DistinctValuesCount <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 고유 열 값 개수

다음 예제 규칙은 State 열에 3개 이상의 고유 값이 포함되어 있는지 확인합니다.

```
DistinctValuesCount "State" > 3
DistinctValuesCount "Customer_ID" < 6  where "Customer_ID < 10"
```

샘플 동적 규칙

- `DistinctValuesCount "colA" between avg(last(10))-1 and avg(last(10))+1`
- `DistinctValuesCount "colA" <= index(last(10),2) + std(last(5))`

Entropy

열의 엔트로피 값이 지정된 표현식과 일치하는지 확인합니다. 엔트로피는 메시지에 포함된 정보의 수준을 측정합니다. 열의 값에 대한 확률 분포를 고려할 때 엔트로피는 값을 식별하는 데 필요한 비트 수를 설명합니다.

구문

```
Entropy <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 열 엔트로피

다음 예제 규칙은 Feedback 열의 엔트로피 값이 1보다 큰지 확인합니다.

```
Entropy "Star_Rating" > 1
Entropy "First_Name" > 1  where "Customer_ID < 10"
```

샘플 동적 규칙

- `Entropy "colA" < max(last(10))`
- `Entropy "colA" between min(last(10)) and max(last(10))`

IsComplete

열의 모든 값이 완전한지(null이 아님) 확인합니다.

구문

```
IsComplete <COL_NAME>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형

예: Null 값

다음 예에서는 `email` 열의 모든 값이 null이 아닌지 확인합니다.

```
IsComplete "email"
IsComplete "Email" where "Customer_ID between 1 and 50"
IsComplete "Customer_ID"  where "Customer_ID < 16 and Customer_ID != 12"
IsComplete "passenger_count" where "payment_type<>0"
```

Null 동작

CSV 데이터 형식에 대한 참고 사항: CSV 열의 빈 행은 여러 동작을 표시할 수 있습니다.

- 열의 유형이 `String`인 경우 빈 행은 빈 문자열로 인식되어 `Completeness` 규칙에 실패하지 않습니다.
- 열이 `Int`와 같은 다른 데이터 유형인 경우 빈 행이 `NULL`로 인식되어 `Completeness` 규칙에 실패합니다.

IsPrimaryKey

열에 기본 키가 포함되어 있는지 확인합니다. 열의 모든 값이 고유하고 완전한 경우(null이 아닌 경우) 열에 기본 키가 포함됩니다. 여러 열이 있는 프라이머리 키를 확인할 수도 있습니다.

구문

```
IsPrimaryKey <COL_NAME>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형

예: 기본 키

다음 예제 규칙은 `Customer_ID` 열에 기본 키가 있는지 여부를 확인합니다.

```
IsPrimaryKey "Customer_ID"
IsPrimaryKey "Customer_ID" where "Customer_ID < 10"
```

예: 여러 열이 있는 프라이머리 키입니다. 다음 예제는 유효합니다.

```
IsPrimaryKey "colA" "colB"
```

```
IsPrimaryKey "colA" "colB" "colC"
IsPrimaryKey colA "colB" "colC"
```

IsUnique

열의 모든 값이 고유한지 확인하고 부울 값을 반환합니다.

구문

```
IsUnique <COL_NAME>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형

예시

다음 예제 규칙은 `email` 열의 모든 값이 고유한지 확인합니다.

```
IsUnique "email"
IsUnique "Customer_ID" where "Customer_ID < 10"]
```

다음 예제 규칙은 여러 열을 확인합니다.

```
IsUnique "vendorid" "tpep_pickup_datetime"
```

Mean

열에 있는 모든 값의 평균이 지정된 표현식과 일치하는지 확인합니다.

구문

```
Mean <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short

- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 평균값

다음 예제 규칙은 열에 있는 모든 값의 평균이 임곗값을 초과하는지 여부를 확인합니다.

```
Mean "Star_Rating" > 3
Mean "Salary" < 6200 where "Customer_ID < 10"
```

샘플 동적 규칙

- `Mean "colA" > avg(last(10)) + std(last(2))`
- `Mean "colA" between min(last(5)) - 1 and max(last(5)) + 1`

Null 동작

`Mean` 규칙은 평균을 계산할 때 `NULL` 값이 있는 행을 무시합니다. 예:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

`units`열의 평균은 (0 + 20 + 40)/3 = 20이 됩니다. 101행과 103행은 이 계산에서 고려되지 않습니다.

ReferentialIntegrity

기본 데이터 세트의 열 세트 값이 어느 범위까지 참조 데이터 세트의 열 세트 값에 대한 하위 세트인지를 확인합니다.

구문

```
ReferentialIntegrity <PRIMARY_COLS> <REFERENCE_DATASET_COLS> <EXPRESSION>
```

- PRIMARY_COLS - 기본 데이터 세트의 쉼표로 구분된 열 이름 목록입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- REFERENCE_DATASET_COLS - 이 파라미터는 마침표로 구분된 두 부분을 포함합니다. 첫 번째 부분은 참조 데이터 세트의 별칭입니다. 두 번째 부분은 참조 데이터 세트에서 괄호로 묶인 쉼표로 구분된 열 이름 목록입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: zip code 열의 참조 무결성 확인

다음 예제 규칙은 기본 데이터 세트의 `zipcode` 열에 있는 값 중 90%가 넘는 항목이 `reference` 데이터 세트의 `zipcode` 열에 있는지 확인합니다.

```
ReferentialIntegrity "zipcode" "reference.zipcode" >= 0.9
```

예: city 및 state 열의 참조 무결성 확인

다음 예제에서는 city 및 state 정보가 포함된 열이 기본 데이터 세트과 참조 데이터 세트에 있습니다. 두 데이터 세트의 열 이름은 서로 다릅니다. 이 규칙은 기본 데이터 세트의 열 값 세트가 참조 데이터 세트의 열 값 세트와 정확히 같은지 확인합니다.

```
ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" = 1.0
```

샘플 동적 규칙

- `ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" > avg(last(10))`
- `ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" between min(last(10)) - 1 and max(last(10)) + 1`

RowCount

지정된 표현식을 기준으로 데이터 세트의 행 수를 확인합니다. 표현식에서 > 및 < 같은 연산자를 사용하여 행 수 또는 행 범위를 지정할 수 있습니다.

구문

```
RowCount <EXPRESSION>
```

- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 행 수 수치 검사

다음 예제 규칙은 행 수가 지정된 범위 내에 있는지 확인합니다.

```
RowCount between 10 and 100
RowCount between 1 and 50 where "Customer_ID < 10"
```

샘플 동적 규칙

```
RowCount > avg(last(10)) *0.8
```

RowCountMatch

주어진 표현식과 비교하여 기본 데이터 세트의 행 수와 참조 데이터 세트의 행 수 비율을 확인합니다.

구문

```
RowCountMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS - 행 수를 비교할 참조 데이터 세트의 별칭입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 참조 데이터 세트를 기준으로 행 수 확인

다음 예제 규칙은 기본 데이터 세트의 행 수가 참조 데이터 세트 행 수의 90% 이상인지 확인합니다.

```
RowCountMatch "reference" >= 0.9
```

StandardDeviation

지정된 표현식을 기준으로 열에 있는 모든 값의 표준 편차를 검사합니다.

구문

```
StandardDeviation <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 표준 편차

다음 예제 규칙은 colA 열에 있는 값의 표준 편차가 지정된 값보다 작은지 여부를 확인합니다.

```
StandardDeviation "Star_Rating" < 1.5
StandardDeviation "Salary" < 3500 where "Customer_ID < 10"
```

샘플 동적 규칙

- `StandardDeviation "colA" > avg(last(10) + 0.1`
- `StandardDeviation "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Null 동작

StandardDeviation 규칙은 표준 편차를 계산할 때 NULL 값이 있는 행을 무시합니다. 예:

```
+---+-----------+-----------+
|id |units1     |units2     |
+---+-----------+-----------+
|100|0          |0          |
|101|null       |0          |
|102|20         |20         |
|103|null       |0          |
|104|40         |40         |
```

```
+---+-----------+-----------+
```

units1 열의 표준 편차는 101열과 103열을 고려하지 않고 16.33으로 계산됩니다. units2 열의 표준 편차는 16이 됩니다.

Sum

지정된 표현식을 기준으로 열에 있는 모든 값의 합계를 확인합니다.

구문

```
Sum <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 합계

다음 예제 규칙은 열에 있는 모든 값의 합계가 지정된 임곗값을 초과하는지 여부를 확인합니다.

```
Sum "transaction_total" > 500000
Sum "Salary" < 55600 where "Customer_ID < 10"
```

샘플 동적 규칙

- `Sum "ColA" > avg(last(10))`
- `Sum "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Null 동작

Sum 규칙은 합계를 계산할 때 NULL 값이 있는 행을 무시합니다. 예:

```
+---+-----------+
|id |units      |
+---+-----------+
```

```
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

units 열의 합계는 101열과 103열을 고려하지 않고 (0 + 20 + 40) = 60으로 계산됩니다.

SchemaMatch

기본 데이터 세트의 스키마가 참조 데이터 세트의 스키마와 일치하는지 확인합니다. 스키마 검사는 열 단위로 수행됩니다. 이름이 동일하고 유형이 동일하면 두 열의 스키마가 일치합니다. 열 순서는 중요하지 않습니다.

구문

```
SchemaMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS - 스키마를 비교할 때 사용할 참조 데이터 세트의 별칭입니다.

  지원되는 열 유형: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: SchemaMatch

다음 예제 규칙은 기본 데이터 세트의 스키마가 참조 데이터 세트의 스키마와 정확히 일치하는지 확인합니다.

```
SchemaMatch "reference" = 1.0
```

Uniqueness

지정된 표현식을 기준으로 열에서 고유한 값의 백분율을 검사합니다. 고유한 값은 정확히 한 번 발생합니다.

구문

```
Uniqueness <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 섹션을 참조하세요.

예제

다음 예제 규칙은 열의 고유 값 백분율이 특정 숫자 기준과 일치하는지 여부를 확인합니다.

```
Uniqueness "email" = 1.0
Uniqueness "Customer_ID" != 1.0 where "Customer_ID < 10"
```

다음 예제 규칙은 여러 열을 확인합니다.

```
Uniqueness "vendorid" "tpep_pickup_datetime" = 1
```

샘플 동적 규칙

- `Uniqueness "colA" between min(last(10)) and max(last(10))`
- `Uniqueness "colA" >= avg(last(10))`

UniqueValueRatio

지정된 표현식을 기준으로 열의 고유 값 비율을 검사합니다. 고유 값 비율은 고유 값의 비율을 열에 있는 모든 고유 값의 수로 나눈 값입니다. 고유 값은 정확히 한 번 나타나는 반면 개별 값은 한 번 이상 나타납니다.

예를 들어 [a, a, b] 세트에는 고유 값 1개(b)와 개별 값 2개(a 및 b)가 포함됩니다. 따라서 세트의 고유 값 비율은 ½ = 0.5입니다.

구문

```
UniqueValueRatio <COL_NAME> <EXPRESSION>
```

- COL_NAME - 데이터 품질 규칙을 평가할 열의 이름입니다.

  지원되는 열 유형: 모든 열 유형
- EXPRESSION - 부울 값을 생성하기 위해 규칙 유형 응답에 대해 실행할 표현식입니다. 자세한 내용은 Expressions 단원을 참조하십시오.

예: 고유 값 비율

이 예에서는 열의 고유 값 비율을 값 범위와 비교하여 확인합니다.

```
UniqueValueRatio "test_score" between 0 and 0.5
UniqueValueRatio "Customer_ID" between 0 and 0.9 where "Customer_ID < 10"
```

샘플 동적 규칙

- `UniqueValueRatio "colA" > avg(last(10))`
- `UniqueValueRatio "colA" <= index(last(10),2) + std(last(5))`

DetectAnomalies

지정된 데이터 품질 규칙의 이상을 탐지합니다. DetectAnomalies 규칙을 실행할 때마다 지정된 규칙에 대한 평가 값이 저장됩니다. 수집된 데이터가 충분하면 이상 탐지 알고리즘이 해당 규칙에 대한 모든 기록 데이터를 가져와서 이상 탐지를 실행합니다. 이상이 탐지되면 DetectAnomalies 규칙이 실패합니다. 어떤 이상이 탐지되었는지에 대한 자세한 정보는 관찰에서 확인할 수 있습니다.

구문

```
    DetectAnomalies <RULE_NAME> <RULE_PARAMETERS>
```

RULE_NAME - 이상을 평가하고 탐지하려는 규칙의 이름입니다. 지원되는 규칙:

- "RowCount"
- "Completeness"
- "Uniqueness"
- "Mean"

- "Sum"
- "StandardDeviation"
- "Entropy"
- "DistinctValuesCount"
- "UniqueValueRatio"
- "ColumnLength"
- "ColumnValues"
- "ColumnCorrelation"
- "CustomSQL"
- "ColumnCount"

RULE_PARAMETERS - 일부 규칙을 실행하려면 추가 파라미터가 필요합니다. 필수 파라미터를 확인하려면 해당 규칙 설명서를 참조하세요.

예: RowCount에 대한 이상 항목

예를 들어 RowCount 이상을 탐지하려면 RowCount를 규칙 이름으로 입력합니다.

```
DetectAnomalies "RowCount"
```

예: ColumnLength에 대한 이상 항목

예를 들어 ColumnLength 이상을 탐지하려면 규칙 이름으로 ColumnLength와 열 이름을 입력합니다.

```
DetectAnomalies "ColumnLength" "id"
```

FileFreshness

FileFreshness를 사용하면 사용자가 제공한 조건에 따라 데이터 파일을 새로 고칩니다. 파일의 마지막 수정 시간을 사용하여 데이터 파일 또는 전체 폴더가 최신 상태인지 확인합니다.

이 규칙은 두 가지 지표를 수집합니다.

- 설정한 규칙에 따른 FileFreshness 규정 준수
- 규칙에 따라 스캔된 파일 수

```
{"Dataset.*.FileFreshness.Compliance":1,"Dataset.*.FileCount":1}
```

이상 탐지에서는 이러한 지표를 고려하지 않습니다.

파일 최신성 확인

다음 규칙은 tickets.parquet가 지난 24시간 동안 생성되도록 보장합니다.

```
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/tickets.parquet" >
 (now() - 24 hours)
```

폴더 최신성 확인

다음 규칙은 폴더의 모든 파일이 지난 24시간 이내에 생성되거나 수정된 경우 통과됩니다.

```
FileFreshness "s3://bucket/" >= (now() -1 days)
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" >= (now() - 24 hours)
```

임계치를 사용하여 폴더 또는 파일 최신성 확인

다음 규칙은 지난 10일 동안 "tickets" 폴더에 있는 파일의 10%가 생성되거나 수정된 경우 통과됩니다.

```
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" < (now() - 10 days)
 with threshold > 0.1
```

특정 날짜의 파일 또는 폴더 확인

특정 날짜의 파일 최신성을 확인할 수 있습니다.

```
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" > "2020-01-01"
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between "2023-01-01"
 and "2024-01-01"
```

시간 제한하여 파일 또는 폴더 확인

FileFreshness를 사용하여 특정 시간에 따라 파일이 도착했는지 확인할 수 있습니다.

```
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between now() and
 (now() - 45 minutes)
```

```
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between "9:30 AM" and
 "9:30 PM"
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" > (now() - 10 minutes)
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" > now()
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between (now() - 2
 hours) and (now() + 15 minutes)
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between (now() - 3
 days) and (now() + 15 minutes)
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between "2001-02-07"
 and (now() + 15 minutes)
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" > "21:45""
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" > "2024-01-01"
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between "02:30" and
 "04:30"
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" between "9:30 AM" and
 "22:15"
```

주요 고려 사항:

- FileFreshness는 일, 시간 및 분 단위를 사용하여 파일을 평가할 수 있습니다.
- 경우에 따라 오전/오후 및 24시간을 지원합니다.
- 재정의가 지정되지 않는 한 시간은 UTC로 계산됩니다.
- 날짜는 00:00 시간의 UTC로 계산됩니다.

시간 기반 FileFreshness는 다음과 같이 작동합니다.

```
FileFreshness "s3://amzn-s3-demo-bucket/artifacts/file/tickets/" > "21:45"
```

- 먼저 시간 “21:45”를 UTC 형식의 오늘 날짜와 결합하여 날짜-시간 필드를 생성합니다.
- 다음으로 날짜-시간이 지정한 시간대로 변환됩니다.
- 마지막으로 규칙이 평가됩니다.

선택적 파일 기반 규칙 태그:

태그를 사용하면 규칙 동작을 제어할 수 있습니다.

recentFiles

이 태그는 가장 최근 파일을 먼저 유지하여 처리되는 파일 수를 제한합니다.

```
FileFreshness "s3://amzn-s3-demo-bucket/" between (now() - 100 minutes) and (now() + 10
 minutes) with recentFiles = 1
```

timeZone

허용되는 시간대 재정의는 지원되는 시간대에 Allowed Time Zones를 참조하세요.

```
FileFreshness "s3://path/" > "21:45" with timeZone = "America/New_York"
```

```
FileFreshness "s3://path/" > "21:45" with timeZone = "America/Chicago"
```

```
FileFreshness "s3://path/" > "21:45" with timeZone = "Europe/Paris"
```

```
FileFreshness "s3://path/" > "21:45" with timeZone = "Asia/Shanghai"
```

```
FileFreshness "s3://path/" > "21:45" with timeZone = "Australia/Darwin"
```

데이터 프레임에서 직접 파일 이름 추론

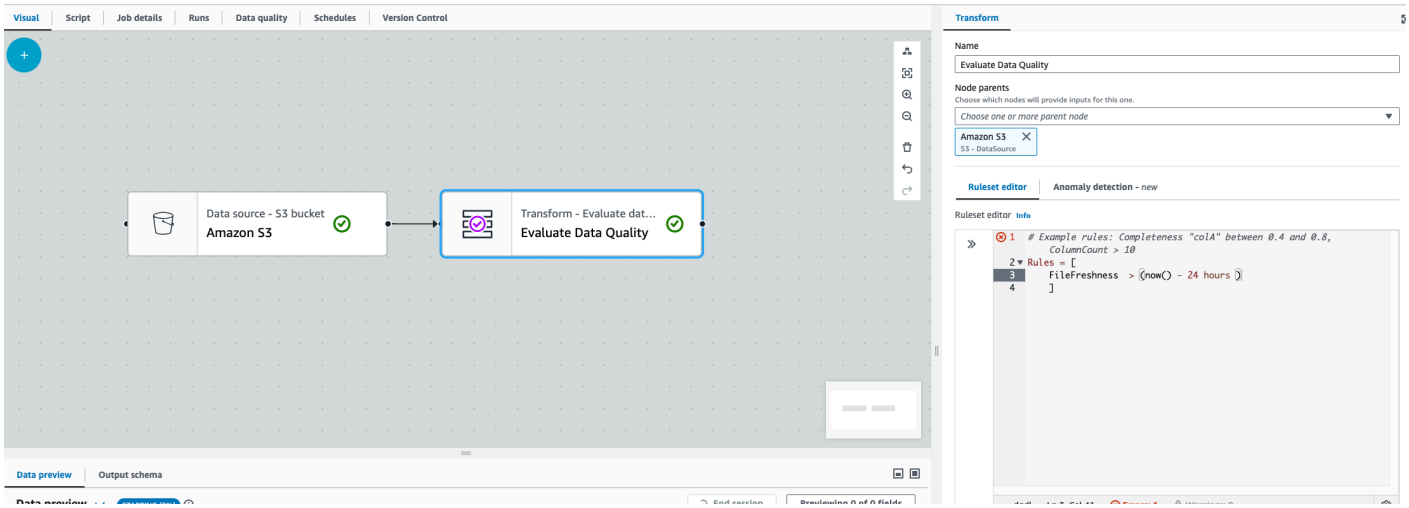
항상 파일 경로를 제공하지 않아도 됩니다. 예를 들어 AWS Glue Data Catalog에서 규칙을 작성할 때 카탈로그 테이블에서 사용 중인 폴더를 찾기 어려울 수 있습니다. AWS Glue Data Quality에서는 데이터 프레임을 채우는 데 사용되는 특정 폴더나 파일을 찾고 해당 항목이 최신 상태인지를 감지할 수 있습니다.

> **Note**
>
> 이 기능은 파일이 DynamicFrame 또는 DataFrame으로 성공적으로 읽기된 경우에만 작동합니다.

```
FileFreshness > (now() - 24 hours)
```

이 규칙은 동적 프레임 또는 데이터 프레임을 채우는 데 사용되는 폴더 경로나 파일을 찾습니다. Amazon S3 경로 또는 Amazon S3 기반 AWS Glue Data Catalog 테이블에 대해 사용할 수 있습니다. 이때 몇 가지 고려 사항이 있습니다.

1. AWS Glue ETL에서 Amazon S3 또는 AWS Glue Data Catalog 변환 직후 EvaluateDataQuality 변환이 있어야 합니다.



2. 이 규칙은 AWS Glue 대화형 세션에서 작동하지 않습니다.

두 경우 모두에서 시도하거나 AWS Glue에서 파일을 찾을 수 없는 경우 AWS Glue에서 다음 오류가 발생합니다. “`Unable to parse file path from DataFrame`”

FileMatch

FileMatch 규칙을 사용하면 파일을 다른 파일 또는 체크섬과 비교할 수 있습니다. 이는 다음과 같은 몇 가지 시나리오에서 유용할 수 있습니다.

1. 외부 소스에서 수신한 파일 검증: FileMatch를 사용하여 체크섬과 비교해 외부 소스에서 올바른 파일을 수신하도록 보장할 수 있습니다. 이 방식으로 수집 중인 데이터의 무결성을 검증할 수 있습니다.
2. 서로 다른 두 폴더의 데이터 비교: FileMatch를 사용하여 두 폴더 간의 파일을 비교할 수 있습니다.

이 규칙은 규칙에 따라 스캔한 파일 수라는 하나의 지표를 수집합니다.

```
{"Dataset.*.FileCount":1}
```

체크섬으로 파일 검증:

FileMatch는 파일과 설정된 체크섬을 수락하여 하나 이상의 체크섬이 파일과 일치하는지 확인합니다.

```
FileMatch "s3://amzn-s3-demo-bucket/file.json" in ["3ee0d8617ac041793154713e5ef8f319"]
 with hashAlgorithm = "MD5"
```

```
FileMatch "s3://amzn-s3-demo-bucket/file.json" in ["3ee0d8617ac041793154713e5ef8f319"]
 with hashAlgorithm = "SHA-1"
FileMatch "s3://amzn-s3-demo-bucket/file.json" in ["3ee0d8617ac041793154713e5ef8f319"]
 with hashAlgorithm = "SHA-256"
FileMatch "s3://amzn-s3-demo-bucket/file.json" in ["3ee0d8617ac041793154713e5ef8f319"]
```

다음 표준 알고리즘이 지원됩니다.

- MD5
- SHA-1
- SHA-256

알고리즘을 제공하지 않으면 기본값은 SHA-256입니다.

체크섬 세트로 폴더의 모든 파일 검증:

```
FileMatch "s3://amzn-s3-demo-bucket /" in ["3ee0d8617ac041793154713e5ef8f319",
 "7e8617ac041793154713e5ef8f319"] with hashAlgorithm = "MD5"
FileMatch "s3://amzn-s3-demo-bucket /internal-folder/" in
 ["3ee0d8617ac041793154713e5ef8f319", "7e8617ac041793154713e5ef8f319"]
```

다른 폴더의 파일 비교

```
FileMatch "s3://original_bucket/" "s3://archive_bucket/"
FileMatch "s3://original_bucket/internal-folder/" "s3://original_bucket/other-folder/"
```

FileMatch는 `original_bucket`에서 파일 콘텐츠를 확인하고 `archive_bucket`의 내용과 일치하는지 확인합니다. 정확히 일치하지 않으면 규칙은 실패합니다. 내부 폴더 또는 개별 파일의 콘텐츠를 확인할 수도 있습니다.

FileMatch는 개별 파일을 서로 비교하여 확인할 수도 있습니다.

```
FileMatch "s3://amzn-s3-demo-bucket /file_old.json" "s3://amzn-s3-demo-bucket /
file_new.json"
```

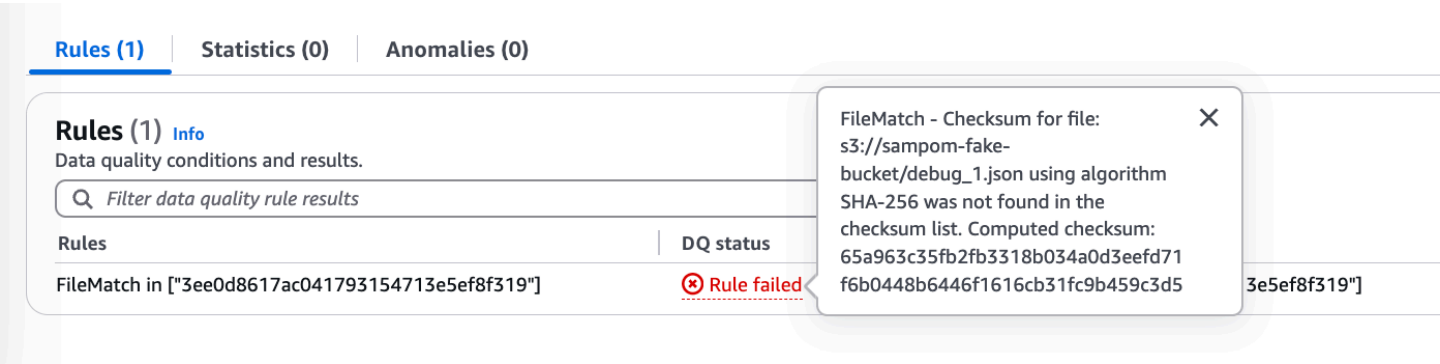
데이터 프레임에서 직접 파일 이름 추론

항상 파일 경로를 제공하지 않아도 됩니다. 예를 들어 AWS Glue Data Catalog(Amazon S3 지원)에서 규칙을 작성하는 경우 카탈로그 테이블에서 사용 중인 폴더를 찾기 어려울 수 있습니다. AWS Glue Data Quality에서는 데이터 프레임을 채우는 데 사용되는 특정 폴더나 파일을 찾을 수 있습니다.

> **Note**
>
> 이 기능은 파일이 DynamicFrame 또는 DataFrame으로 성공적으로 읽기된 경우에만 작동합니다.

```
FileMatch in ["3ee0d8617ac041793154713e5ef8f319"] with hashAlgorithm = "MD5"
FileMatch in ["3ee0d8617ac041793154713e5ef8f319"] with hashAlgorithm = "SHA-1"
FileMatch in ["3ee0d8617ac041793154713e5ef8f319"] with hashAlgorithm = "SHA-256"
FileMatch in ["3ee0d8617ac041793154713e5ef8f319"]
```

제공된 체크섬이 계산된 체크섬과 다른 경우 FileMatch는 차이를 알립니다.



선택적 파일 기반 규칙 태그:

태그를 사용하면 규칙 동작을 제어할 수 있습니다.

recentFiles

이 태그는 가장 최근 파일을 먼저 유지하여 처리되는 파일 수를 제한합니다.

```
FileMatch "s3://amzn-s3-demo-bucket/file.json" in ["3ee0d8617ac04179sam4713e5ef8f319"]
 with recentFiles = 1
```

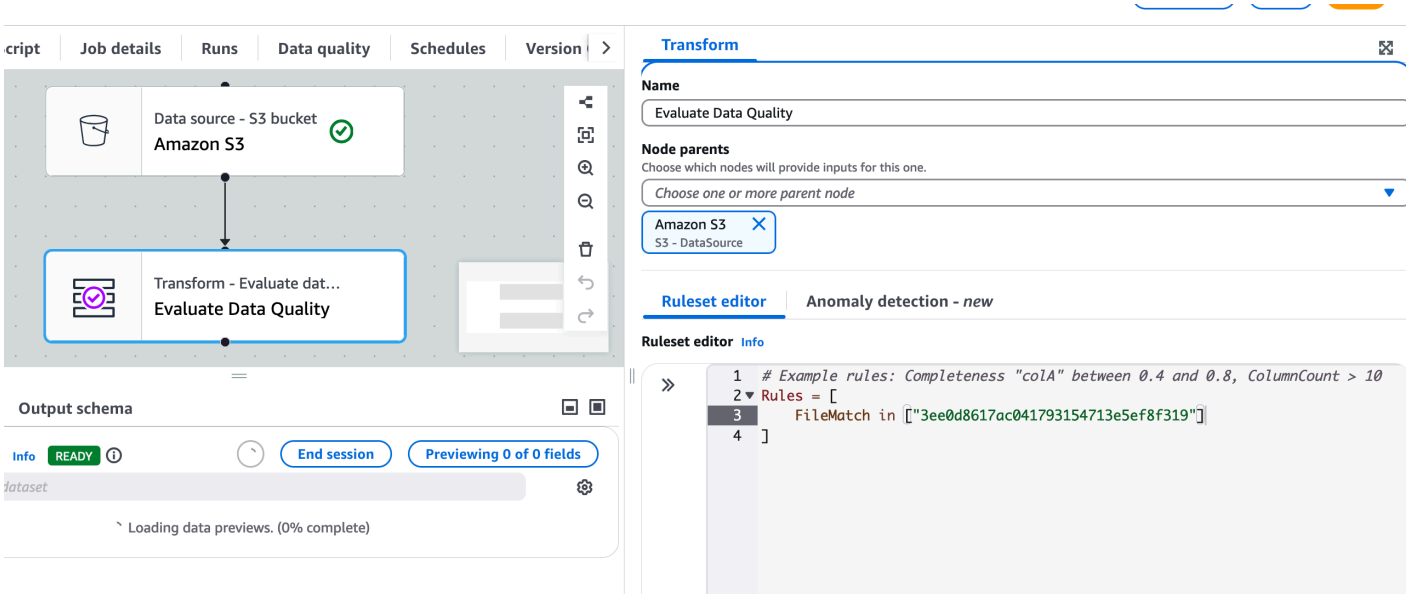
matchFileName

이 태그는 파일의 이름이 중복되지 않게 합니다. 기본 동작은 false입니다.

```
FileMatch "s3://amzn-s3-demo-bucket/file.json" in ["3ee0d8617ac04179sam4713e5ef8f319"]
 with matchFileName = "true"
```

이때 몇 가지 고려 사항이 있습니다.

1. AWS Glue ETL에서 Amazon S3 또는 AWS Glue Data Catalog 변환 직후 EvaluateDataQuality 변환이 있어야 합니다.



2. 이 규칙은 AWS Glue 대화형 세션에서 작동하지 않습니다.

FileUniqueness

파일 고유성을 사용하면 데이터 생산자로부터 수신한 데이터에 중복 파일이 없는지 확인할 수 있습니다.

다음 데이터 통계를 수집합니다.

1. 규칙에 따라 스캔된 파일 수
2. 파일의 고유성 비율

```
Dataset.*.FileUniquenessRatio: 1.00, Dataset.*.FileCount: 8.00
```

폴더에서 중복 파일 찾기:

```
FileUniqueness "s3://bucket/" > 0.5
FileUniqueness "s3://bucket/folder/" = 1
```

데이터 프레임에서 직접 폴더 이름을 추론하여 중복 감지:

항상 파일 경로를 제공하지 않아도 됩니다. 예를 들어 AWS Glue Data Catalog에서 규칙을 작성할 때 카탈로그 테이블에서 사용 중인 폴더를 찾기 어려울 수 있습니다. AWS Glue Data Quality에서는 데이터 프레임을 채우는 데 사용되는 특정 폴더나 파일을 찾을 수 있습니다.

> **Note**
>
> 추론을 사용하는 경우 파일 기반 규칙은 DynamicFrame 또는 DataFrame으로 성공적으로 읽은 파일만 감지할 수 있습니다.

```
FileUniqueness > 0.5
```

선택적 파일 기반 규칙 태그:

태그를 사용하면 규칙 동작을 제어할 수 있습니다.

recentFiles

이 태그는 가장 최근 파일을 먼저 유지하여 처리되는 파일 수를 제한합니다.

```
FileUniqueness "s3://amzn-s3-demo-bucket/" > 0.5 with recentFiles = 1
```

matchFileName

이 태그는 파일의 이름이 중복되지 않게 합니다. 기본 동작은 false입니다.

```
FileUniqueness "s3://amzn-s3-demo-bucket/" > 0.5 with matchFileName = "true"
```

이때 몇 가지 고려 사항이 있습니다.

1. AWS Glue ETL에서 Amazon S3 또는 AWS Glue Data Catalog 변환 직후 EvaluateDataQuality 변환이 있어야 합니다.
2. 이 규칙은 AWS Glue 대화형 세션에서 작동하지 않습니다.

FileSize

FileSize 규칙 유형을 사용하면 파일이 특정 파일 크기 기준을 충족하는지 확인할 수 있습니다. 다음 사용 사례에 유용합니다.

1. 생산자가 처리를 위해 빈 파일이나 상당히 작은 파일을 전송하지 않는지 확인합니다.

2. 대상 버킷에 성능 문제로 이어질 수 있는 더 작은 파일이 없는지 확인합니다.

FileSize는 다음 지표를 수집합니다.

1. 규정 준수: 설정한 규칙 임계치를 충족하는 파일의 비율(%) 반환
2. 파일 수: 규칙에 따라 스캔된 파일 수
3. 최소 파일 크기(바이트)
4. 최대 파일 크기(바이트)

```
Dataset.*.FileSize.Compliance: 1.00,
Dataset.*.FileCount: 8.00,
Dataset.*.MaximumFileSize: 327413121.00,
Dataset.*.MinimumFileSize: 204558920.00
```

이러한 지표에 대해서는 이상 탐지가 지원되지 않습니다.

파일 크기 검증

file.dat가 2MB를 초과하면 이 규칙이 통과됩니다.

```
FileSize "s3://amzn-s3-demo-bucket/file.dat" > 2 MB
```

지원되는 단위에는 B(바이트), MB(메가 바이트), GB(기가 바이트) 및 TB(테라 바이트)가 포함됩니다.

폴더에서 파일 크기 검증

```
FileSize "s3://bucket/" > 5 B
FileSize "s3://bucket/" < 2 GB
```

s3://amzn-s3-demo-bucket의 파일 중 70%가 2GB~1TB인 경우 이 규칙을 통과합니다.

```
FileSize "s3://amzn-s3-demo-bucket/" between 2 GB and 1 TB  with threshold > 0.7
```

데이터 프레임에서 직접 파일 이름 추론

항상 파일 경로를 제공하지 않아도 됩니다. 예를 들어 Data Catalog에서 규칙을 작성할 때 카탈로그 테이블에서 사용 중인 폴더를 찾기 어려울 수 있습니다. AWS Glue Data Quality에서는 데이터 프레임을 채우는 데 사용되는 특정 폴더나 파일을 찾을 수 있습니다.

> **Note**
>
> 이 기능은 파일이 DynamicFrame 또는 DataFrame으로 성공적으로 읽기된 경우에만 작동합니다.

```
FileSize < 10 MB with threshold > 0.7
```

선택적 파일 기반 규칙 태그:

태그를 사용하면 규칙 동작을 제어할 수 있습니다.

recentFiles

이 태그는 가장 최근 파일을 먼저 유지하여 처리되는 파일 수를 제한합니다.

```
FileSize "s3://amzn-s3-demo-bucket/" > 5 B with recentFiles = 1
```

matchFileName

이 태그는 파일의 이름이 중복되지 않게 합니다. 기본 동작은 false입니다.

```
FileSize "s3://amzn-s3-demo-bucket/" > 5 B with matchFileName = "true"
```

이때 몇 가지 고려 사항이 있습니다.

1. AWS Glue ETL에서는 Amazon S3 또는 Data Catalog 변환 직후 DataQuality 평가 변환이 있어야 합니다.
2. 이 규칙은 AWS Glue 대화형 세션에서 작동하지 않습니다.

API를 사용하여 데이터 품질 측정 및 관리

이 주제에서는 API를 사용하여 데이터 품질을 측정하고 관리하는 방법에 대해 설명합니다.

목차

- 사전 조건
- AWS Glue Data Quality 권장 사용

- AWS Glue Data Quality 규칙 세트 사용
- AWS Glue Data Quality 실행 사용
- AWS Glue Data Quality 결과 사용

사전 조건

- 최신 AWS Glue Data Quality API가 포함되도록 boto3 버전이 최신 버전인지 확인합니다.
- 최신 CLI를 포함하도록 AWS CLI 버전이 최신인지 확인합니다.

AWS Glue 작업을 사용하여 이러한 API를 실행하는 경우 다음 옵션을 사용하여 boto3 라이브러리를 최신 버전으로 업데이트할 수 있습니다.

```
—additional-python-modules boto3==<version>
```

AWS Glue Data Quality 권장 사용

AWS Glue Data Quality 권장 실행을 시작하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def start_data_quality_rule_recommendation_run(self, database_name, table_name,
 role_arn):
        """
        Starts a recommendation run that is used to generate rules when you don't know
 what rules to write. AWS Glue Data Quality
        analyzes the data and comes up with recommendations for a potential ruleset.
 You can then triage the ruleset
        and modify the generated ruleset to your liking.

        :param database_name: The name of the AWS Glue database which contains the
 dataset.
        :param table_name: The name of the AWS Glue table against which we want a
 recommendation
```

```
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access
 Management (IAM) role that grants permission to let AWS Glue access the resources it
 needs.

        """
        try:
            response = self.client.start_data_quality_rule_recommendation_run(
                DataSource={
                    'GlueTable': {
                        'DatabaseName': database_name,
                        'TableName': table_name
                    }
                },
                Role=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't start data quality recommendation run %s. Here's why: %s:
%s", name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['RunId']
```

권장 실행의 경우 pushDownPredicates 또는 catalogPartitionPredicates를 사용하여 성능을 개선하고 카탈로그 소스의 특정 파티션에서만 권장을 실행할 수 있습니다.

```
client.start_data_quality_rule_recommendation_run(
            DataSource={
                'GlueTable': {
                    'DatabaseName': database_name,
                    'TableName': table_name,
                    'AdditionalOptions': {
                        'pushDownPredicate': "year=2022"
                    }
                }
            },
            Role=role_arn,
            NumberOfWorkers=2,
            CreatedRulesetName='<rule_set_name>'
  )
```

AWS Glue Data Quality 권장 실행 결과를 얻으려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_rule_recommendation_run(self, run_id):
        """
        Gets the specified recommendation run that was used to generate rules.

        :param run_id: The id of the data quality recommendation run

        """
        try:
            response =
 self.client.get_data_quality_rule_recommendation_run(RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get data quality recommendation run %. Here's why: %s: %s",
 run_id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

위의 응답 객체에서 실행에서 권장한 규칙 세트를 추출하여 이후 단계에서 사용할 수 있습니다.

```
print(response['RecommendedRuleset'])

Rules = [
    RowCount between 2000 and 8000,
    IsComplete "col1",
    IsComplete "col2",
    StandardDeviation "col3" between 58138330.8 and 64258155.09,
    ColumnValues "col4" between 1000042965 and 1214474826,
    IsComplete "col5"
]
```

필터링 및 나열할 수 있는 모든 권장 실행 목록을 가져오려면:

```
response = client.list_data_quality_rule_recommendation_runs(
    Filter={
        'DataSource': {
            'GlueTable': {
                'DatabaseName': '<database_name>',
                'TableName': '<table_name>'
            }
        }
)
```

기존 AWS Glue Data Quality 권장 작업을 취소하려면:

```
response = client.cancel_data_quality_rule_recommendation_run(
    RunId='dqrun-d4b6b01957fdd79e59866365bf9cb0e40fxxxxxxx'
)
```

AWS Glue Data Quality 규칙 세트 사용

AWS Glue Data Quality 규칙 세트를 생성하려면:

```
response = client.create_data_quality_ruleset(
    Name='<ruleset_name>',
    Ruleset='Rules = [IsComplete "col1", IsPrimaryKey "col2", RowCount between 2000 and
 8000]',
    TargetTable={
        'TableName': '<table_name>',
        'DatabaseName': '<database_name>'
    }
)
```

데이터 품질 규칙 세트를 가져오려면:

```
response = client.get_data_quality_ruleset(
    Name='<ruleset_name>'
)
print(response)
```

이 API를 사용하여 규칙 세트를 추출할 수 있습니다.

```
print(response['Ruleset'])
```

테이블의 모든 데이터 품질 규칙 세트를 나열하려면:

```
response = client.list_data_quality_rulesets()
```

API 내의 필터 조건을 사용하여 특정 데이터베이스 또는 테이블에 연결된 모든 규칙 세트를 필터링할 수 있습니다.

```
response = client.list_data_quality_rulesets(
    Filter={
        'TargetTable': {
            'TableName': '<table_name>',
            'DatabaseName': '<database_name>'
        }
    },
)
```

데이터 품질 규칙 세트를 업데이트하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def update_data_quality_ruleset(self, ruleset_name, ruleset_string):
        """
        Update an AWS Glue Data Quality Ruleset

        :param ruleset_name: The name of the AWS Glue Data Quality ruleset to update
        :param ruleset_string: The DQDL ruleset string to update the ruleset with

        """
        try:
            response = self.client.update_data_quality_ruleset(
                Name=ruleset_name,
                Ruleset=ruleset_string
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't update the AWS Glue Data Quality ruleset. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

데이터 품질 규칙 세트를 삭제하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def delete_data_quality_ruleset(self, ruleset_name):
        """
        Delete a AWS Glue Data Quality Ruleset

        :param ruleset_name: The name of the AWS Glue Data Quality ruleset to delete

        """
        try:
            response = self.client.delete_data_quality_ruleset(
                Name=ruleset_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete the AWS Glue Data Quality ruleset. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

AWS Glue Data Quality 실행 사용

AWS Glue Data Quality 실행을 시작하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def start_data_quality_ruleset_evaluation_run(self, database_name, table_name,
role_name, ruleset_list):
        """
        Start an AWS Glue Data Quality evaluation run

        :param database_name: The name of the AWS Glue database which contains the
dataset.
        :param table_name: The name of the AWS Glue table against which we want to
evaluate.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access
Management (IAM) role that grants permission to let AWS Glue access the resources it
needs.
        :param ruleset_list: The list of AWS Glue Data Quality ruleset names to
evaluate.

        """
        try:
            response = client.start_data_quality_ruleset_evaluation_run(
                DataSource={
                    'GlueTable': {
                        'DatabaseName': database_name,
                        'TableName': table_name
                    }
                },
                Role=role_name,
                RulesetNames=ruleset_list
            )
        except ClientError as err:
            logger.error(
                "Couldn't start the AWS Glue Data Quality Run. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['RunId']
```

pushDownPredicate 또는 catalogPartitionPredicate 파라미터를 전달하여 카탈로그 테이블 내의 특정 파티션 세트만 대상으로 데이터 품질 실행을 수행할 수 있습니다. 예:

```
response = client.start_data_quality_ruleset_evaluation_run(
    DataSource={
        'GlueTable': {
            'DatabaseName': '<database_name>',
            'TableName': '<table_name>',
            'AdditionalOptions': {
                'pushDownPredicate': 'year=2023'
            }
        }
    },
    Role='<role_name>',
    NumberOfWorkers=5,
    Timeout=123,
    AdditionalRunOptions={
        'CloudWatchMetricsEnabled': False
    },
    RulesetNames=[
        '<ruleset_name>',
    ]
)
```

또한 ROW 또는 COLUMN 수준에서 규칙 세트의 복합 규칙을 평가하는 방법도 구성할 수 있습니다. 복합 규칙의 작동 방식에 대한 자세한 내용은 설명서에서 복합 규칙 작동 방식을 참조하십시오.

요청에서 복합 규칙 평가 방법을 설정하는 방법을 보여주는 예:

```
response = client.start_data_quality_ruleset_evaluation_run(
    DataSource={
        'GlueTable': {
            'DatabaseName': '<database_name>',
            'TableName': '<table_name>',
            'AdditionalOptions': {
                'pushDownPredicate': 'year=2023'
            }
        }
    },
    Role='<role_name>',
    NumberOfWorkers=5,
    Timeout=123,
    AdditionalRunOptions={
        'CompositeRuleEvaluationMethod':ROW
    },
    RulesetNames=[
```

```
        '<ruleset_name>',
    ]
)
```

AWS Glue Data Quality 실행에 대한 정보를 가져오려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_ruleset_evaluation_run(self, run_id):
        """
        Get details about an AWS Glue Data Quality Run

        :param run_id: The AWS Glue Data Quality run ID to look up

        """
        try:
            response = self.client.get_data_quality_ruleset_evaluation_run(
                RunId=run_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't look up the AWS Glue Data Quality run ID. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

AWS Glue Data Quality 실행 결과를 가져오려면:

해당 AWS Glue Data Quality 실행에 대해 다음 방법을 사용하여 실행 평가 결과를 추출할 수 있습니다.

```
response = client.get_data_quality_ruleset_evaluation_run(
    RunId='d4b6b01957fdd79e59866365bf9cb0e40fxxxxxxx'
)
```

```
resultID = response['ResultIds'][0]

response = client.get_data_quality_result(
    ResultId=resultID
)

print(response['RuleResults'])
```

AWS Glue Data Quality 실행을 모두 나열하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def list_data_quality_ruleset_evaluation_runs(self, database_name, table_name):
        """
        Lists all the AWS Glue Data Quality runs against a given table

        :param database_name: The name of the database where the data quality runs
        :param table_name: The name of the table against which the data quality runs
 were created

        """
        try:
            response = self.client.list_data_quality_ruleset_evaluation_runs(
                Filter={
                    'DataSource': {
                        'GlueTable': {
                            'DatabaseName': database_name,
                            'TableName': table_name
                        }
                    }
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't list the AWS Glue Quality runs. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

```
        else:
            return response
```

특정 시간 사이의 결과 또는 특정 테이블에 대해 실행된 결과만 표시하도록 filter 절을 수정할 수 있습니다.

진행 중인 AWS Glue Data Quality 실행을 중지하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def cancel_data_quality_ruleset_evaluation_run(self, result_id):
        """
        Cancels a given AWS Glue Data Quality run

        :param result_id: The result id of a AWS Glue Data Quality run to cancel

        """
        try:
            response = self.client.cancel_data_quality_ruleset_evaluation_run(
                ResultId=result_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't cancel the AWS Glue Data Quality run. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

AWS Glue Data Quality 결과 사용

AWS Glue Data Quality 실행 결과를 가져오려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
```

```
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_result(self, result_id):
        """
        Outputs the result of an AWS Glue Data Quality Result

        :param result_id: The result id of an AWS Glue Data Quality run

        """
        try:
            response = self.client.get_data_quality_result(
                ResultId=result_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get the AWS Glue Data Quality result. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

특정 데이터 품질 결과에 대해 수집된 통계를 보려면:

```
import boto3
from botocore.exceptions import ClientError
import logging

logger = logging.getLogger(__name__)
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_profile_for_data_quality_result(self, result_id):
        """
        Outputs the statistic profile for a AWS Glue Data Quality Result
```

```
        :param result_id: The result id of a AWS Glue Data Quality run

        """
        try:
            response = self.glue_client.get_data_quality_result(
                ResultId=result_id
            )

            # the profile contains all statistics gathered for the result
            profile_id = response['ProfileId']
            profile = self.glue_client.list_data_quality_statistics(
                ProfileId = profile_id
            )
            return profile
        except ClientError as err:
            logger.error(
                "Couldn't retrieve Data Quality profile. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

여러 데이터 품질 실행에서 수집된 통계의 시계열을 보려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_statistics_for_data_quality_result(self, profile_id):
        """
        Outputs an array of datapoints for each statistic in the input result.

        :param result_id: The profile id of a AWS Glue Data Quality run

        """
        try:
            profile = self.glue_client.list_data_quality_statistics(
                ProfileId = profile_id
            )
            statistics = [self.glue_client.list_data_quality_statistics(
                StatisticId = s['StatisticId']
```

```
            ) for s in profile['Statistics']]
            return statistics
        except ClientError as err:
            logger.error(
                "Couldn't retrieve Data Quality statistics. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

특정 통계에 대한 이상 탐지 모델을 보려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_model_training_result_for_statistic(self, statistic_id, profile_id):
        """
        Outputs the details (bounds) of anomaly detection training for the given
 statistic at the given profile.

        :param statistic_id the model's statistic (the timeseries it is tracking)
        :param profile_id the profile associated with the model (a point in the
 timeseries)

        """
        try:
            model = self.glue_client.get_data_quality_model_result(
                ProfileId = profile_id, StatisticId = statistic_id
            )
            return model
        except ClientError as err:
            logger.error(
                "Couldn't retrieve Data Quality model results. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

통계 모델의 이상 탐지 기준에서 데이터 포인트를 제외하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def apply_exclusions_to_statistic(self, statistic_id, profile_ids):
        """
        Annotate some points along a given statistic timeseries.

        This example excludes the provided values; INCLUDE can also be used to undo
 this action.

        :param statistic_id the statistic timeseries to annotate
        :param profile_id the profiles we want to exclude (points in the timeseries)

        """

        try:
            response = self.glue_client.batch_put_data_quality_statistic_annotation(
                    InclusionAnnotations = [
                        {'ProfileId': prof_id,
                        'StatisticId': statistic_id,
                        'InclusionAnnotation': 'EXCLUDE'} for prof_id in profile_ids
                    ]
            )
            return response['FailedInclusionAnnotations']
        except ClientError as err:
            logger.error(
                "Couldn't store Data Quality annotations. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

특정 통계에 대한 이상 탐지 모델을 보려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client
```

```
    def get_model_training_status_for_statistic(self, statistic_id, profile_id):
        """
        Outputs the status of anomaly detection training for the given statistic at the
 given profile.

        :param statistic_id the model's statistic (the timeseries it is tracking)
        :param profile_id the profile associated with the model (a point in the
 timeseries)

        """
        try:
            model = self.glue_client.get_data_quality_model(
                ProfileId = profile_id, StatisticId = statistic_id
            )
            return model
        except ClientError as err:
            logger.error(
                "Couldn't retrieve Data Quality statistics. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

이상 탐지 기준에서 특정 데이터 품질 실행의 모든 결과를 제외하려면:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def apply_exclusions_to_profile(self, profile_id):
        """
        Exclude datapoints produced by a run across statistic timeseries.

        This example excludes the provided values; INCLUDE can also be used to undo
 this action.

        :param profile_id the profiles we want to exclude (points in the timeseries)

        """
        try:
            response = self.glue_client.put_data_quality_profile_annotation(
```

```
                    ProfileId = profile_id,
                    InclusionAnnotation = "EXCLUDE"
            )
            return response
        except ClientError as err:
            logger.error(
                "Couldn't store Data Quality annotations. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

특정 데이터 품질 실행의 결과를 가져와 결과를 표시하려면:

AWS Glue Data Quality `runID`를 사용되면 아래와 같이 `resultID`을(를) 추출하여 실제 결과를 얻을 수 있습니다.

```
response = client.get_data_quality_ruleset_evaluation_run(
    RunId='dqrun-abca77ee126abe1378c1da1ae0750d7dxxxx'
)

resultID = response['ResultIds'][0]

response = client.get_data_quality_result(
    ResultId=resultID
)

print(resp['RuleResults'])
```

알림, 배포 및 예약 설정

이 주제에서는 AWS Glue Data Quality에 대한 알림, 배포 및 예약을 설정하는 방법을 설명합니다.

목차

- Amazon EventBridge 통합에서 알림 설정
  - 이벤트 패턴의 추가 구성 옵션
  - 알림을 이메일 형식으로 지정
- CloudWatch 통합에서 경보 및 알림 설정
- 데이터 품질 결과를 쿼리하여 대시보드 구축
- AWS CloudFormation을 사용하여 데이터 품질 규칙 배포

- 데이터 품질 규칙 예약

Amazon EventBridge 통합에서 알림 설정

AWS Glue Data Quality는 데이터 품질 규칙 세트 평가 실행 완료 시 생성되는 EventBridge 이벤트의 게시를 지원합니다. 이를 통해 데이터 품질 규칙 실패 시 알림을 쉽게 설정할 수 있습니다.

다음은 데이터 카탈로그에서 데이터 품질 규칙 세트를 평가할 때 발생하는 샘플 이벤트입니다. 이 정보를 사용하여 Amazon EventBridge에서 사용할 수 있는 데이터를 검토할 수 있습니다. 추가 API 직접 호출을 통해 자세한 내용을 확인할 수 있습니다. 예를 들어 결과 ID로 `get_data_quality_result` API를 직접 호출하여 특정 실행의 세부 정보를 가져옵니다.

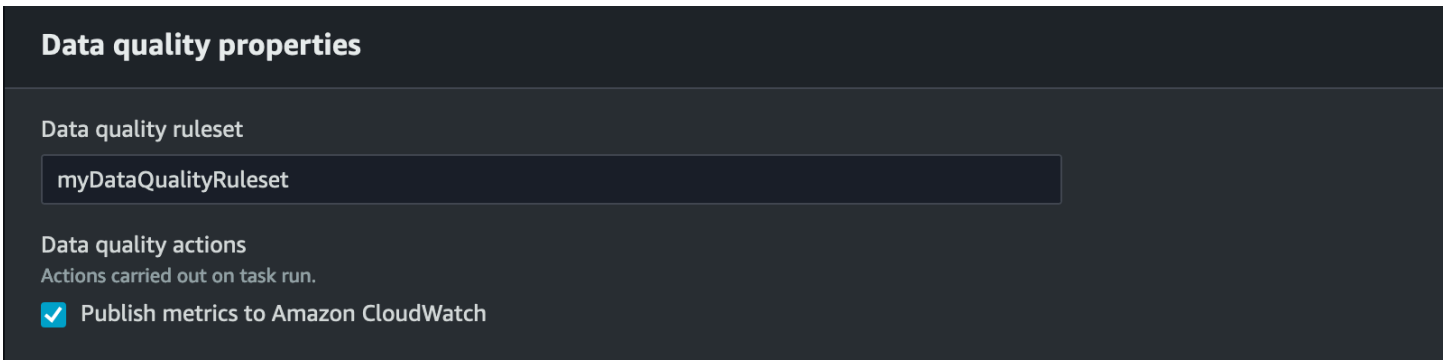
```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Data Quality Evaluation Results Available",
    "source":"aws.glue-dataquality",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
        "context": {
                    "contextType": "GLUE_DATA_CATALOG",
                    "runId":"dqrun-12334567890",
                    "databaseName": "db-123",
                    "tableName": "table-123",
                    "catalogId": "123456789012"
                    },
        "resultID": "dqresult-12334567890",
        "rulesetNames": ["rulset1"],
        "state":"SUCCEEDED",
        "score": 1.00,
        "rulesSucceeded": 100,
        "rulesFailed": 0,
        "rulesSkipped": 0
    }
}
```

다음은 AWS Glue ETL 또는 AWS Glue Studio 노트북에서 데이터 품질 규칙 세트를 평가할 때 게시되는 샘플 이벤트입니다.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Data Quality Evaluation Results Available",
    "source":"aws.glue-dataquality",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
        "context": {
                    "contextType": "GLUE_JOB",
                    "jobId": "jr-12334567890",
                    "jobName": "dq-eval-job-1234",
                    "evaluationContext": "",
                    }
        "resultID": "dqresult-12334567890",
        "rulesetNames": ["rulset1"],
        "state":"SUCCEEDED",
        "score": 1.00
        "rulesSucceeded": 100,
        "rulesFailed": 0,
        "rulesSkipped": 0
    }
}
```

Data Catalog와 ETL 작업 모두에서 데이터 품질 평가를 실행하는 경우, 기본적으로 선택되는 Amazon CloudWatch에 지표 게시 옵션을 선택한 상태로 유지해야 EventBridge에 게시할 수 있습니다.

EventBridge 알림 설정



생성된 이벤트를 수신하고 대상을 정의하려면 Amazon EventBridge 규칙을 구성해야 합니다. 규칙을 생성하려면:

1. Amazon EventBridge 콘솔을 엽니다.
2. 탐색 표시줄의 버스 섹션에서 규칙을 선택합니다.
3. [Create Rule]을 선택합니다.
4. 규칙 세부 정보 정의에서:
   a. 이름에 `myDQRule`을 입력합니다.
   b. 설명을 입력합니다(선택 사항).
   c. 이벤트 버스의 경우 이벤트 버스를 선택합니다. 이벤트 버스가 아직 없는 경우 기본값으로 둡니다.
   d. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택하고 다음을 선택합니다.
5. 이벤트 패턴 작성에서:
   a. 이벤트 소스에서 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
   b. 샘플 이벤트 섹션은 건너뜁니다.
   c. 생성 방법으로 패턴 양식 사용을 선택합니다.
   d. 이벤트 패턴의 경우:
      i. 이벤트 소스에 대한 AWS 서비스를 선택합니다.
      ii. AWS 서비스에 대한 Glue Data Quality를 선택합니다.
      iii. 이벤트 유형에 대해 사용할 수 있는 데이터 품질 평가 결과를 선택합니다.
      iv. 특정 상태에 대해 실패를 선택합니다. 그러면 다음과 비슷한 이벤트 패턴이 나타납니다.

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "state": ["FAILED"]
  }
}
```

      v. 더 많은 구성 옵션은 이벤트 패턴의 추가 구성 옵션 섹션을 참조하세요.
6. 대상 선택에서:
   a. 대상 유형에 대해 AWS 서비스를 선택합니다.
   b. 대상 선택 드롭다운을 사용하여 연결하려는 AWS 서비스(SNS, Lambda, SQS 등)를 선택하고 다음을 선택합니다.
7. 태그 구성에서 새 태그 추가를 클릭하여 선택적 태그를 추가하고 다음을 선택합니다.
8. 모든 선택 항목의 요약 페이지가 표시됩니다. 하단에서 규칙 생성을 선택합니다.

이벤트 패턴의 추가 구성 옵션

성공 또는 실패 시 이벤트 필터링 외에도 다양한 파라미터에서 이벤트를 추가로 필터링할 수 있습니다.

이렇게 하려면 이벤트 패턴 섹션으로 이동한 다음 패턴 편집을 선택하여 추가 파라미터를 지정합니다. 이벤트 패턴의 필드는 대소문자를 구분합니다. 다음은 이벤트 패턴 구성에 대한 예제입니다.

특정 규칙 세트를 평가하는 특정 테이블의 이벤트를 캡처하려면 다음 유형의 패턴을 사용합니다.

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "context": {
      "contextType": ["GLUE_DATA_CATALOG"],
      "databaseName": "db-123",
       "tableName": "table-123",
    },
    "rulesetNames": ["ruleset1", "ruleset2"]
    "state": ["FAILED"]
  }
}
```

ETL 환경에서 특정 작업의 이벤트를 캡처하려면 다음 유형의 패턴을 사용합니다.

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "context": {
      "contextType": ["GLUE_JOB"],
      "jobName": ["dq_evaluation_job1", "dq_evaluation_job2"]
    },
    "state": ["FAILED"]
  }
}
```

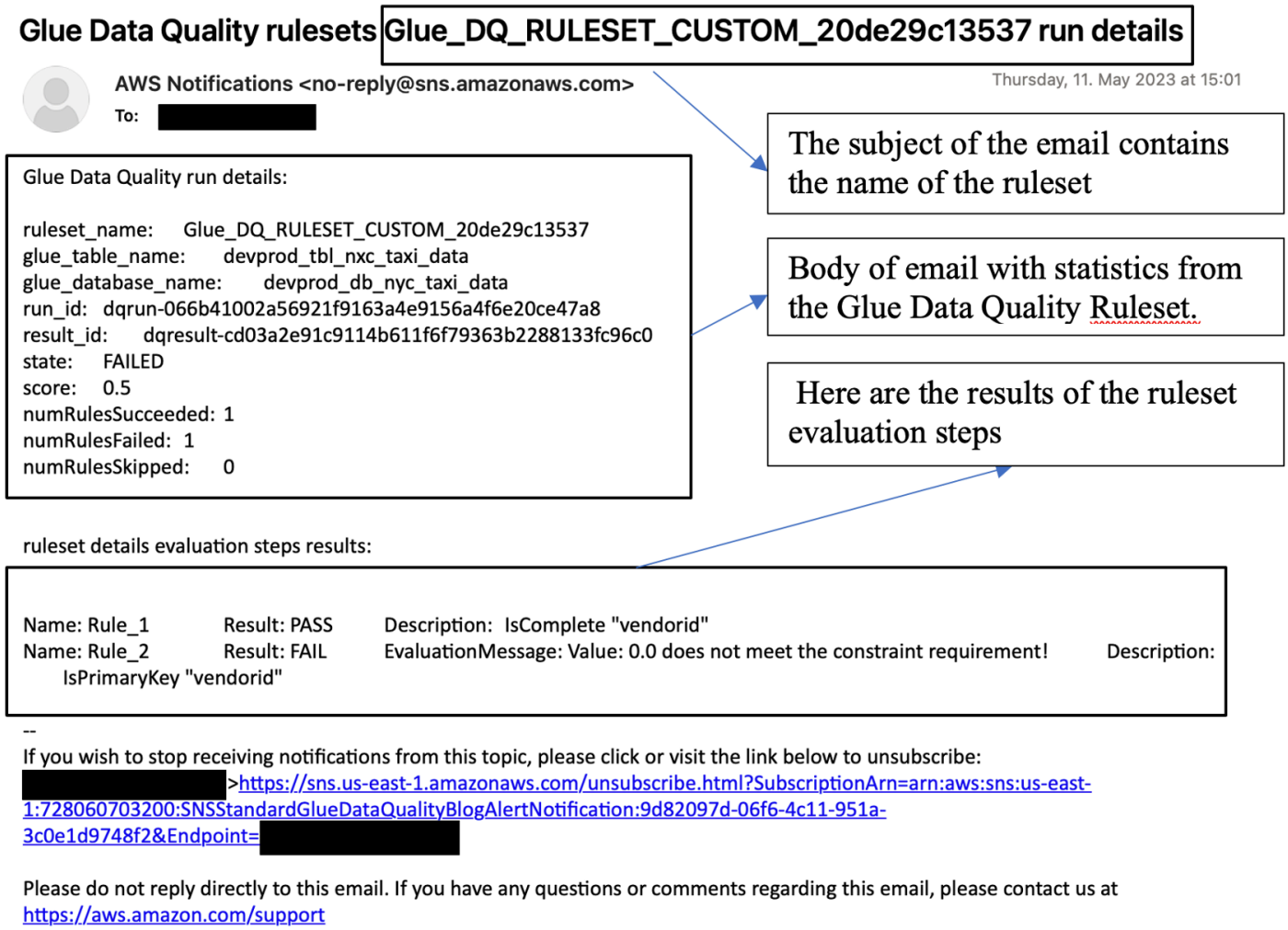
점수가 특정 임계값(예: 70%) 미만인 이벤트를 캡처하려면:

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
```

```
    "score": [{
      "numeric": ["<=", 0.7]
    }]
  }
}
```

알림을 이메일 형식으로 지정

비즈니스 팀에 올바른 형식의 이메일 알림을 보내야 하는 경우가 있습니다. Amazon EventBridge와 AWS Lambda를 사용하면 됩니다.



다음 샘플 코드를 사용하여 데이터 품질 알림 형식을 지정하여 이메일을 생성할 수 있습니다.

```
import boto3
import json
from datetime import datetime
```

```
sns_client = boto3.client('sns')
glue_client = boto3.client('glue')

sns_topic_arn = 'arn:aws:sns:<region-code>:<account-id>:<sns-topic-name>'



def lambda_handler(event, context):
    log_metadata = {}
    message_text = ""
    subject_text = ""

    if event['detail']['context']['contextType'] == 'GLUE_DATA_CATALOG':
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['tableName'] = str(event['detail']['context']['tableName'])
        log_metadata['databaseName'] = str(event['detail']['context']['databaseName'])
        log_metadata['runId'] = str(event['detail']['context']['runId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])
        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_table_name: {}\n".format(log_metadata['tableName'])
        message_text += "glue_database_name: {}\n".format(log_metadata['databaseName'])
        message_text += "run_id: {}\n".format(log_metadata['runId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
 {}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
 details".format(log_metadata['ruleset_name'])

    else:
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['jobName'] = str(event['detail']['context']['jobName'])
```

```
        log_metadata['jobId'] = str(event['detail']['context']['jobId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])

        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_job_name: {}\n".format(log_metadata['jobName'])
        message_text += "job_id: {}\n".format(log_metadata['jobId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    resultID = str(event['detail']['resultId'])
    response = glue_client.get_data_quality_result(ResultId=resultID)
    RuleResults = response['RuleResults']
    message_text += "\n\nruleset details evaluation steps results:\n\n"
    subresult_info = []

    for dic in RuleResults:
        subresult = "Name: {}\t\tResult: {}\t\tDescription: \t{}".format(dic['Name'],
dic['Result'], dic['Description'])
        if 'EvaluationMessage' in dic:
            subresult += "\t\tEvaluationMessage: {}".format(dic['EvaluationMessage'])
        subresult_info.append({
            'Name': dic['Name'],
            'Result': dic['Result'],
            'Description': dic['Description'],
            'EvaluationMessage': dic.get('EvaluationMessage', '')
        })
        message_text += "\n" + subresult

    log_metadata['resultrun'] = subresult_info
```

```
    sns_client.publish(
        TopicArn=sns_topic_arn,
        Message=message_text,
        Subject=subject_text
    )

    return {
        'statusCode': 200,
        'body': json.dumps('Message published to SNS topic')
    }
```

CloudWatch 통합에서 경보 및 알림 설정

Amazon EventBridge를 사용하여 데이터 품질 알림을 설정하는 것이 좋습니다. Amazon EventBridge는 고객에게 알림을 보낼 때 일회성 설정을 요구하기 때문입니다. 하지만 익숙하기 때문에 Amazon CloudWatch를 선호하는 고객도 있습니다. 이러한 고객을 위해 Amazon CloudWatch와의 통합을 제공합니다.

각 AWS Glue Data Quality 평가에서는 데이터 품질 실행당 `glue.data.quality.rules.passed`(통과한 규칙의 수를 나타냄) 및 `glue.data.quality.rules.failed`(실패한 규칙의 수를 나타냄)와 같은 지표 페어를 생성합니다. 이 생성된 지표를 사용하여 해당 데이터 품질 실행이 임계값 아래로 떨어질 경우 사용자에게 알리는 경보를 생성할 수 있습니다. Amazon SNS 알림을 통해 이메일을 보내는 경보 설정을 시작하려면 아래 단계를 수행합니다.

Amazon SNS 알림을 통해 이메일을 보내는 경보 설정을 시작하려면 아래 단계를 수행합니다.

1. Amazon CloudWatch 콘솔을 엽니다.
2. 지표 아래에서 모든 지표를 선택합니다. Glue Data Quality라는 사용자 지정 네임스페이스 아래에 추가 네임스페이스가 표시됩니다.

> **Note**
>
> AWS Glue 데이터 품질 실행을 시작할 때 Amazon CloudWatch에 지표 게시 확인란이 활성화되어 있는지 확인합니다. 그렇지 않으면 특정 실행에 대한 지표가 Amazon CloudWatch에 게시되지 않습니다.

Glue Data Quality 네임스페이스 아래에서 테이블 및 규칙 세트별로 지표가 생성되었음을 알 수 있습니다. 이 주제의 목적에 따라 이 값이 1을 초과하는 경우 glue.data.quality.rules.failed 규칙 및 경보를 사용합니다(즉, 실패한 규칙 평가 수가 1보다 크면 알림을 제공하려고 함).

3. 경보를 생성하려면 경보 아래에서 모든 경보를 선택합니다.
4. 경보 생성을 선택하세요.
5. 지표 선택을 선택하세요.
6. 생성한 테이블에 해당하는 glue.data.quality.rules.failed 지표를 선택한 다음 지표 선택을 선택합니다.
7. 지표 및 조건 지정 탭의 지표 섹션 아래에서:
   a. Statistic(통계)에서 Sum(합계)를 선택합니다.
   b. 기간에서 1분을 선택합니다.
8. 조건 섹션에서:
   a. 임계값 유형(Threshold type)에서 정적(Static)을 선택합니다.
   b. glue.data.quality.rules.failed 조건이 다음과 같을 때마다...에서 이상을 선택합니다.
   c. 기준...에서 임계값으로 1을 입력합니다.

   이렇게 선택하면 glue.data.quality.rules.failed 지표가 1 이상의 값을 생성하는 경우 경보가 트리거됩니다. 하지만 데이터가 없으면 허용 가능한 값으로 간주합니다.
9. 다음을 선택합니다.
10. 작업 구성에서:
    a. 경보 상태 트리거에서 경보 내를 선택합니다.
    b. 다음 SNS 주제로 알림 보내기 섹션에서 새 주제를 생성하여 새 SNS 주제를 통해 알림 보내기를 선택합니다.
    c. 알림을 수신할 이메일 엔드포인트에 이메일 주소를 입력합니다. 그리고 주제 생성을 클릭합니다.
    d. Next(다음)를 선택합니다.
11. 경보 이름에 myFirstDQAlarm을 입력한 후 다음을 선택합니다.
12. 모든 선택 항목의 요약 페이지가 표시됩니다. 하단에서 경보 생성을 선택합니다.

이제 Amazon CloudWatch 경보 대시보드에서 경보가 생성되는 것을 볼 수 있습니다.

데이터 품질 결과를 쿼리하여 대시보드 구축

대시보드를 구축하여 데이터 품질 결과를 표시할 수 있습니다. 이렇게 하는 방법은 두 가지입니다.

다음 코드를 사용하여 Amazon S3에 데이터를 쓰도록 Amazon EventBridge 설정:

```
import boto3
import json
from datetime import datetime


s3_client = boto3.client('s3')
glue_client = boto3.client('glue')


s3_bucket = 's3-bucket-name'

def write_logs(log_metadata):
    try:
        filename = datetime.now().strftime("%m%d%Y%H%M%S") + ".json"
        key_opts = {
            'year': datetime.now().year,
            'month': "{:02d}".format(datetime.now().month),
            'day': "{:02d}".format(datetime.now().day),
            'filename': filename
        }
        s3key = "gluedataqualitylogs/year={year}/month={month}/day={day}/
{filename}".format(**key_opts)
        s3_client.put_object(Bucket=s3_bucket, Key=s3key,
 Body=json.dumps(log_metadata))
    except Exception as e:
        print(f'Error writing logs to S3: {e}')


def lambda_handler(event, context):
    log_metadata = {}
    message_text = ""
    subject_text = ""

    if event['detail']['context']['contextType'] == 'GLUE_DATA_CATALOG':
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['tableName'] = str(event['detail']['context']['tableName'])
        log_metadata['databaseName'] = str(event['detail']['context']['databaseName'])
```

```
        log_metadata['runId'] = str(event['detail']['context']['runId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])
        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_table_name: {}\n".format(log_metadata['tableName'])
        message_text += "glue_database_name: {}\n".format(log_metadata['databaseName'])
        message_text += "run_id: {}\n".format(log_metadata['runId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    else:
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['jobName'] = str(event['detail']['context']['jobName'])
        log_metadata['jobId'] = str(event['detail']['context']['jobId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])

        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_job_name: {}\n".format(log_metadata['jobName'])
        message_text += "job_id: {}\n".format(log_metadata['jobId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
```

```
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    resultID = str(event['detail']['resultId'])
    response = glue_client.get_data_quality_result(ResultId=resultID)
    RuleResults = response['RuleResults']
    message_text += "\n\nruleset details evaluation steps results:\n\n"
    subresult_info = []

    for dic in RuleResults:
        subresult = "Name: {}\t\tResult: {}\t\tDescription: \t{}".format(dic['Name'],
dic['Result'], dic['Description'])
        if 'EvaluationMessage' in dic:
            subresult += "\t\tEvaluationMessage: {}".format(dic['EvaluationMessage'])
        subresult_info.append({
            'Name': dic['Name'],
            'Result': dic['Result'],
            'Description': dic['Description'],
            'EvaluationMessage': dic.get('EvaluationMessage', '')
        })
        message_text += "\n" + subresult

    log_metadata['resultrun'] = subresult_info

    write_logs(log_metadata)


    return {
        'statusCode': 200,
        'body': json.dumps('Message published to SNS topic')
    }
```

Amazon S3에 데이터를 쓴 후 AWS Glue 크롤러를 사용하여 Athena에 등록하고 테이블을 쿼리할 수 있습니다.

데이터 품질 평가 중에 Amazon S3 위치 구성:

AWS Glue 데이터 카탈로그 또는 AWS Glue ETL에서 데이터 품질 작업을 실행할 때 Amazon S3 위치를 제공하여 데이터 품질 결과를 Amazon S3에 쓸 수 있습니다. 아래 구문을 통해 데이터 품질 결과를 읽기 위해 대상을 참조하여 테이블을 생성할 수 있습니다.

`CREATE EXTERNAL TABLE` 및 `MSCK REPAIR TABLE` 쿼리는 별도로 실행해야 합니다.

```
CREATE EXTERNAL TABLE <my_table_name>(
    catalogid string,
    databasename string,
    tablename string,
    dqrunid string,
    evaluationstartedon timestamp,
    evaluationcompletedon timestamp,
    rule string,
    outcome string,
    failurereason string,
    evaluatedmetrics string)
PARTITIONED BY (
    `year` string,
    `month` string,
    `day` string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (

 'paths'='catalogId,databaseName,dqRunId,evaluatedMetrics,evaluationCompletedOn,evaluationStart
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://glue-s3-dq-bucket-us-east-2-results/'
TBLPROPERTIES (
    'classification'='json',
    'compressionType'='none',
    'typeOfData'='file');
```

```
MSCK REPAIR TABLE <my_table_name>;
```

위 테이블을 생성한 후에 Amazon Athena를 사용하여 분석 쿼리를 실행할 수 있습니다.

AWS CloudFormation을 사용하여 데이터 품질 규칙 배포

AWS CloudFormation을 사용하여 데이터 품질 규칙을 생성할 수 있습니다. 자세한 내용은 AWS Glue에 대한 AWS CloudFormation을 참조하세요.

데이터 품질 규칙 예약

다음 방법을 사용하여 데이터 품질 규칙을 예약할 수 있습니다.

- 데이터 카탈로그에서 데이터 품질 규칙을 예약합니다. 코드를 사용하지 않는 사용자도 이 옵션을 사용하여 데이터 품질 검사를 쉽게 예약할 수 있습니다. AWS Glue Data Quality는 Amazon EventBridge에서 예약을 생성합니다. 데이터 품질 규칙을 예약하려면:
  - 규칙 세트로 이동한 다음 실행을 클릭합니다.
  - 실행 빈도에서 원하는 예약을 선택하고 작업 이름을 제공합니다. 이 작업 이름은 EventBridge에서의 예약 이름입니다.
- Amazon EventBridge 및 AWS Step Functions를 사용하여 데이터 품질 규칙에 대한 평가 및 권장 사항을 조정합니다.

AWS Glue Data Quality의 저장 데이터 암호화

AWS Glue Data Quality는 기본적으로 암호화를 제공하여 민감한 고객 저장 데이터를 AWS 자체 암호화 키를 사용하여 보호합니다.

AWS 소유 키

AWS Glue Data Quality는 이러한 키를 사용하여 고객의 데이터 품질 자산을 자동으로 암호화합니다. 사용자는 AWS 소유 키를 확인, 관리 또는 사용하거나 사용을 감사할 수 없습니다. 하지만 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다. 자세한 내용은 AWS KMS 개발자 가이드에서 AWS 소유 키를 참조하세요.

기본적으로 저장 데이터를 암호화하면 민감한 데이터 보호와 관련된 운영 오버헤드와 복잡성을 줄이는 데 도움이 됩니다. 동시에 엄격한 암호화 규정 준수 및 규제 요구 사항을 충족하는 안전한 애플리케이션을 구축할 수 있습니다.

이 암호화 계층을 비활성화하거나 다른 암호화 유형을 선택할 수는 없지만 Data Quality 리소스를 생성할 때 고객 관리형 키를 선택하여 기존 AWS 소유 암호화 키에 두 번째 암호화 계층을 추가할 수 있습니다.

고객 관리형 키

고객 관리형 키: AWS Glue Data Quality는 사용자가 생성하고 소유하고 관리하는 대칭형 고객 관리형 키를 사용하도록 지원합니다. 따라서 기존 AWS 소유 암호화에 두 번째 암호화 계층이 추가됩니다. 이 암호화 계층을 완전히 제어할 수 있으므로 다음과 같은 작업을 수행할 수 있습니다.

- 키 정책 수립 및 유지
- IAM 정책 수립 및 유지
- 키 정책 활성화 및 비활성화
- 키 암호화 자료 교체
- 태그 추가
- 키 별칭 만들기
- 삭제를 위한 스케줄 키

자세한 내용은 AWS KMS Developer Guide의 Customer managed keys를 참조하십시오.

다음 표에는 AWS Glue Data Quality가 다양한 데이터 품질 자산을 암호화하는 방법이 요약되어 있습니다.

데이터 유형	AWS 소유 키 암호화	고객 관리형 키 암호화
데이터 품질 규칙 세트 영구 DQ 규칙 세트에서 참조하는 DQDL 규칙 세트 문자열입니다. 이러한 영구 규칙 세트는 현재 AWS Glue Data Catalog 환경에서만 사용됩니다.	활성화됨	활성화됨
데이터 품질 규칙/분석기 결과 규칙 세트에 있는 각 규칙의 통과/실패 상태와 규칙과 분석기 모두에서 수집한 지표를 포함하는 결과 아티팩트입니다.	활성화됨	활성화됨
관찰 데이터에서 이상이 탐지되면 관찰 항목이 생성됩니다. 여기에는 예상 상한과 하한에 대한 정보와 이러한 한도를 기반으로 제안되는 규칙이 포함됩니	활성화됨	활성화됨

데이터 유형	AWS 소유 키 암호화	고객 관리형 키 암호화
다. 생성된 경우 Data Quality 결과와 함께 표시됩니다.		
통계 지표 값(예: RowCount, Completeness), 열 이름, 기타 메타데이터 등 지정된 규칙 세트로 데이터를 평가한 후 수집된 지표에 대한 정보를 포함합니다.	활성화됨	활성화됨
이상 탐지 통계 모델 통계 모델에는 고객 데이터에 대한 이전 평가를 기반으로 생성된 특정 지표에 대한 상한과 하한의 시계열이 포함됩니다.	활성화됨	활성화됨

> **Note**
>
> AWS Data Quality는 AWS 소유 키를 사용하여 저장된 데이터를 자동으로 암호화하여 개인 식별 데이터를 무료로 보호할 수 있습니다. 그러나 고객 관리형 키 사용에는 AWS KMS 비용이 부과됩니다. 요금에 대한 자세한 내용은 AWS KMS 요금 부분을 참조하세요.
> AWS KMS에 대한 자세한 내용은 AWS KMS 섹션을 참조하세요.

고객 관리형 키 생성

AWS Management Console 또는 AWS KMS API를 사용하여 대칭형 고객 관리형 키를 만들 수 있습니다.

대칭형 고객 관리형 키를 생성하려면

- 자세한 내용은 AWS Key Management Service 개발자 가이드의 대칭 암호화 AWS KMS 키 생성의 단계를 따르세요.

키 정책

키 정책에서는 고객 관리형 키에 대한 액세스를 제어합니다. 모든 고객 관리형 키에는 키를 사용할 수 있는 사람과 키를 사용하는 방법을 결정하는 문장이 포함된 정확히 하나의 키 정책이 있어야 합니다. 고객 관리형 키를 만들 때 키 정책을 지정할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 가이드에서 AWS KMS 키의 키 정책을 참조하세요.

Data Quality 리소스에서 고객 관리형 키를 사용하려면 키 정책에서 다음 API 작업을 허용해야 합니다.

- kms:Decrypt - GenerateDataKeyWithoutPlaintext를 사용하여 AWS KMS 키로 암호화된 사이퍼텍스트를 해독합니다.
- kms:DescribeKey - Amazon Location에서 키의 유효성을 확인할 수 있도록 고객 관리형 키 세부 정보를 제공합니다.
- kms:GenerateDataKeyWithoutPlaintext - AWS KMS 외부에서 사용할 고유한 대칭 데이터 키를 반환합니다. 이 작업은 지정한 대칭 암호화 KMS 키로 암호화된 데이터 키를 반환합니다. 키의 바이트 수는 무작위이며 호출자 또는 KMS 키와는 무관합니다. 고객이 실행해야 하는 KMS 호출을 줄이는 데 사용됩니다.
- kms:ReEncrypt* - 사이퍼텍스트를 해독한 다음 AWS KMS 내에서 완전히 다시 암호화합니다. 이 작업을 사용하여 데이터가 암호화되는 KMS 키를 변경할 수 있습니다. 예를 들어 KMS 키를 수동으로 교체하거나 암호문을 보호하는 KMS 키를 변경할 수 있습니다. 또한 사이퍼텍스트의 암호화 컨텍스트를 변경하는 등 동일한 KMS 키로 암호문을 다시 암호화하는 데에도 사용할 수 있습니다.

다음은 Amazon Location에 추가할 수 있는 정책 설명 예시입니다.

```
"Statement" : [
    {
        "Sid" : "Allow access to principals authorized to use AWS Glue Data Quality",
        "Effect" : "Allow",
        "Principal" : {
            "AWS": "arn:aws:iam::<account_id>:role/ExampleRole"
        },
        "Action" : [
            "kms:Decrypt",
            "kms:DescribeKey",
            "kms:GenerateDataKeyWithoutPlaintext",
            "kms:ReEncrypt*"
        ],
        "Resource" : "*",
        "Condition" : {
```

```
            "StringEquals" : {
                "kms:ViaService" : "glue.amazonaws.com",
                "kms:CallerAccount" : "111122223333"
            }
        },
    {
        "Sid": "Allow access for key administrators",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
          },
        "Action" : [
            "kms:*"
         ],
        "Resource": "arn:aws:kms:region:111122223333:key/key_ID"
    },
    {
        "Sid" : "Allow read-only access to key metadata to the account",
        "Effect" : "Allow",
        "Principal" : {
            "AWS" : "arn:aws:iam::111122223333:root"
        },
        "Action" : [
            "kms:Describe*",
            "kms:Get*",
            "kms:List*",
        ],
        "Resource" : "*"
    }
]
```

AWS Glue Data Quality의 KMS 키 사용에 관한 참고 사항

AWS Glue Data Quality는 키 전환을 지원하지 않습니다. 즉, A 키로 데이터 품질 자산을 암호화하고 B 키로 전환하면, B 키를 사용하기 위해 A 키로 암호화된 데이터를 다시 암호화하지 않습니다. 어쨌든 B 키로 전환할 수는 있지만 이전에 A 키로 암호화된 데이터에 액세스하려면 A 키에 대한 액세스 권한을 유지해야 합니다.

정책의 권한 지정에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드에서 키 정책의 AWS 서비스에 대한 권한을 참조하세요.

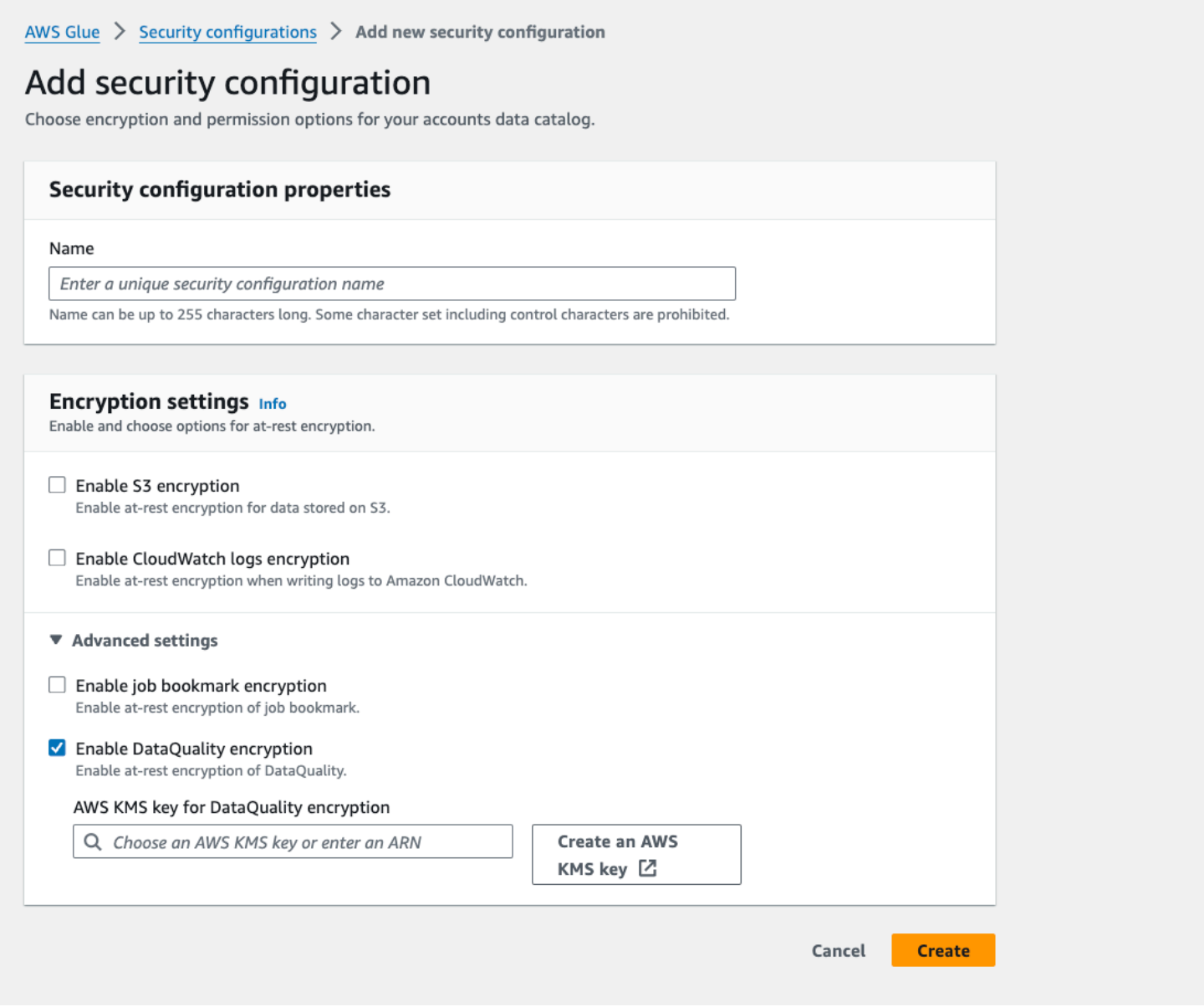
키 액세스 관련 문제 해결에 대한 내용은 AWS Key Management Service 개발자 가이드에서 키 액세스 문제 해결을 참조하세요.

보안 구성 생성

AWS Glue의 보안 구성 리소스에는 암호화된 데이터를 쓸 때 필요한 속성이 포함되어 있습니다.

데이터 품질 자산을 암호화하려면:

1. 암호화 설정의 고급 설정에서 데이터 품질 암호화 활성화를 선택합니다.
2. KMS 키를 선택하거나 AWS KMS 키 생성을 선택합니다.



AWS Glue Data Quality 암호화 컨텍스트

암호화 컨텍스트는 데이터에 대한 추가 컨텍스트 정보를 포함하는 선택적 키-값 페어 세트입니다.

AWS KMS는 암호화 컨텍스트를 추가 인증 데이터로 사용하여 인증된 암호화를 지원합니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하는 경우, AWS KMS는 암호화된 데이터에 암호화 컨텍스트를 바인딩합니다. 데이터 복호화를 위해, 이 요청에 동일한 암호화 컨텍스트를 포함합니다.

AWS Glue Data Quality 암호화 컨텍스트 예

```
"encryptionContext": {
    "kms-arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    "branch-key-id": "111122223333+arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    "hierarchy-version": "1",
    "aws-crypto-ec:aws:glue:securityConfiguration": "111122223333:customer-security-
configuration-name",
    "create-time": "2024-06-07T13:47:23:000861Z",
    "tablename": "AwsGlueMlEncryptionKeyStore",
    "type": "beacon:ACTIVE"
}
```

모니터링을 위한 암호화 컨텍스트 사용

대칭형 고객 관리 키를 사용하여 트래커 또는 지오펜스 컬렉션을 암호화하는 경우 감사 기록 및 로그의 암호화 컨텍스트를 사용하여 고객 관리 키가 사용되는 방식을 식별할 수도 있습니다. 암호화 컨텍스트는 AWS CloudTrail 또는 Amazon CloudWatch Logs에서 생성되는 로그에도 나타납니다.

AWS Glue Data Quality의 암호화 키 모니터링

AWS KMS 고객 관리형 키를 AWS Glue Data Quality 리소스에 사용하는 경우 AWS CloudTrail 또는 Amazon CloudWatch Logs를 사용하여 AWS Glue Data Quality가 AWS KMS에 보내는 요청을 추적할 수 있습니다.

다음은 AWS Glue Data Quality에서 고객 관리 키로 암호화된 데이터에 액세스하기 위해 직접적으로 호출한 KMS 운영을 모니터링하기 위한 `GenerateDataKeyWithoutPlainText` 및 `Decrypt`의 AWS CloudTrail 이벤트입니다.

Decrypt

```
{
```

```
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "arn": "arn:aws:sts::111122223333:role/CustomerRole",
        "accountId": "111122223333",
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-07-02T20:03:10Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "kms-arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "branch-key-id": "111122223333+arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "hierarchy-version": "1",
            "aws-crypto-ec:aws:glue:securityConfiguration": "111122223333:customer-
security-configuration-name",
            "create-time": "2024-06-07T13:47:23:000861Z",
            "tablename": "AwsGlueMlEncryptionKeyStore",
            "type": "branch:ACTIVE",
            "version": "branch:version:ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE"
        }
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKeyWithoutPlaintext

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "arn": "arn:aws:sts::111122223333:role/CustomerRole",
        "accountId": "111122223333",
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-07-02T20:03:10Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "kms-arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "branch-key-id": "111122223333+arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "hierarchy-version": "1",
            "aws-crypto-ec:aws:glue:securityConfiguration": "111122223333:customer-
security-configuration-name",
            "create-time": "2024-06-07T13:47:23:000861Z",
            "tablename": "AwsGlueMlEncryptionKeyStore",
            "type": "branch:version:ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE"
        }
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
```

```
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

ReEncyrpt

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "arn": "arn:aws:sts::111122223333:role/CustomerRole",
        "accountId": "111122223333",
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-07-17T21:34:41Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ReEncrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "destinationEncryptionContext": {
             "kms-arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "branch-key-id": "111122223333+arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "hierarchy-version": "1",
            "aws-crypto-ec:aws:glue:securityConfiguration": "111122223333:customer-
security-configuration-name",
            "create-time": "2024-06-07T13:47:23:000861Z",
            "tablename": "AwsGlueMlEncryptionKeyStore",
            "type": "branch:ACTIVE"
            "version": "branch:version:12345678-SAMPLE"
        },
```

```
        "destinationKeyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "sourceAAD": "1234567890-SAMPLE+Z+lqoYOHj7VtWxJLrvh+biUFbliYDAQkobM=",
        "sourceKeyId": "arn:aws:kms:ap-southeast-2:585824196334:key/17ca05ca-a8c1-40d7-
b7fd-30abb569a53a",
        "destinationEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "sourceEncryptionContext": {
            "kms-arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "branch-key-id": "111122223333+arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
            "hierarchy-version": "1",
            "aws-crypto-ec:aws:glue:securityConfiguration": "111122223333:customer-
security-configuration-name",
            "create-time": "2024-06-07T13:47:23:000861Z",
            "tablename": "AwsGlueMlEncryptionKeyStore",
            "type": "branch:version:ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE"
        },
        "destinationAAD": "1234567890-SAMPLE",
        "sourceEncryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
```

```
}
```

자세히 알아보기

다음 리소스에서 키에 대한 추가 정보를 확인할 수 있습니다.

- AWS Key Management Service 기본 개념에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드를 참조하세요.
- AWS Key Management Service의 보안 모범 사례에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드를 참조하세요.

AWS Glue Data Quality 오류 해결

AWS Glue Data Quality에서 오류가 발생하면 다음 해결 방법을 사용하여 문제의 원인을 찾아 수정할 수 있습니다.

목차

- 오류: 누락된 AWS Glue Data Quality 모듈
- 오류: AWS Lake Formation 권한 부족
- 오류: 규칙 세트의 이름이 고유하지 않음
- 오류: 특수 문자가 있는 테이블
- 오류: 큰 규칙 세트로 인한 오버플로 오류
- 오류: 전체 규칙 상태에서 실패임
- AnalysisException: 기본 데이터베이스의 존재를 확인할 수 없음
- 오류 메시지: Provided key map not suitable for given data frames
- 사용자 클래스에서 예외: java.lang.RuntimeException: 데이터를 가져오지 못했습니다. 자세한 내용은 CloudWatch의 로그를 확인하세요.
- 시작 오류: 버킷에 대해 S3에서 다운로드하는 중 오류 발생
- InvalidInputException(상태: 400): 데이터 품질 규칙을 구문 분석할 수 없음
- 오류: Eventbridge에서 사용자가 설정한 일정에 따라 Glue DQ 작업을 트리거하지 않습니다.
- CustomSQL 오류
- 동적 규칙
- 사용자 클래스의 예외: org.apache.spark.sql.AnalysisException: org.apache.hadoop.hive.ql.metadata.HiveException

- UNCLASSIFIED_ERROR; IllegalArgumentException: Parsing Error: No rules or analyzers provided., no viable alternative at input

오류: 누락된 AWS Glue Data Quality 모듈

오류 메시지: No module named 'awsgluedq'.

해결 방법: 이 오류는 지원되지 않는 버전에서 AWS Glue Data Quality을 실행할 때 발생합니다. AWS Glue Data Quality는 Glue 버전 3.0 이상에서만 지원됩니다.

오류: AWS Lake Formation 권한 부족

오류 메시지: Exception in User Class: `com.amazonaws.services.glue.model.AccessDeniedException`: Insufficient Lake Formation permission(s) on impact_sdg_involvement (Service: AWS Glue; Status Code: 400; Error Code: AccessDeniedException; Request ID: 465ae693-b7ba-4df0-a4e4-6b17xxxxxxxx; Proxy: null).

해결 방법: AWS Lake Formation에 충분한 권한을 제공해야 합니다.

오류: 규칙 세트의 이름이 고유하지 않음

오류 메시지: Exception in User Class: ...services.glue.model.AlreadyExistsException: Another ruleset with the same name already exists.

해결 방법: 규칙 세트는 글로벌 항목이며 고유해야 합니다.

오류: 특수 문자가 있는 테이블

오류 메시지: Exception in User Class: org.apache.spark.sql.AnalysisException: cannot resolve "C" given input columns: [primary.data_end_time, primary.data_start_time, primary.end_time, primary.last_updated, primary.message, primary.process_date, primary.rowhash, primary.run_by, primary.run_id, primary.start_time, primary.status]; line 1 pos 44;.

해결 방법: 현재 '.' 같은 특수 문자가 있는 테이블에서는 AWS Glue Data Quality를 실행할 수 없다는 제한 사항이 있습니다.

오류: 큰 규칙 세트로 인한 오버플로 오류

오류 메시지: Exception in User Class: java.lang.StackOverflowError.

해결 방법: 규칙 개수가 2,000개가 넘는 큰 규칙 세트가 있는 경우 이 문제가 발생할 수 있습니다. 규칙을 여러 규칙 세트로 구분합니다.

오류: 전체 규칙 상태에서 실패임

오류 조건: 내 규칙 세트는 성공했지만 전체 규칙 상태는 실패입니다.

해결 방법: 이 오류는 게시하는 동안 Amazon CloudWatch에 지표를 게시하는 옵션을 선택했기 때문에 발생했을 수 있습니다. 데이터 세트가 VPC에 있는 경우 VPC는 AWS Glue에서 Amazon CloudWatch에 지표를 게시하는 것을 허용하지 않을 수 있습니다. 이 경우 VPC에서 Amazon CloudWatch에 액세스할 수 있도록 엔드포인트를 설정해야 합니다.

AnalysisException: 기본 데이터베이스의 존재를 확인할 수 없음

오류 조건: AnalysisException: 기본 데이터베이스의 존재를 확인할 수 없음: com.amazonaws.services.glue.model.AccessDeniedException: 기본에서 Lake Formation 권한 부족 (서비스: AWS Glue, 상태 코드: 400, 오류 코드: AccessDeniedException, 요청 ID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX, 프록시: null)

해결 방법: AWS Glue 작업의 카탈로그 통합에서 AWS Glue는 항상 기본 데이터베이스의 존재 여부와 AWS Glue `GetDatabase API`의 사용 여부를 확인하려고 합니다. `DESCRIBE` Lake Formation 권한이 부여되지 않거나 `GetDatabase IAM` 권한이 부여되면 기본 데이터베이스의 존재를 확인할 때 작업에 실패합니다.

이 문제를 해결하려면:

1. Lake Formation에서 기본 데이터베이스에 대한 `DESCRIBE` 권한을 추가합니다.
2. Lake Formation에서 데이터베이스 생성자로 AWS Glue 작업에 연결된 IAM 역할을 구성합니다. 그러면 자동으로 기본 데이터베이스가 생성되고 역할에 필요한 Lake Formation 권한이 부여됩니다.
3. `--enable-data-catalog` 옵션을 비활성화합니다. (이는 AWS Glue Studio에서 Hive 메타스토어로 Data Catalog 사용으로 표시됩니다.)

   작업에 Spark SQL Data Catalog 통합이 필요하지 않은 경우 이를 비활성화할 수 있습니다.

오류 메시지: Provided key map not suitable for given data frames

오류 조건: 제공된 키 맵이 해당 데이터 프레임에 적합하지 않습니다.

해결 방법: DataSetMatch 규칙 유형을 사용하고 있으며 조인 키가 중복되었습니다. 조인 키는 고유해야 하며 NULL이 아니어야 합니다. 고유한 조인 키를 사용할 수 없는 경우에는 AggregateMatch와 같은 다른 규칙 유형을 사용하여 요약 데이터와 일치시키는 것이 좋습니다.

사용자 클래스에서 예외: java.lang.RuntimeException: 데이터를 가져오지 못했습니다. 자세한 내용은 CloudWatch의 로그를 확인하세요.

오류 조건: 사용자 클래스에서 예외: java.lang.RuntimeException: 데이터를 가져오지 못했습니다. 자세한 내용은 CloudWatch의 로그를 확인하세요.

해결 방법: 이 문제는 Amazon S3 기반 테이블에서 Amazon RDS 또는 Amazon Redshift와 비교되는 DQ 규칙을 생성할 때 발생합니다. 이 경우에는 AWS Glue에서 연결을 로드할 수 없습니다. 대신 Amazon Redshift 또는 Amazon RDS 데이터 세트에 DQ 규칙을 설정하세요. 이는 알려진 버그입니다.

시작 오류: 버킷에 대해 S3에서 다운로드하는 중 오류 발생

오류 조건: 시작 오류: 버킷에 대해 S3에서 다운로드하는 중 오류 발생: `aws-glue-ml-data-quality-assets-us-east-1, key: jars/aws-glue-ml-data-quality-etl.jar.Access Denied (Service: Amazon S3; Status Code: 403; Please refer logs for details)` .

해결 방법: AWS Glue Data Quality에 전달된 역할의 권한에서 이전 Amazon S3 위치에서의 읽기를 허용해야 합니다. 이 IAM 정책은 다음 역할에 연결되어야 합니다.

```
{
  "Sid": "allowS3",
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::aws-glue-ml-data-quality-assets-<region>/*"
}
```

자세한 권한은 데이터 품질 권한 부여를 참조하세요. 이 라이브러리는 데이터 세트의 데이터 품질을 평가하는 데 필요합니다.

InvalidInputException(상태: 400): 데이터 품질 규칙을 구문 분석할 수 없음

오류 조건: InvalidInputException(상태: 400): 데이터 품질 규칙을 구문 분석할 수 없습니다.

해결 방법: 이 오류에는 여러 가지 가능성이 있습니다. 한 가지 가능성으로, 규칙에 작은따옴표가 포함되었을 수 있습니다. 큰따옴표로 묶여 있는지 확인합니다. 예:

```
Rules = [
ColumnValues "tipo_vinculo" in ["CODO", "DOCO", "COCO", "DODO"] AND "categoria" = 'ES'
    AND "cod_bandera" = 'CEP'
```

이를 다음과 같이 변경합니다.

```
Rules = [
(ColumnValues "tipovinculo" in [ "CODO", "DOCO", "COCO", "DODO"]) AND (ColumnValues
 "categoria" = "ES")
    AND (ColumnValues "codbandera" = "CEP")
]
```

오류: Eventbridge에서 사용자가 설정한 일정에 따라 Glue DQ 작업을 트리거하지 않습니다.

오류 조건: Eventbridge에서 사용자가 설정한 일정에 따라 AWS Glue Data Quality 작업을 트리거하지 않습니다.

해결 방법: 작업을 트리거하는 역할에 적절한 권한이 없을 수 있습니다. 작업을 시작하는 데 사용하는 역할에 평가 실행 예약에 필요한 IAM 설정에서 언급한 권한이 있는지 확인합니다.

CustomSQL 오류

오류 조건: `The output from CustomSQL must contain at least one column that matches the input dataset for AWS Glue Data Quality to provide row level results. The SQL query is a valid query but no columns from the SQL result are present in the Input Dataset. Ensure that matching columns are returned from the SQL.`

해결 방법: SQL 쿼리가 유효한 경우 기본 테이블에서 열만 선택했는지 확인합니다. 기본 테이블의 열에서 sum, count 등의 집계 함수를 선택하면 이 오류가 발생할 수 있습니다.

오류 조건: `There was a problem when executing your SQL statement: cannot resolve "Col".`

해결 방법: 이 열은 기본 테이블에 없습니다.

오류 조건: `The columns that are returned from the SQL statement should only belong to the primary table. "In this case, some columns ( Col ) belong to reference table".`

해결 방법: SQL 쿼리에서 기본 테이블을 다른 참조 테이블과 조인할 때 기본 테이블에 대해 행 수준 결과를 생성하려면 select 문에 기본 테이블의 열 이름만 있는지 확인합니다.

동적 규칙

오류 조건: `Dynamic rules require job context, and cannot be evaluated in interactive session or data preview..`

원인: 해당 오류 메시지는 규칙 세트에 동적 DQ 규칙이 있는 경우 데이터 미리 보기 결과 또는 다른 대화형 세션에 나타날 수 있습니다. 동적 규칙은 특정 작업 이름 및 평가 컨텍스트와 관련된 기록 지표를 참조하므로 대화형 세션에서는 평가할 수 없습니다.

해결 방법: AWS Glue 작업을 실행하면 이전 지표가 생성되며 이후 동일한 작업에 대한 작업 실행 시 참조할 수 있습니다.

오류 조건:

- `[RuleType] rule only supports simple atomic operands in thresholds..`
- `Function last not yet implemented for [RuleType] rule.`

해결 방법: 동적 규칙은 일반적으로 숫자 표현식의 모든 DQDL 규칙 유형에 대해 지원됩니다(데이터 품질 정의 언어(DQDL) 참조 참조). 그러나 여러 지표를 생성하는 일부 규칙인 ColumnValues 및 ColumnLength는 아직 지원되지 않습니다.

오류 조건: `Binary expression operands must resolve to a single number..`

원인: 동적 규칙은 `RowCount > avg(last(5)) * 0.9`와 같은 이진 표현식을 지원합니다. 여기서 이진 표현식은 `avg(last(5)) * 0.9`입니다. 이 규칙은 피연산자 `avg(last(5))` 및 `0.9`가 모두 단일 숫자로 해석되므로 유효합니다. `RowCount > last(5) * 0.9`는 잘못된 예입니다. `last(5)`는 현재 행 수와 유의미하게 비교할 수 없는 목록을 생성하기 때문입니다.

해결 방법: 집계 함수를 사용하여 목록 값 피연산자를 단일 숫자로 줄입니다.

오류 조건:

- Rule threshold results in list, and a single value is expected. Use aggregation functions to produce a single value. Valid example: sum(last(10)), avg(last(10)).
- Rule threshold results in empty list, and a single value is expected.

원인: 동적 규칙을 사용하여 데이터 세트의 일부 기능을 기록 값과 비교할 수 있습니다. 마지막 함수를 사용하는 경우 양의 정수 인수가 제공되면 여러 기록 값을 검색할 수 있습니다. 예를 들어 last(5)는 규칙에 대한 작업 실행에서 관찰된 가장 최근의 값 5개를 검색합니다.

해결 방법: 현재 작업 실행에서 관찰된 값과 유의미한 비교를 수행하려면 집계 함수를 사용하여 이러한 값을 단일 숫자로 줄여야 합니다.

유효한 예:

- RowCount >= avg(last(5))
- RowCount > last(1)
- RowCount < last()

잘못된 예: RowCount > last(5).

오류 조건:

- Function index used in threshold requires positive integer argument.
- Index argument must be an integer. Valid syntax example: RowCount > index(last(10, 2)), which means RowCount must be greater than third most recent execution from last 10 job runs.

해결 방법: 동적 규칙을 작성할 때 index 집계 함수를 사용하여 목록에서 하나의 기록 값을 선택할 수 있습니다. 예를 들어 RowCount > index(last(5), 1)은 현재 작업에서 관찰된 행 수가 해당 작업에서 관찰된 두 번째로 최근 행 수보다 확실하게 큰지 여부를 확인합니다. index는 0으로 인덱싱됩니다.

오류 조건: IllegalArgumentException: Parsing Error: Rule Type: DetectAnomalies is not valid.

해결 방법: 이상 탐지는 AWS Glue 4.0에서만 사용할 수 있습니다.

오류 조건: IllegalArgumentException: Parsing Error: Unexpected condition for rule of type ... no viable alternative at input ....

참고: `...`은 동적입니다. 예: `IllegalArgumentException: Parsing Error: Unexpected condition for rule of type RowCount with number return type, line 4:19 no viable alternative at input '>last'`.

해결 방법: 이상 탐지는 AWS Glue 4.0에서만 사용할 수 있습니다.

사용자 클래스의 예외: org.apache.spark.sql.AnalysisException: org.apache.hadoop.hive.ql.metadata.HiveException

오류 조건: `Exception in User Class: org.apache.spark.sql.AnalysisException: org.apache.hadoop.hive.ql.metadata.HiveException: Unable to fetch table mailpiece_submitted. StorageDescriptor#InputFormat cannot be null for table: mailpiece_submitted (Service: null; Status Code: 0; Error Code: null; Request ID: null; Proxy: null)`

원인: AWS Glue 데이터 카탈로그에서 Apache Iceberg를 사용 중이고 AWS Glue 데이터 카탈로그의 입력 형식 속성이 비어 있습니다.

해결 방법: 이 문제는 DQ 규칙에서 CustomSQL 레이블 유형을 사용할 때 발생합니다. 이 문제를 해결하는 한 가지 방법은 '기본'을 사용하거나 카탈로그 이름 `glue_catalog.`를 `<database>.<table> in Custom ruletype`으로 추가하는 것입니다.

UNCLASSIFIED_ERROR; IllegalArgumentException: Parsing Error: No rules or analyzers provided., no viable alternative at input

오류 조건: `UNCLASSIFIED_ERROR; IllegalArgumentException: Parsing Error: No rules or analyzers provided., no viable alternative at input`

해결 방법: DQDL은 구문 분석할 수 없습니다. 이러한 현상이 발생하는 몇 가지 경우가 있습니다. 복합 규칙을 사용하는 경우 괄호가 올바른지 확인합니다.

```
(RowCount >= avg(last(10)) * 0.6) and (RowCount <= avg(last(10)) * 1.4) instead of
      RowCount >= avg(last(10)) * 0.6 and RowCount <= avg(last(10)) * 1.4
```

AWS Glue의 Amazon Q 데이터 통합

AWS Glue의 Amazon Q 데이터 통합은 AWS Glue의 새로운 생성형 AI 기능으로, 데이터 엔지니어와 ETL 개발자가 자연어를 사용하여 데이터 통합 작업을 구축할 수 있도록 지원합니다. 엔지니어와 개발자는 Amazon Q에 작업 작성과 문제 해결을 요청하거나 AWS Glue 및 데이터 통합에 관해 질문할 수 있습니다.

Amazon Q란 무엇인가요?

> **Note**
>
> Amazon Bedrock 제공: AWS는 자동 침해 탐지 기능을 구현합니다. Amazon Q 데이터 통합은 Amazon Bedrock을 기반으로 구축되었으므로 사용자는 Amazon Bedrock에 구현된 제어 기능을 최대한 활용하여 안전, 보안, 인공 지능(AI)의 책임 있는 사용을 강화할 수 있습니다.

Amazon Q는 AWS 애플리케이션을 이해하고, 구축하고, 확장하고, 운영하는 데 도움을 줄 수 있는 생성형 인공 지능(AI) 기반 대화형 어시스턴트입니다. Amazon Q가 기반으로 하는 모델은 고품질 AWS 콘텐츠로 보강되어 보다 완전하고 실행 가능하며 참조할 수 있는 답변을 제공하여 AWS에서의 구축 속도를 높일 수 있습니다. 자세한 내용은 What is Amazon Q?를 참조하세요.

AWS Glue의 Amazon Q 데이터 통합이란 무엇인가요?

AWS Glue의 Amazon Q 데이터 통합에는 다음과 같은 기능이 포함되어 있습니다.

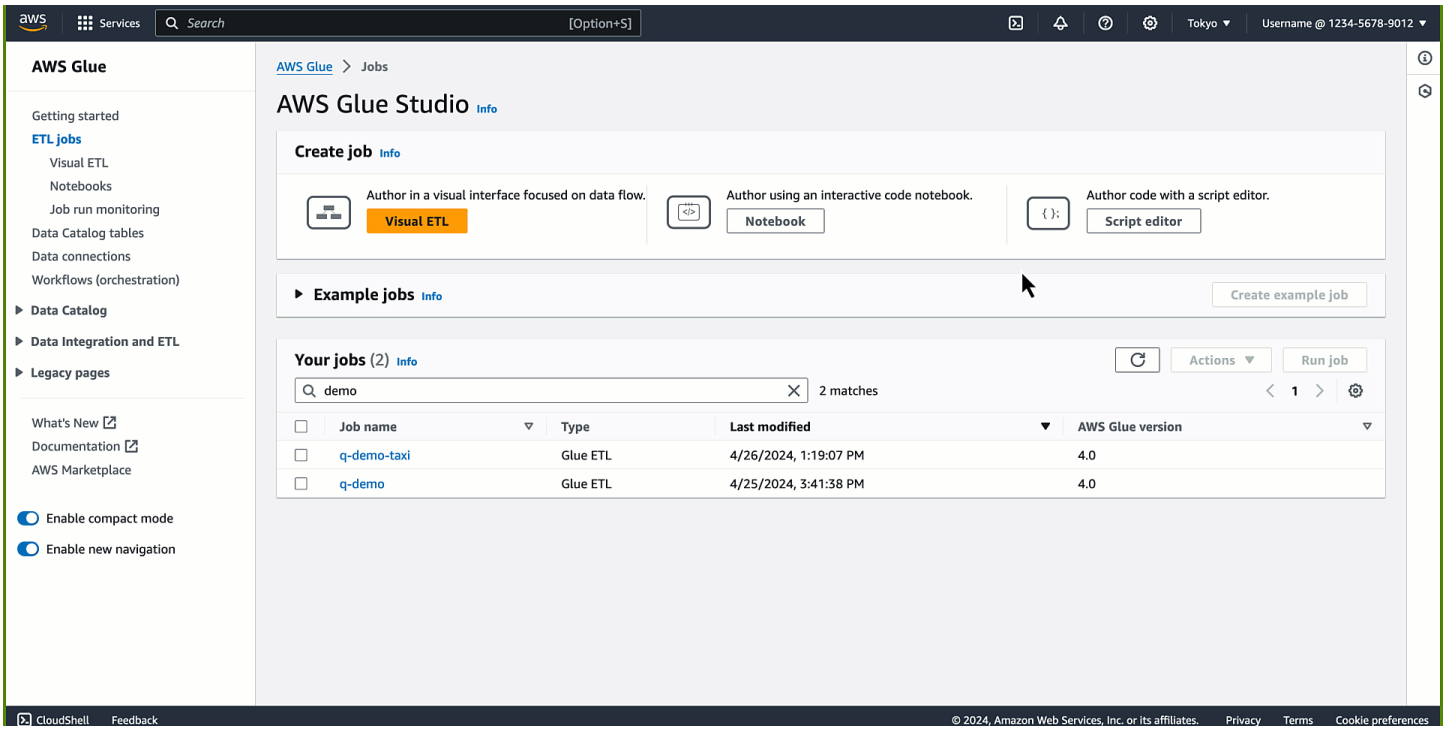
- 채팅 - AWS Glue의 Amazon Q 데이터 통합은 AWS Glue 및 AWS Glue 소스 및 대상 커넥터, AWS Glue ETL 작업, 데이터 카탈로그, 크롤러 및 AWS Lake Formation, 기타 기능 설명서 및 모범 사례와 같은 데이터 통합 도메인에 대한 자연어 질문에 영어로 답변할 수 있습니다. AWS Glue의 Amazon Q 데이터 통합은 단계별 지침과 함께 응답하며 정보 소스에 대한 참조가 포함되어 있습니다.
- 데이터 통합 코드 생성 - AWS Glue의 Amazon Q 데이터 통합은 AWS Glue ETL 스크립트에 대한 질문에 답변하고 영어로 된 자연어 질문이 주어지면 새로운 코드를 생성할 수 있습니다.
- 문제 해결 - AWS Glue의 Amazon Q 데이터 통합은 AWS Glue 작업의 오류를 이해하는 데 도움이 되도록 특별히 설계되었으며 문제의 근본 원인 및 해결을 위한 단계별 지침을 제공합니다.

> **Note**
>
> AWS Glue의 Amazon Q 데이터 통합은 대화가 진행되는 동안 향후 응답에 정보를 제공하기 위해 대화의 컨텍스트를 사용하지 않습니다. AWS Glue의 Amazon Q 데이터 통합에 관한 각 대화는 이전 대화 또는 향후 대화와 무관합니다.

AWS Glue의 Amazon Q 데이터 통합을 사용 중이신가요?

Amazon Q 패널에서 AWS Glue ETL 스크립트에 대한 Amazon Q 코드 생성을 요청하거나 AWS Glue 기능 또는 오류 해결에 대한 질문에 답변할 수 있습니다. 응답은 스크립트를 사용자 지정하고, 검토하고, 실행하는 단계별 지침이 포함된 PySpark의 ETL 스크립트입니다. 질문의 경우 데이터 통합 지식을 기반으로 답변이 생성되며, 답변에는 참조할 수 있는 요약 및 소스 URL이 포함되어 있습니다.

예를 들어 Amazon Q에 'Snowflake에서 읽고 필드 이름을 변경하며 Redshift에 쓰는 Glue 스크립트를 제공하세요'를 요청하고, 이에 대한 응답으로 AWS Glue의 Amazon Q 데이터 통합은 요청된 작업을 수행할 수 있는 AWS Glue 작업 스크립트를 반환합니다. 생성된 코드를 검토하여 요청된 의도를 충족하는지 확인할 수 있습니다. 만족스럽다면 프로덕션에서 AWS Glue 작업으로 배포할 수 있습니다. 통합 기능에 오류와 실패를 설명하고 해결책을 제안하도록 요청하여 작업 문제를 해결할 수 있습니다. Amazon Q는 AWS Glue 또는 데이터 통합 모범 사례에 대한 질문에 대한 답변을 제공합니다.



다음은 AWS Glue의 Amazon Q 데이터 통합이 AWS Glue를 구축하는 데 어떻게 도움이 되는지 보여주는 예제 질문입니다.

AWS Glue ETL 코드 생성:

- S3에서 JSON을 읽고, 매핑 적용을 사용하여 필드를 변환하고, Amazon Redshift에 쓰는 AWS Glue 스크립트를 작성합니다.
- DynamoDB에서 읽고, DropNullFields 변환을 적용하고, S3에 Parquet으로 쓰기 위한 AWS Glue 스크립트를 작성하려면 어떻게 해야 하나요?
- MySQL에서 읽고, 비즈니스 로직을 기반으로 일부 필드를 삭제하고, Snowflake에 쓰는 AWS Glue 스크립트를 알려주세요.
- DynamoDB에서 읽고 S3에 JSON으로 쓸 AWS Glue 작업을 작성합니다.
- S3에 대한 AWS Glue 데이터 카탈로그용 AWS Glue 스크립트를 개발하도록 도와주세요.
- S3에서 JSON을 읽고, null을 삭제하고, Redshift에 쓰는 AWS Glue 작업을 작성합니다.

AWS Glue 기능 설명:

- AWS Glue Data Quality는 어떻게 사용하나요?
- AWS Glue 작업 북마크는 어떻게 사용하나요?
- AWS Glue Auto Scaling을 어떻게 활성화하나요?
- AWS Glue 동적 프레임과 Spark 데이터 프레임의 차이점은 무엇인가요?
- AWS Glue에서 지원하는 다양한 연결 유형에는 어떤 것이 있나요?

AWS Glue 문제 해결:

- AWS Glue 작업 시 발생하는 메모리 부족(OOM) 오류를 해결하는 방법은 무엇인가요?
- AWS Glue Data Quality를 설정할 때 표시될 수 있는 오류 메시지에는 어떤 것이 있으며 어떻게 해결할 수 있나요?
- Amazon S3 액세스 거부 오류가 발생한 AWS Glue 작업은 어떻게 수정하나요?
- AWS Glue 작업의 데이터 셔플 문제를 해결하려면 어떻게 하나요?

Amazon Q 데이터 통합과의 상호 작용 모범 사례

다음은 Amazon Q 데이터 통합과의 상호 작용 모범 사례입니다.

- Amazon Q 데이터 통합과 상호 작용할 경우 구체적인 질문을 하고, 복잡한 요청이 있으면 반복하며, 답변이 정확한지 검증합니다.
- 자연어로 데이터 통합 프롬프트를 제공할 때는 도우미가 필요한 내용을 정확히 이해할 수 있도록 최대한 구체적으로 작성합니다. 'S3에서 데이터 추출'을 요청하는 대신 'S3에서 JSON 파일을 추출하는 AWS Glue 스크립트 작성'과 같은 자세한 내용을 제공합니다.
- 생성된 스크립트를 을 실행하기 전에 검토하여 정확성을 확인합니다. 생성된 스크립트에 오류가 있거나 의도와 맞지 않는 경우 도우미에 수정 방법에 대한 지침을 제공합니다.
- 생성형 AI 기술은 새로운 기술이며 응답에 할루시네이션이라고 하는 실수가 있을 수 있습니다. 환경 또는 워크로드에서 사용하기 전에 오류와 취약성이 있는지 모든 코드를 테스트하고 검토하세요.

AWS Glue 서비스 개선에서 Amazon Q 데이터 통합

AWS Glue의 Amazon Q 데이터 통합이 AWS 서비스에 대한 가장 적절한 정보를 제공하는 데 도움이 되도록, 서비스 개선을 위해 Amazon Q에 묻는 질문 및 해당 응답과 같이 Amazon Q에서 특정 콘텐츠를 사용할 수 있습니다.

사용하는 콘텐츠와 옵트아웃 방법에 대한 자세한 내용은 Amazon Q Developer 사용 설명서의 Amazon Q Developer 서비스 개선을 참조하세요.

고려 사항

AWS Glue의 Amazon Q 데이터 통합을 사용하기 전에 다음 항목을 고려하세요.

- 현재 코드 생성은 PySpark 커널에서만 작동합니다. 생성된 코드는 Python Spark 기반의 AWS Glue 작업용입니다.
- AWS Glue에서 Amazon Q 데이터 통합의 지원되는 코드 생성 기능 조합에 대한 자세한 내용은 지원되는 코드 생성 기능 섹션을 참조하세요.

AWS Glue의 Amazon Q 데이터 통합 설정

다음 섹션에서는 AWS Glue의 Amazon Q 데이터 통합을 설정하는 방법에 대해 설명합니다.

주제

- IAM 권한 구성

IAM 권한 구성

이 주제에서는 Amazon Q 채팅 환경 및 AWS Glue Studio 노트북 환경에 대해 구성하는 IAM 권한을 설명합니다.

주제

- Amazon Q 채팅에 대한 IAM 권한 구성
- AWS Glue Studio 노트북의 IAM 권한 구성

Amazon Q 채팅에 대한 IAM 권한 구성

AWS Glue에서 Amazon Q 데이터 통합에 사용되는 API에 권한을 부여하려면 적절한 AWS ID 및 액세스 관리(IAM) 권한이 필요합니다. 다음 사용자 지정 AWS 정책을 IAM ID(예: 사용자, 역할 또는 그룹)에 첨부하여 권한을 얻을 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartCompletion",
                "glue:GetCompletion"
            ],
            "Resource": [
                "arn:aws:glue:*:*:completion/*"
            ]
        }
    ]
}
```

AWS Glue Studio 노트북의 IAM 권한 구성

AWS Glue 스튜디오 노트북에서 Amazon Q 데이터 통합을 활성화하려면 노트북 IAM 역할에 다음 권한이 연결되어 있어야 합니다.

> **Note**
>
> `codewhisperer` 접두사는 Amazon Q Developer와 병합된 서비스의 기존 이름입니다. 자세한 내용은 Amazon Q Developer 이름 변경 - 변경 사항 요약을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartCompletion",
                "glue:GetCompletion"
            ],
            "Resource": [
                "arn:aws:glue:*:*:completion/*"
            ]
        },
        {
            "Sid": "AmazonQDeveloperPermissions",
            "Effect": "Allow",
            "Action": [
                "codewhisperer:GenerateRecommendations"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> AWS Glue의 Amazon Q 데이터 통합에는 프로그래밍 방식으로 사용할 수 있는 AWS SDK를 통해 사용할 수 있는 API가 없습니다. Amazon Q 채팅 패널 또는 AWS Glue Studio 노트북을 통해 이 환경을 활성화하기 위한 IAM 정책에는 `StartCompletion` 및 `GetCompletion`의 두 가지 API가 사용됩니다.

권한 할당

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- AWS IAM Identity Center의 사용자 및 그룹: 권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.
- ID 제공업체를 통해 IAM에서 관리되는 사용자: ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(페더레이션)의 지침을 따릅니다.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따릅니다.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르십시오.

지원되는 코드 생성 기능

다음은 Amazon Q 데이터 통합의 코드 생성 기능 조합입니다.

소스 및 대상	변환
다음과 같은 형식 유형을 사용하는 S3: json, csv, parquet, hudi, delta	Drop
AWS Glue Data Catalog	Aggregate
Redlake	DropDuplicates
Amazon DynamoDB	조인
MySQL	필터
Oracle	RenameColumns
PostgresSQL	FillNull
Microsoft SQL Server	DropNull
Amazon DocumentDB / MongoDB	WithColumns
Snowflake	SQL 쿼리
Google BigQuery	결합

소스 및 대상	변환
Teradata	Select
Amazon OpenSearch Service	
Vertica	
SAP HANA	
Amazon Redshift	

상호 작용 예제

AWS Glue의 Amazon Q 데이터 통합을 통해 Amazon Q 패널에 질문을 입력할 수 있습니다. AWS Glue에서 제공하는 데이터 통합 기능에 관한 질문을 입력할 수 있습니다. 참조 문서와 함께 자세한 답변이 반환됩니다.

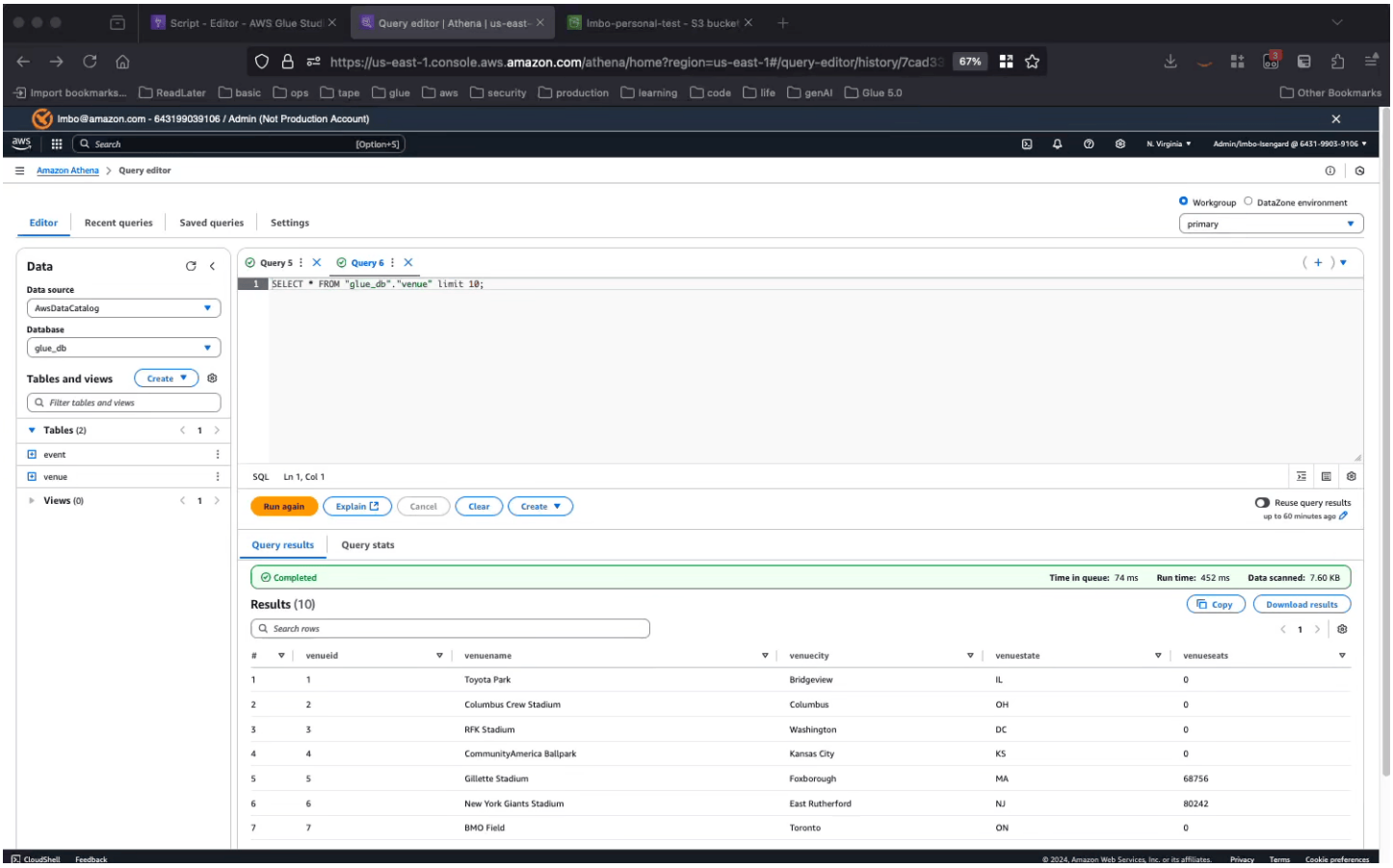
또 다른 사용 사례는 AWS Glue ETL 작업 스크립트를 생성하는 것입니다. 데이터 추출, 변환, 로드 작업을 수행하는 방법에 관해 질문할 수 있습니다. 생성된 PySpark 스크립트가 반환됩니다.

주제

- Amazon Q 채팅 상호 작용
- AWS Glue Studio 노트북 상호 작용

Amazon Q 채팅 상호 작용

AWS Glue 콘솔에서 새 작업 작성을 시작하고 Amazon Q에 "Create a Glue ETL flow connect to two Glue catalog tables venue and event in my database glue_db, join the results on the venue's venueid and event's e_venueid, and then filter on venue state with condition as venuestate=='DC' and write to s3://amzn-s3-demo-bucket/codegen/BDB-9999/output/ in CSV format."라고 요청합니다.



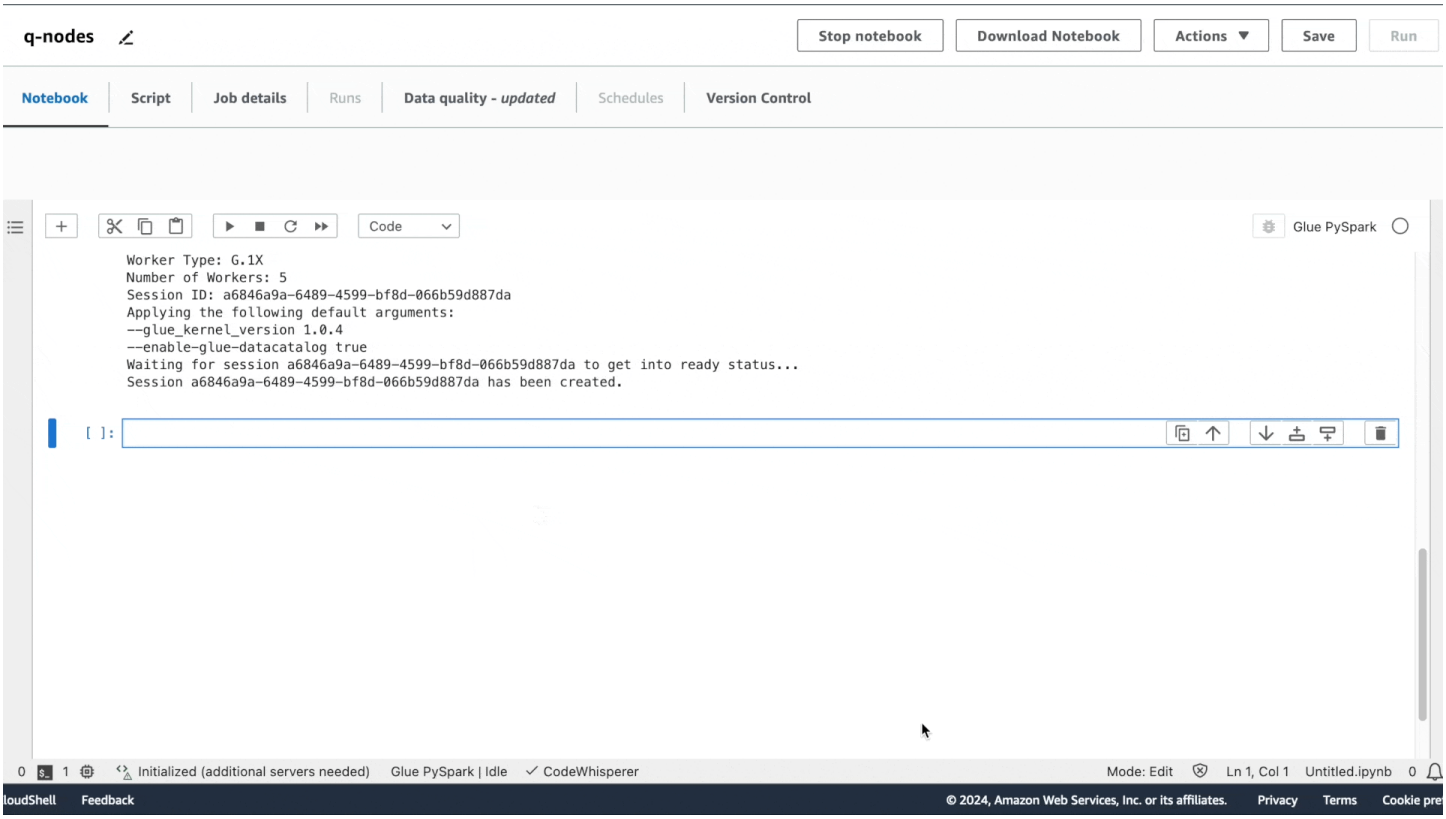
코드가 생성되는 것을 확인할 수 있습니다. 이 응답을 통해 목적에 맞게 AWS Glue 코드를 작성하는 방법을 배우고 이해할 수 있습니다. 생성된 코드를 복사하여 스크립트 편집기에 붙여넣고 자리 표시자를 구성할 수 있습니다. 작업에서 IAM 역할 및 AWS Glue 연결을 구성한 후 작업을 저장하고 실행합니다. 작업이 완료되면 요약 데이터가 예상대로 Amazon S3에 유지되고 다운스트림 워크로드에서 사용할 수 있는지 확인할 수 있습니다.

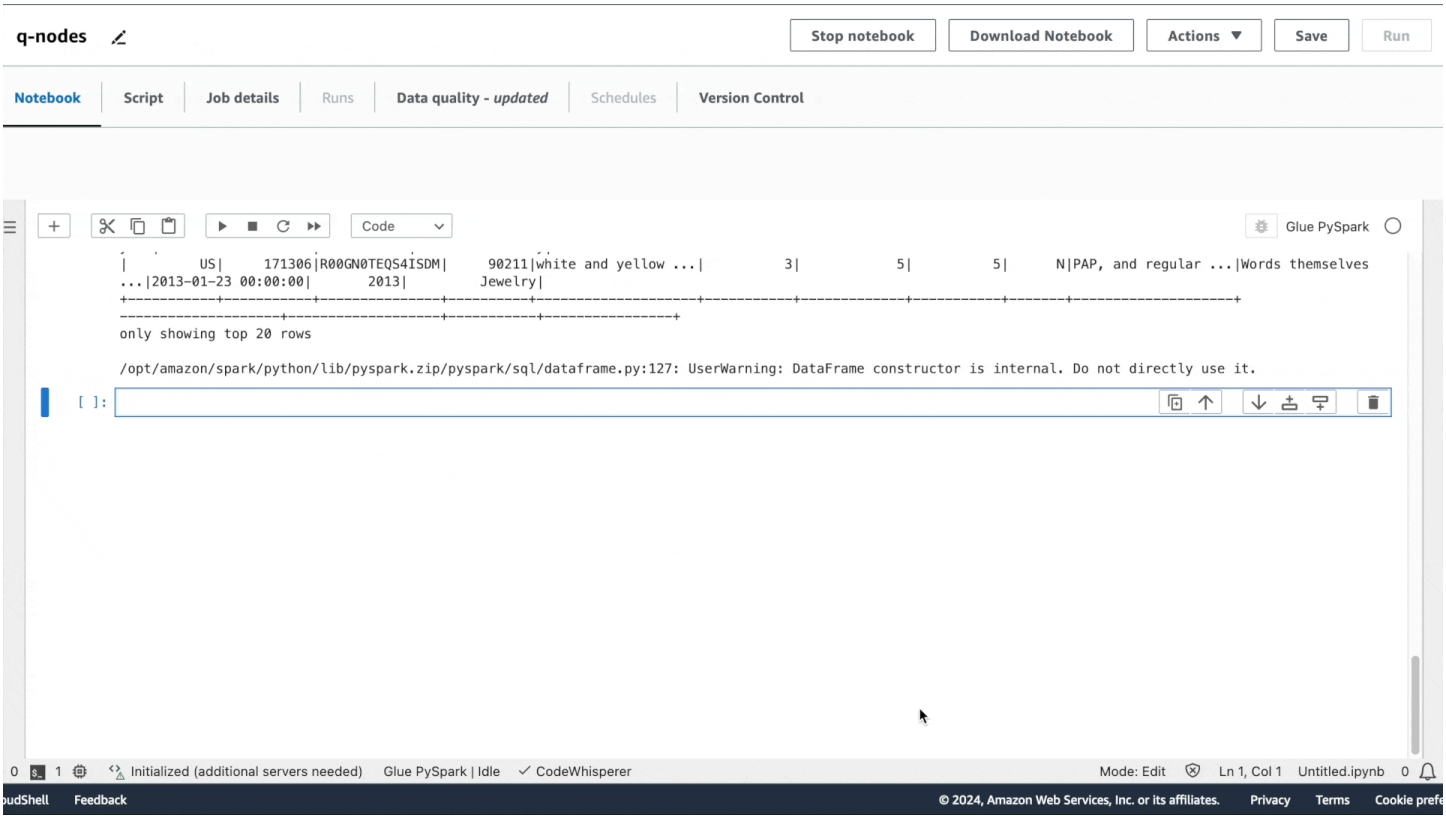
AWS Glue Studio 노트북 상호 작용

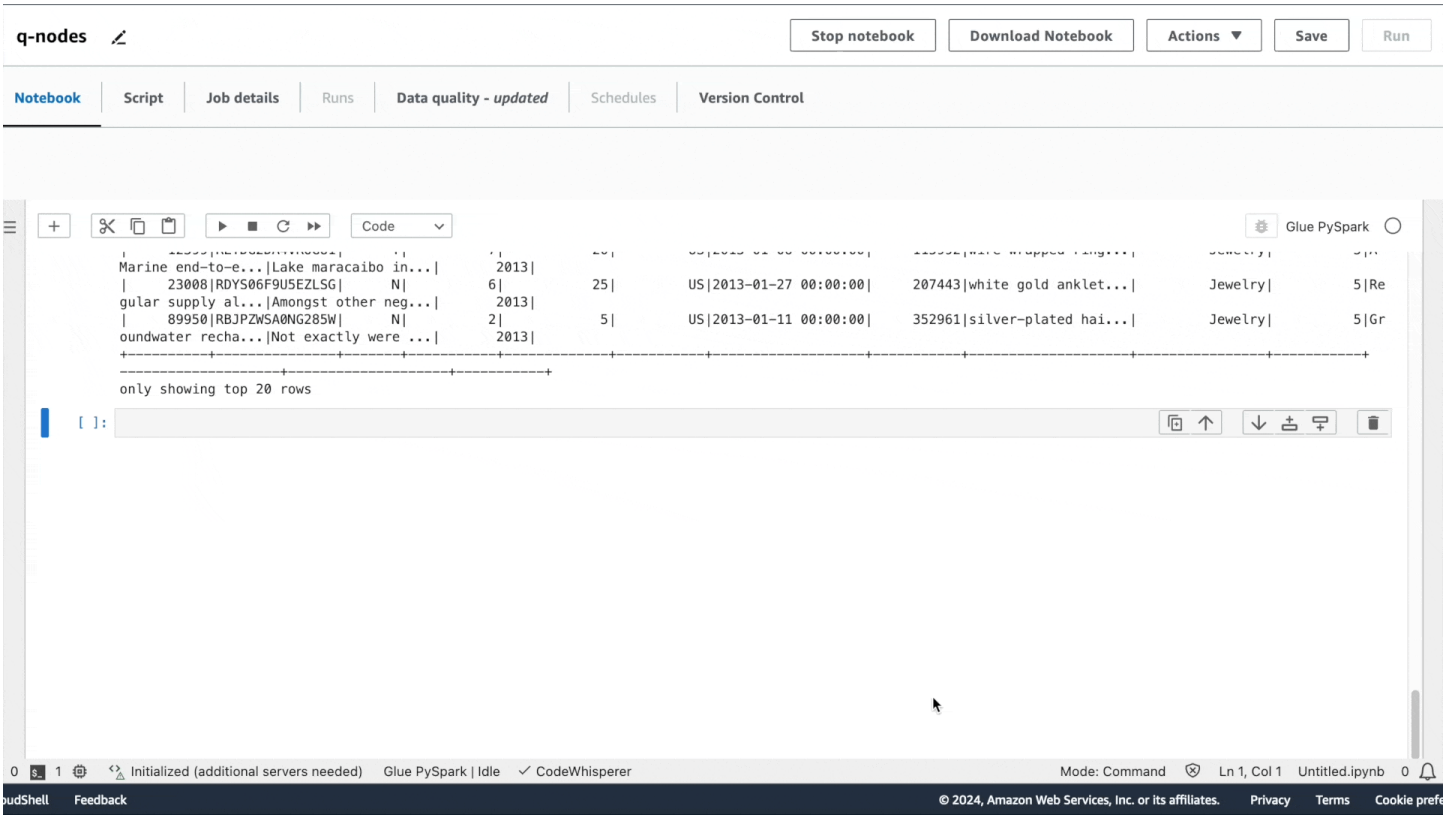
> **Note**
>
> AWS Glue Studio 노트북의 Amazon Q 데이터 통합 경험은 여전히 DynamicFrame 기반 데이터 통합 흐름에 중점을 둡니다.

새 셀을 추가하고 의견을 입력하여 원하는 목표를 설명합니다. Tab 및 Enter를 누르면 권장 코드가 표시됩니다.

첫 번째 의도는 데이터 추출('Glue Data Catalog 테이블을 읽는 코드 제공') 이후 'star_rating > 3의 필터 변환을 적용하는 코드 제공' 및 'S3에 Parquet으로 프레임을 쓰는 코드 제공'과 같습니다.







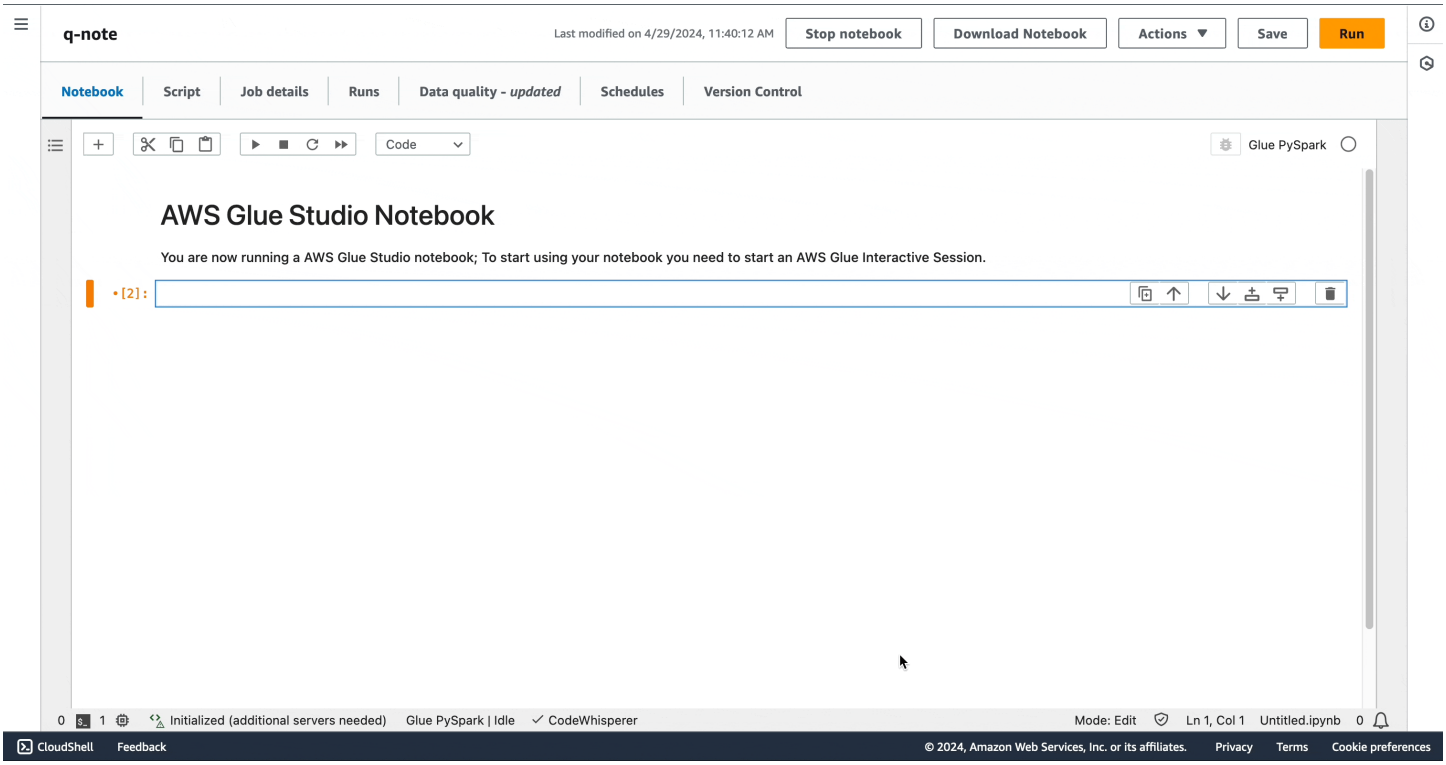
Amazon Q 채팅 경험과 마찬가지로 코드가 권장됩니다. 탭을 누르면 권장 코드가 선택됩니다.

생성된 코드에서 소스에 적합한 옵션을 입력하여 각 셀을 실행할 수 있습니다. 실행 중 언제든지 `show()` 메서드를 사용하여 데이터세트 샘플을 미리 볼 수도 있습니다.

실행을 선택하거나 프로그래밍 방식으로 노트북을 작업으로 실행할 수 있습니다.

복잡한 프롬프트

하나의 복잡한 프롬프트를 포함하는 전체 스크립트를 생성할 수 있습니다. '서로 결합해야 하는 S3의 JSON 데이터와 Oracle의 데이터가 있습니다. 두 소스 모두에서 읽고 조인한 다음, Redshift에 결과를 쓰는 Glue 스크립트를 제공하세요.'



노트북에서 AWS Glue의 Amazon Q 데이터 통합이 Amazon Q 채팅에서 생성된 것과 동일한 코드 조각을 생성했음을 알 수 있습니다.

실행을 선택하거나 프로그래밍 방식으로 노트북을 작업으로 실행할 수 있습니다.

AWS Glue에서 Amazon Q 데이터 통합과 함께 컨텍스트 인식 사용

이제 Amazon Q 데이터 통합에서 쿼리 기반 컨텍스트 인식 및 PySpark DataFrame 코드 생성을 사용하여 데이터 처리 작업을 더 효율적으로 생성할 수 있습니다. 예를 들어, "create a job to load sales data from Redshift table ‘analytics.salesorder’ using connection ‘erp_conn’, filter order_amount below 50 dollars, and save to Amazon S3 in parquet format." 프롬프트를 사용하여 PySpark 코드를 생성할 수 있습니다.

Amazon Q는 프롬프트를 기반으로 스크립트를 생성하고 연결 구성, 스키마 세부 정보, 데이터베이스/테이블 이름, 변환을 위한 열 사양과 같은 질문에서 제공된 세부 정보를 사용하여 데이터 통합 워크플로 설정을 설정합니다. 연결 옵션 암호와 같은 민감한 정보는 계속 수정됩니다.

프롬프트에서 필수 정보를 제공하지 않으면 Amazon Q는 자리 표시자를 배치하므로 코드를 실행하기 전에 생성된 코드를 적절한 값으로 업데이트해야 합니다.

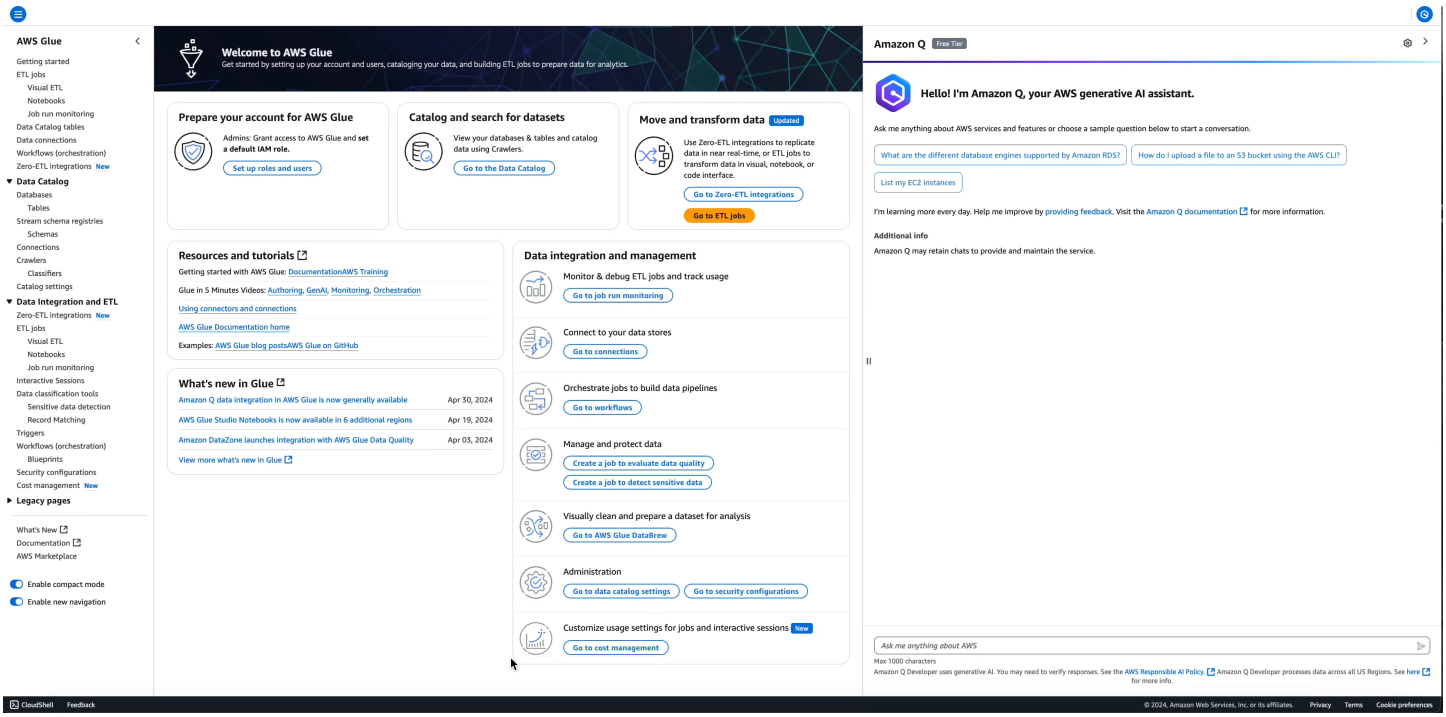
다음은 컨텍스트 인식을 사용하는 방법에 대한 예입니다.

예: 상호 작용

Amazon S3 파일 읽기 및 Amazon S3 버킷에 쓰기

프롬프트: Create an AWS Glue spark job that reads a file s3://amzn-s3-demo-bucket-input-bucket-name/input-filename.csv and saves it into the parquet file under directory s3://amzn-s3-demo-bucket-output-bucket-name/output-s3-prefix/

```
import sys
from awsglue.transforms import *
from pyspark.context import SparkContext
from pyspark.sql import SparkSession

sc = SparkContext.getOrCreate()
spark = SparkSession.builder.getOrCreate()

# Script generated for node S3DataSource
S3DataSource_dsource1 = spark.read.format("csv").load(
    "s3://amzn-s3-demo-bucket-input-bucket-name/input-filename.csv"
)

# Script generated for node S3DataSink
S3DataSource_dsource1.write.format("parquet").mode("append").save(
    "s3://amzn-s3-demo-bucket-output-bucket-name/output-s3-prefix/"
)
```



Lakehouse에서 데이터 가져오기 및 데이터베이스에 쓰기

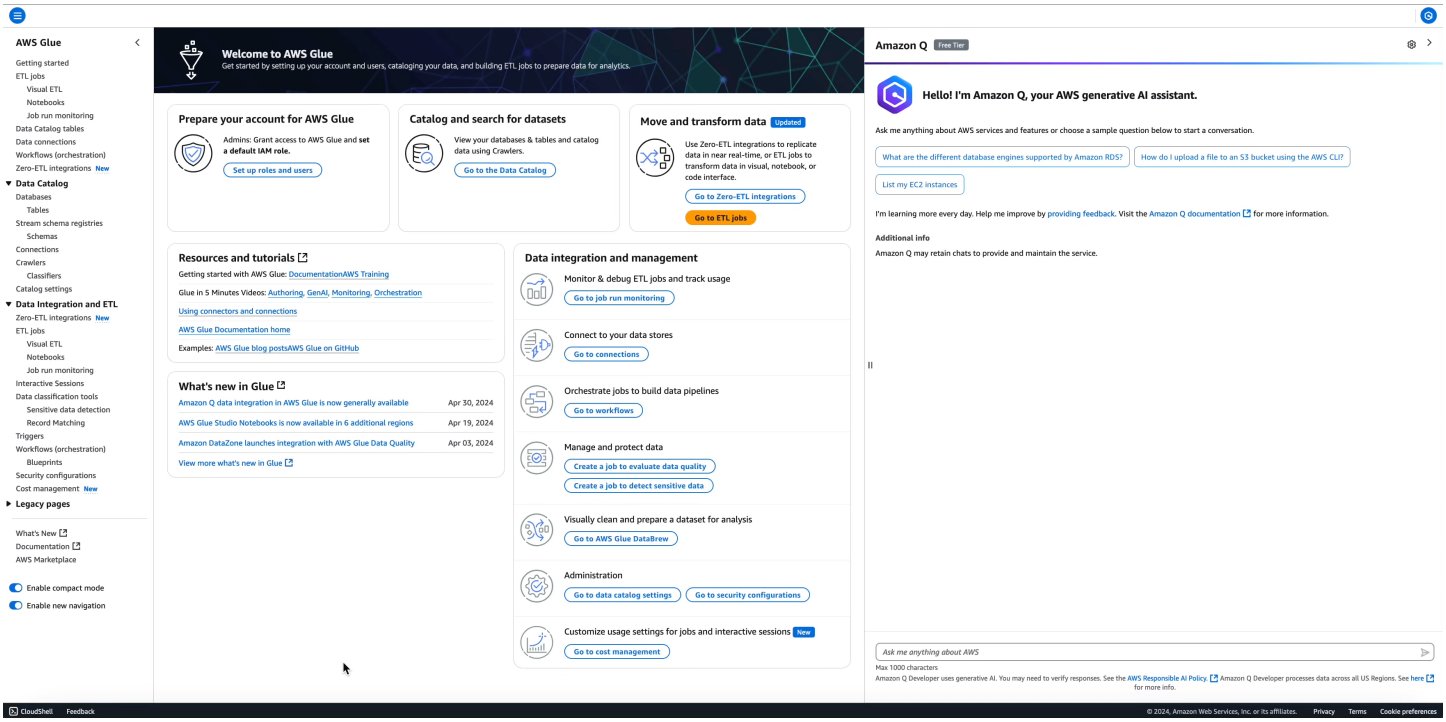
프롬프트: `write an ETL script to read from a Lakehouse table my-table in database my-database and write it to a RDS MySQL table my-target-table`

정보를 제공하지 않은 필드(예: 필요한 connectionName은 MySQL 데이터 싱크용이고 생성된 코드에서 기본값은 자리 표시자 <connection-name>)의 경우 스크립트를 실행하기 전에 필요한 정보를 입력할 수 있도록 자리 표시자가 유지됩니다.

생성된 스크립트:

```
import sys
from awsglue.transforms import *
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from connectivity.adapter import CatalogConnectionHelper

sc = SparkContext.getOrCreate()
spark = SparkSession.builder.getOrCreate()

# Script generated for node S3DataSource
S3DataSource_dsource1 = spark.read.format("parquet").load(
    "s3://amzn-lakehouse-demo-bucket/my-database/my-table"
)
```

```
# Script generated for node ConnectionV2DataSink
ConnectionV2DataSink_dsink1_additional_options = {"dbtable": "my-target-table"}
CatalogConnectionHelper(spark).write(
    S3DataSource_dsource1,
    "mysql",
    "<connection-name>",
    ConnectionV2DataSink_dsink1_additional_options,
)
```



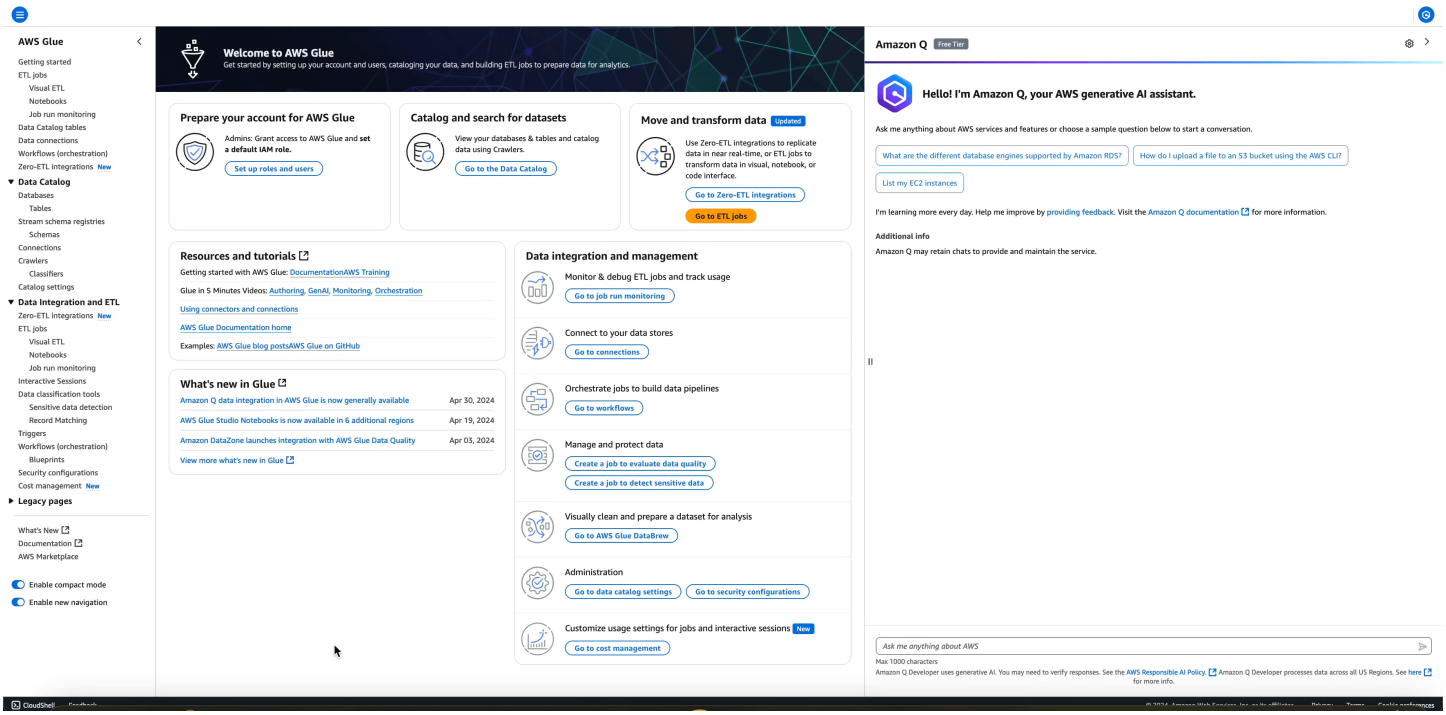
예: 전체 ETL 워크플로

다음 예제에서는 `Create a AWS Glue ETL Script read from two AWS Glue Data Catalog tables venue and event in my database glue_db_4fthqih3vvk1if, join the results on the field venueid, filter on venue state with condition as venuestate=='DC' after joining the results and write output to an Amazon S3 S3 location s3://amz-s3-demo-bucket/output/ in CSV format` 프롬프트를 사용하여 AWS Glue에 전체 ETL 워크플로를 완료하는 AWS Glue 스크립트를 생성하도록 요청하는 방법을 보여 줍니다.

워크플로에는 서로 다른 데이터 소스(두 AWS Glue Data Catalog 테이블)의 읽기, 읽기 후 두 읽기의 결과를 조인하는 변환 두 개, 특정 조건에 기반하는 필터링, 변환된 출력을 Amazon S3 대상에 CSV 형식으로 쓰기가 포함됩니다.

생성된 작업은 아래와 같이 데이터 소스, 변환 및 싱크 작업에 대한 세부 정보를 사용자 질문에서 추출한 해당되는 정보로 채웁니다.

```
import sys
from awsglue.transforms import *
from pyspark.context import SparkContext
from pyspark.sql import SparkSession

sc = SparkContext.getOrCreate()
spark = SparkSession.builder.getOrCreate()

# Script generated for node CatalogDataSource
CatalogDataSource_dsource1 = spark.sql("select * from
 `glue_db_4fthqih3vvk1if`.`venue`")

# Script generated for node CatalogDataSource
CatalogDataSource_dsource2 = spark.sql("select * from
 `glue_db_4fthqih3vvk1if`.`event`")

# Script generated for node JoinTransform
JoinTransform_transform1 = CatalogDataSource_dsource1.join(
    CatalogDataSource_dsource2,
    (CatalogDataSource_dsource1["venueid"] == CatalogDataSource_dsource2["venueid"]),
    "inner",
)

# Script generated for node FilterTransform
FilterTransform_transform2 = JoinTransform_transform1.filter("venuestate=='DC'")

# Script generated for node S3DataSink
FilterTransform_transform2.write.format("csv").mode("append").save(
    "s3://amz-s3-demo-bucket/output//output/"
)
```



제한 사항

- 컨텍스트 전달:
  - 컨텍스트 인식 기능은 동일한 대화 내에서 이전 사용자 쿼리의 컨텍스트만 전달합니다. 바로 이전 쿼리를 넘어서는 컨텍스트는 유지되지 않습니다.
- 노드 구성 지원:
  - 현재 컨텍스트 인식에서는 다양한 노드에 대한 필수 구성의 하위 세트만 지원합니다.
  - 선택적 필드에 대한 지원은 향후 릴리스에서 계획되어 있습니다.
- 가용성:
  - 컨텍스트 인식 및 DataFrame 지원은 Q Chat 및 SageMaker Unified Studio 노트북에서 사용할 수 있습니다. 그러나 이러한 기능은 아직 AWS Glue Studio 노트북에서 사용할 수 없습니다.

AWS Glue에서 오케스트레이션

다음 섹션에서는 AWS Glue에서 작업 오케스트레이션에 대한 정보를 제공합니다.

주제

- 트리거를 사용하여 작업 및 크롤러 시작
- AWS Glue에서 블루프린트 및 워크플로를 사용하여 복잡한 ETL 활동 수행
- AWS Glue에서 블루프린트 개발

트리거를 사용하여 작업 및 크롤러 시작

AWS Glue에서는 트리거라는 Data Catalog 객체를 생성할 수 있습니다. 이 객체를 사용하면 하나 이상의 크롤러 또는 추출, 변환, 로드 작업을 수동 또는 자동으로 시작할 수 있습니다. 트리거를 사용하여 종속 작업 및 크롤러 체인을 설계할 수 있습니다.

> **Note**
>
> 워크플로우를 정의하여 동일한 작업을 수행할 수 있습니다. 워크플로우는 복잡한 다중 작업 ETL 작업을 생성하는 데 권장됩니다. 자세한 내용은 the section called “블루프린트 및 워크플로를 사용하여 복잡한 ETL 활동 수행” 단원을 참조하십시오.

주제

- AWS Glue 트리거
- 트리거 추가
- 트리거 활성화 및 비활성화

AWS Glue 트리거

트리거가 실행되면 지정된 작업 및 크롤러를 시작할 수 있습니다. 트리거는 일정에 따라 또는 이벤트 조합을 기반으로 요청 시 실행됩니다.

> **Note**
>
> 단일 트리거로 2개의 크롤러만 활성화할 수 있습니다. 여러 개의 데이터 스토어를 크롤링하려면 여러 크롤러를 동시에 실행하지 말고 크롤러마다 각각 여러 개의 소스를 사용합니다.

트리거는 여러 상태 중 하나에 존재할 수 있습니다. 트리거는 `CREATED`, `ACTIVATED` 또는 `DEACTIVATED`입니다. 일시적 상태(예: `ACTIVATING`)도 있습니다. 트리거 실행을 일시적으로 중지하려면 비활성화할 수 있습니다. 그런 다음 나중에 다시 활성화할 수 있습니다.

트리거에는 다음 세 가지 유형이 있습니다.

예약됨

`cron` 기반의 시간 기반 트리거

일정에 따라 일련의 작업 또는 크롤러에 대한 트리거를 생성할 수 있습니다. 작업 또는 크롤러가 실행되는 빈도, 실행되는 요일 및 실행되는 시간과 같은 제약 조건을 지정할 수 있습니다. 이 제약 조건은 `cron`을 기반으로 합니다. 트리거를 위한 일정을 설정하고자 한다면 cron의 기능 및 제한을 고려해야 합니다. 예를 들어, 매월 31일에 크롤러를 실행하고자 한다면 매월 31일이 없다는 점을 유의하기 바랍니다. Cron에 대한 자세한 내용은 작업 및 크롤러를 위한 시간 기반 일정 섹션을 참조하세요.

조건

이전 작업이나 크롤러 또는 여러 작업이나 크롤러가 조건 목록을 만족할 때 실행되는 트리거입니다.

조건부 트리거를 생성할 때 감시할 작업 목록과 크롤러 목록을 지정합니다. 감시한 각 작업 또는 크롤러에 대해 감시할 상태(예: 성공, 실패, 시간 초과 등)를 지정합니다. 감시한 작업 또는 크롤러가 지정된 상태로 종료되면 트리거가 실행됩니다. 감시한 이벤트 중 일부 또는 전부가 발생할 때 트리거가 실행되도록 구성할 수 있습니다.

예를 들어 작업 J1과 작업 J2가 모두 성공적으로 완료되면 작업 J3을 시작하도록 트리거 T1을 구성하고 작업 J1 또는 작업 J2 중 하나가 실패할 경우 작업 J4를 시작하도록 또 다른 트리거 T2를 구성할 수 있습니다.

다음 표에는 트리거가 감시하는 작업 및 크롤러 완료 상태(이벤트)가 나와 있습니다.

작업 완료 상태	크롤러 완료 상태
• SUCCEEDED • STOPPED • FAILED • TIMEOUT	• SUCCEEDED • FAILED • CANCELLED

온디맨드

활성화할 때 실행되는 트리거입니다. 온디맨드 트리거는 ACTIVATED 또는 DEACTIVATED 상태로 전환되지 않습니다. 항상 CREATED 상태로 유지됩니다.

트리거가 존재하는 즉시 실행할 수 있도록 플래그를 설정하여 생성할 때 예약된 트리거와 조건부 트리거를 활성화할 수 있습니다.

> **Important**
>
> 다른 작업 또는 크롤러 완료 시 실행되는 작업 또는 크롤러를 종속이라고 합니다. 종속 작업 또는 크롤러는 완료된 작업 또는 크롤러가 트리거에 의해 시작된 경우에만 시작됩니다. 종속 체인 내의 모든 작업 또는 크롤러는 단일 예약됨 또는 온디맨드 트리거의 하위 요소여야 합니다.

트리거를 사용하여 작업 파라미터 전달

트리거는 시작하는 작업에 파라미터를 전달할 수 있습니다. 매개 변수에는 작업 인수, 시간 초과 값, 보안 구성 등이 포함됩니다. 트리거가 여러 작업을 시작하면 파라미터가 각 작업에 전달됩니다.

다음은 트리거에 의해 전달된 작업 인수에 대한 규칙입니다.

- 키 - 값 쌍의 키가 기본 작업 인수와 일치하면 전달된 인수는 기본 인수를 재정의합니다. 키가 기본 인수와 일치하지 않으면 인수는 추가 인수로 작업에 전달됩니다.
- 키 - 값 쌍의 키가 재정의할 수 없는 인수와 일치하면 전달된 인수는 무시됩니다.

자세한 내용은 AWS Glue API의 the section called “트리거” 단원을 참조하십시오.

트리거 추가

AWS Glue 콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS Glue API를 사용하여 트리거를 추가할 수 있습니다.

> **Note**
>
> 현재 AWS Glue 콘솔은 트리거를 사용하여 작업할 때 크롤러가 아닌 작업만 지원합니다. AWS CLI 또는 AWS Glue API를 사용하여 작업과 크롤러로 트리거를 구성할 수 있습니다.

트리거를 추가하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 ETL 아래에서 트리거를 선택합니다. 트리거 추가를 선택합니다.
3. 다음 속성을 제공합니다.

   명칭

   트리거에 고유한 이름을 지정합니다.

   트리거 유형

   다음 중 하나를 지정하세요.

   - 일정: 트리거는 특정 빈도와 시간에 실행됩니다.
   - 작업 이벤트: 조건부 트리거입니다. 목록의 작업 중 일부 또는 전부가 지정된 상태와 일치하면 트리거가 실행됩니다. 트리거가 실행되기 위해서는 감시한 작업이 트리거에 의해 시작되었어야 합니다. 어떤 작업을 선택하든지 하나의 작업 이벤트(완료 상태)만 감시할 수 있습니다.
   - 온디맨드: 트리거가 활성화되면 실행됩니다.
4. 트리거 마법사를 완료합니다. [검토(Review)] 페이지에서 [생성 시 트리거 사용(Enable trigger on creation)]을 선택하여 [일정(Schedule)] 및 [작업 이벤트(Job events)](조건부) 트리거를 즉시 활성화할 수 있습니다.

트리거를 추가하려면(AWS CLI)

- 다음과 유사한 명령을 입력합니다.

```
aws glue create-trigger --name MyTrigger --type SCHEDULED --schedule  "cron(0 12 *
 * ? *)" --actions CrawlerName=MyCrawler --start-on-creation
```

이 명령은 라는 MyTrigger라는 스케줄 트리거를 생성합니다. 이 트리거는 매일 오후 12시(UTC)에 실행되며 MyCrawler라는 크롤러를 시작합니다. 트리거는 활성화된 상태로 생성됩니다.

자세한 내용은 the section called “AWS Glue 트리거” 단원을 참조하십시오.

작업 및 크롤러를 위한 시간 기반 일정

AWS Glue에서 작업 및 크롤러를 위한 시간 기반 일정을 정의합니다. 일정 정의는 Unix식 cron 구문을 사용합니다. 협정시계시(UTC)의 시간을 지정하고 일정을 위한 최소한의 정확도는 5분입니다.

일정을 사용하여 실행되도록 작업 및 크롤러 구성에 대한 자세한 내용은 트리거를 사용하여 작업 및 크롤러 시작 섹션을 참조하세요.

cron 표현식

cron 표현식에는 각각 공백으로 구분되는 필수 필드 6개가 있습니다.

구문

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

필드	값	와일드카드
Minutes	0~59	, - * /
Hours	0~23	, - * /
Day-of-month	1~31	, - * ? / L W
월	1~12 또는 JAN-DEC	, - * /
요일	1~7 또는 SUN~SAT	, - * ? / L
연도	1970~2199	, - * /

와일드카드

- ,(쉼표) 와일드카드는 추가 값을 포함합니다. Month 필드에서 JAN,FEB,MAR는 1월, 2월, 3월을 포함한다는 의미입니다.
- -(대시) 와일드카드는 범위를 지정합니다. Day 필드에서 1~15는 지정된 달의 1일에서 15일까지 포함한다는 의미입니다.
- *(별표) 와일드카드는 필드의 모든 값을 포함합니다. Hours 필드에서 *는 모든 시간을 포함한다는 의미입니다.
- /(슬래시) 와일드카드로 증분을 지정합니다. Minutes 필드에서 **1/10**을 입력하면 지정한 시간의 1분부터 시작해서 매 10분 간격(예를 들어, 11분, 21분, 31분 등)을 지정할 수 있습니다.
- ?(물음표) 와일드카드는 어떤 한 가지나 다른 것을 지정합니다. Day-of-month 필드에 7을 입력하고 Day-of-week 필드에는 ?을 입력하면 매월 7일이 무슨 요일이든 상관없이 7번째 되는 날을 지정한다는 의미입니다.
- Day-of-month 또는 Day-of-week 필드에서 L 와일드카드는 해당 월 또는 주의 마지막 날을 지정합니다.
-] 필드에서는 W 와일드카드로 어떤 한 평일을 지정할 수 있습니다. Day-of-month Day-of-month 필드에서 3W를 해당 월의 세 번째 평일에 가장 가까운 날을 지정할 수 있습니다.

Limits

- 동일한 cron 표현식에 Day-of-month와 Day-of-week 필드를 지정할 수 없습니다. 이 필드 중 하나에 값을 지정하는 경우에는 다른 필드에서 반드시 ?(물음표)를 사용해야 합니다.
- 5분보다 빠른 속도로 이어지는 cron 식은 지원되지 않습니다.

예시

일정을 생성할 때는 다음과 같은 Cron 문자열을 사용할 수 있습니다.

분	시간	일	월	요일	연도	의미
0	10	*	*	?	*	매일 오전 10시(UTC)에 실행
15	12	*	*	?	*	매일 오후 12시 15분

분	시간	일	월	요일	연도	의미
						(UTC)에 실행
0	18	?	*	월-금	*	매주 월요일부터 금요일까지 오후 6시(UTC)에 실행
0	8	1	*	?	*	매월 1일 오전 8시(UTC)에 실행
0/15	*	*	*	?	*	15분마다 실행
0/10	*	?	*	월-금	*	월요일부터 금요일까지 10분마다 실행
0/5	8~17	?	*	월-금	*	월요일부터 금요일까지 오전 8시부터 오후 5시 55분(UTC) 사이에 5분마다 실행

예를 들어, 매일 12시 15분(UTC)에 실행하려면 다음과 같이 지정합니다.

```
cron(15 12 * * ? *)
```

트리거 활성화 및 비활성화

AWS Glue 콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS Glue API를 사용하여 트리거를 활성화하거나 비활성화할 수 있습니다.

트리거를 활성화 또는 비활성화하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 ETL 아래에서 트리거를 선택합니다.
3. 원하는 트리거 옆의 확인란을 선택하고 작업 메뉴에서 트리거 활성화를 선택하여 트리거를 활성화하거나 트리거 를 선택하여 트리거를 비활성화합니다.

트리거를 활성화 또는 비활성화하려면(AWS CLI)

- 다음 명령 중 하나를 입력합니다.

```
aws glue start-trigger --name MyTrigger

aws glue stop-trigger --name MyTrigger
```

  트리거를 시작하면 트리거가 활성화되고 트리거를 중지하면 트리거가 비활성화됩니다. 온디맨드 트리거를 활성화하면 즉시 실행됩니다.

자세한 내용은 the section called “AWS Glue 트리거” 단원을 참조하십시오.

AWS Glue에서 블루프린트 및 워크플로를 사용하여 복잡한 ETL 활동 수행

조직의 복잡한 추출, 변환, 로드 프로세스 중 일부는 여러 종속 AWS Glue 작업 및 크롤러를 사용하여 가장 잘 구현될 수 있습니다. AWS Glue 워크플로를 사용하면 AWS Glue에서 단일 엔터티로 실행하고 추적할 수 있는 복잡한 다중 작업, 다중 크롤러 ETL 프로세스를 설계할 수 있습니다. 워크플로를 생성하고 워크플로에서 작업, 크롤러 및 트리거를 지정한 후 온디맨드로 또는 일정에 따라 워크플로를 실행할 수 있습니다.

주제

- AWS Glue의 워크플로 개요

- AWS Glue에서 워크플로 수동 생성 및 구축
- Amazon EventBridge 이벤트로 AWS Glue 워크플로 시작
- 워크플로를 시작한 EventBridge 이벤트 보기
- AWS Glue에서 워크플로 실행 및 모니터링
- 워크플로 실행 중지
- 워크플로 실행 복구 및 재개
- AWS Glue에서 워크플로 실행 속성 가져오기 및 설정
- AWS Glue API를 사용하여 워크플로 쿼리
- AWS Glue의 블루프린트 및 워크플로 제한 사항
- AWS Glue의 블루프린트 오류 해결
- AWS Glue 블루프린트에 대한 페르소나 및 역할의 권한

AWS Glue의 워크플로 개요

AWS Glue에서 여러 크롤러, 작업 및 트리거가 포함된 복잡한 ETL(추출, 변환 및 로드) 활동을 생성 및 시각화할 수 있습니다. 각 워크플로는 모든 구성 요소의 실행과 모니터링을 관리합니다. 워크플로는 각 구성 요소를 실행할 때 실행 진행률과 상태를 기록합니다. 이는 더 큰 태스크의 개요와 각 단계의 세부 정보를 제공합니다. AWS Glue 콘솔은 워크플로우의 시각적 표현을 그래프로 제공합니다.

AWS Glue 블루프린트에서 워크플로를 생성하거나 AWS Management Console 또는 AWS Glue API를 사용하여 구성 요소를 한 번에 수동으로 워크플로를 구축할 수 있습니다. 청사진에 대한 자세한 내용은 the section called “블루프린트 개요”을 참조하십시오.

워크플로 내의 트리거는 작업과 크롤러를 모두 시작할 수 있으며 작업 또는 크롤러가 완료되면 실행될 수 있습니다. 트리거를 사용하여 상호 종속된 작업과 크롤러로 이루어진 대형 체인을 생성할 수 있습니다. 작업 및 크롤러 종속성을 정의하는 워크플로 내의 트리거 외에도 각 워크플로에는 시작 트리거가 있습니다. 시작 트리거에는 다음 세 가지 유형이 있습니다.

- 일정 - 정의한 일정에 따라 워크플로가 시작됩니다. 일정은 매일, 매주, 매월 등이 될 수도 있고 `cron` 표현식을 기반으로 한 사용자 정의 일정이 될 수도 있습니다.
- 온디맨드 - 워크플로가 AWS Glue 콘솔, API 또는 AWS CLI에서 수동으로 시작됩니다.
- EventBridge 이벤트 – 단일 Amazon EventBridge 이벤트 또는 Amazon EventBridge 이벤트 배치가 발생하면 워크플로가 시작됩니다. 이 트리거 유형을 사용하면 AWS Glue는 이벤트 중심 아키텍처에서 이벤트 소비자가 될 수 있습니다. 모든 EventBridge 이벤트 유형은 워크플로를 시작할 수 있습니다. 일반적인 사용 사례는 Amazon S3 버킷에 새 객체가 도착하는 것입니다(S3 `PutObject` 작업).

이벤트 배치로 워크플로를 시작한다는 것은 지정된 수의 이벤트가 수신되거나 지정된 시간이 경과할 때까지 기다리는 것을 의미합니다. EventBridge 이벤트 트리거를 생성할 때 선택적으로 배치 조건을 지정할 수 있습니다. 배치 조건을 지정하는 경우 배치 크기(이벤트 수)를 지정해야 하며 선택적으로 배치 기간(초 수)을 지정할 수 있습니다. 기본 및 최대 배치 기간은 900초(15분)입니다. 먼저 충족되는 배치 조건이 워크플로를 시작합니다. 첫 번째 이벤트가 도착하면 배치 기간이 시작됩니다. 트리거를 생성할 때 배치 조건을 지정하지 않으면 배치 크기는 기본적으로 1로 설정됩니다.

워크플로가 시작되면 배치 조건이 재설정되고 이벤트 트리거는 워크플로를 다시 시작하기 위해 충족되는 다음 배치 조건을 감시하기 시작합니다.

다음 표는 배치 크기와 배치 기간이 함께 작동하여 워크플로를 트리거하는 방법을 보여줍니다.

배치 크기	배치 기간	결과 트리거 조건
10		10개의 EventBridge 이벤트가 도착하거나 첫 번째 이벤트가 도착한 후 15분 중 먼저 발생하는 시점에 워크플로가 트리거됩니다. (기간 크기를 지정하지 않으면 기본적으로 15분으로 설정됩니다.)
10	2분	10개의 EventBridge 이벤트가 도착하거나 첫 번째 이벤트가 도착한 후 2분 중 먼저 발생하는 시점에 워크플로가 트리거됩니다.
1		워크플로는 첫 번째 이벤트가 도착하면 트리거됩니다. 기간 크기는 관련이 없습니다. EventBridge 이벤트 트리거를 생성할 때 배치 조건을 지정하지 않으면 배치 크기의 기본값은 1입니다.

`GetWorkflowRun` API 작업은 워크플로를 트리거한 배치 조건을 반환합니다.

워크플로 시작 방법에 관계 없이 워크플로를 생성할 때 동시 워크플로 실행의 최대 수를 지정할 수 있습니다.

이벤트 또는 이벤트 배치가 결국 실패하는 워크플로 실행을 시작하는 경우 해당 이벤트 또는 이벤트 배치는 더 이상 워크플로 실행 시작에 대해 고려되지 않습니다. 새 워크플로 실행은 다음 이벤트 또는 이벤트 배치가 도착할 때만 시작됩니다.

> **Important**
>
> 워크플로 내 작업, 크롤러, 트리거의 총수를 100개 이하로 제한합니다. 100개가 넘게 포함될 경우 워크플로 실행을 재개하거나 중지하려고 할 때 오류가 발생할 수 있습니다.

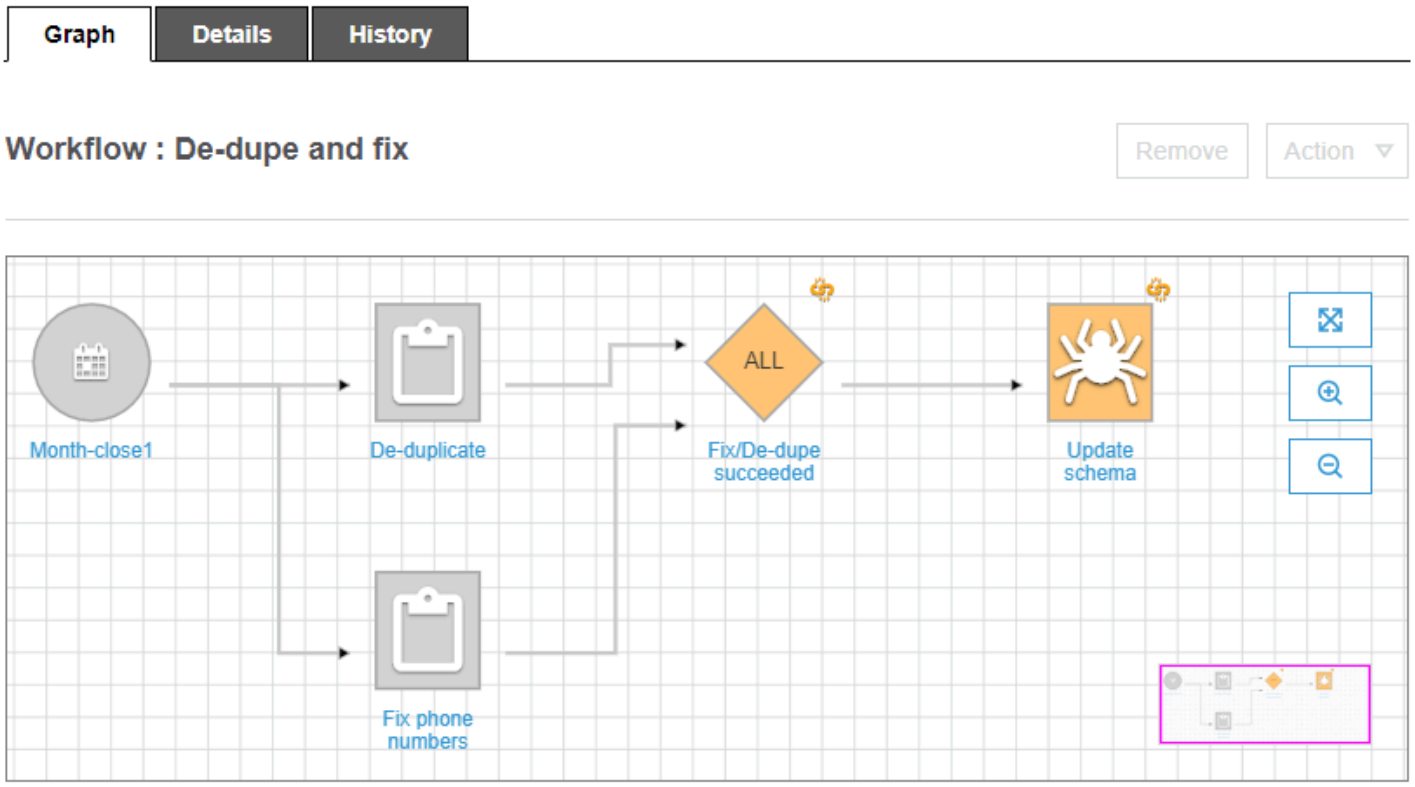
이벤트 조건이 충족되더라도 워크플로에 대해 설정된 동시성 제한을 초과하면 워크플로 실행이 시작되지 않습니다. 예상 이벤트 볼륨에 따라 워크플로 동시성 제한을 조정하는 것이 좋습니다. AWS Glue는 초과 동시성 제한으로 인해 실패한 워크플로 실행을 다시 시도하지 않습니다. 마찬가지로, 예상 이벤트 볼륨을 기반으로 워크플로 내에서 작업 및 크롤러에 대한 동시성 제한을 조정하는 것이 좋습니다.

워크플로 실행 속성

워크플로우 실행 전체의 상태를 공유 및 관리하려면 기본 워크플로우 실행 속성을 정의하면 됩니다. 이름/값 페어인 이러한 속성은 워크플로우의 모든 작업에서 사용할 수 있습니다. AWS Glue API를 사용하여 작업은 워크플로우 실행 속성을 검색하고 워크플로우에 나중에 오는 작업을 위해 수정할 수 있습니다.

워크플로 그래프

다음 이미지에서는 AWS Glue 콘솔에서 매우 기본적인 워크플로의 그래프를 보여줍니다. 워크플로우에는 구성 요소가 수십 개 있을 수 있습니다.



이 워크플로는 2개의 작업 `De-duplicate` 및 `Fix phone numbers`를 시작하는 일정 트리거 `Month-close1`에 의해 시작됩니다. 두 작업이 성공적으로 완료되면 이벤트 트리거 `Fix/De-dupe succeeded`가 크롤러 `Update schema`를 시작합니다.

정적 및 동적 워크플로 보기

워크플로우별로 정적 보기와 동적 보기라는 개념이 있습니다. 정적 보기는 워크플로우의 설계를 나타냅니다. 동적 보기는 각 작업 및 크롤러에 대한 최신 실행 정보를 포함하는 런타임 보기입니다. 실행 정보에는 성공 상태와 오류 세부 정보가 포함됩니다.

워크플로우가 실행 중이면 콘솔에 완료된 작업과 아직 실행될 작업을 그래픽으로 나타내는 동적 보기가 표시됩니다. AWS Glue API를 사용하여 실행 중인 워크플로의 동적 보기를 검색할 수도 있습니다. 자세한 내용은 AWS Glue API를 사용하여 워크플로 쿼리 단원을 참조하십시오.

다음 사항도 참조하세요.

- the section called “블루프린트에서 워크플로 생성”
- the section called “워크플로 수동 생성 및 구축”
- 워크플로(워크플로 API의 경우)

AWS Glue에서 워크플로 수동 생성 및 구축

AWS Glue 콘솔을 사용하여 한 번에 한 노드씩 워크플로를 수동으로 생성하고 구축할 수 있습니다.

워크플로우에는 작업, 크롤러 및 트리거가 포함됩니다. 수동으로 워크플로를 생성하기 전에 워크플로에 포함시킬 작업과 크롤러를 생성합니다. 워크플로에 요청 시 실행되는 크롤러를 지정하는 것이 좋습니다. 워크플로우를 구축하는 동안 새 트리거를 생성할 수도 있고, 기존 트리거를 워크플로우에 복제할 수도 있습니다. 트리거를 복제하면 트리거와 연결된 모든 카탈로그 객체(이를 실행하는 작업 또는 크롤러 및 시작하는 작업 또는 크롤러)가 워크플로에 추가됩니다.

> **Important**
>
> 워크플로 내 작업, 크롤러, 트리거의 총수를 100개 이하로 제한합니다. 100개가 넘게 포함될 경우 워크플로 실행을 재개하거나 중지하려고 할 때 오류가 발생할 수 있습니다.

워크플로우 그래프에 트리거를 추가하고 각 트리거에 대해 감시되는 이벤트와 작업을 정의하여 워크플로우를 구축합니다. 온디맨드 트리거 또는 일정 트리거인 시작 트리거를 시작하고, 이벤트(조건부) 트리거를 추가하여 그래프를 완성합니다.

1단계: 워크플로우 생성

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 ETL 아래에서 Workflows(워크플로우)를 선택합니다.
3. Add workflow(워크플로우 추가)를 선택하고 Add a new ETL workflow(새 ETL 워크플로우 추가) 양식을 작성합니다.

   추가하는 선택적 기본 실행 속성은 워크플로우의 모든 작업에 대한 인수로 사용할 수 있습니다. 자세한 내용은 AWS Glue에서 워크플로 실행 속성 가져오기 및 설정 단원을 참조하십시오.
4. Add workflow(워크플로우 추가)를 선택합니다.

   새 워크플로우가 Workflows(워크플로우) 페이지의 목록에 나타납니다.

2단계: 시작 트리거 추가

1. Workflows(워크플로우) 페이지에서 새 워크플로우를 선택합니다. 그런 다음 페이지 하단에서 [그래프(Graph)] 탭이 선택되어 있는지 확인합니다.

2. Add trigger(트리거 추가)를 선택하고 Add trigger(트리거 추가) 대화 상자에서 다음 중 하나를 수행합니다.

   - Clone existing(기존 복제)을 선택하고 복제할 트리거를 선택합니다. 그런 다음 추가를 선택합니다.

     트리거가 감시하는 작업 및 크롤러와 트리거가 시작하는 작업 및 크롤러와 함께 트리거가 그래프에 나타납니다.

     실수로 잘못된 트리거를 선택한 경우 그래프에서 해당 트리거를 선택한 후 Remove(제거)를 선택합니다.

   - Add new(새로 추가)를 선택하고 Add trigger(트리거 추가) 양식을 작성합니다.

     1. [트리거 유형(Trigger type)]에 대해 [일정(Schedule)], [온디맨드(On demand)] 또는 [EventBridge 이벤트(EventBridge event)]를 선택합니다.

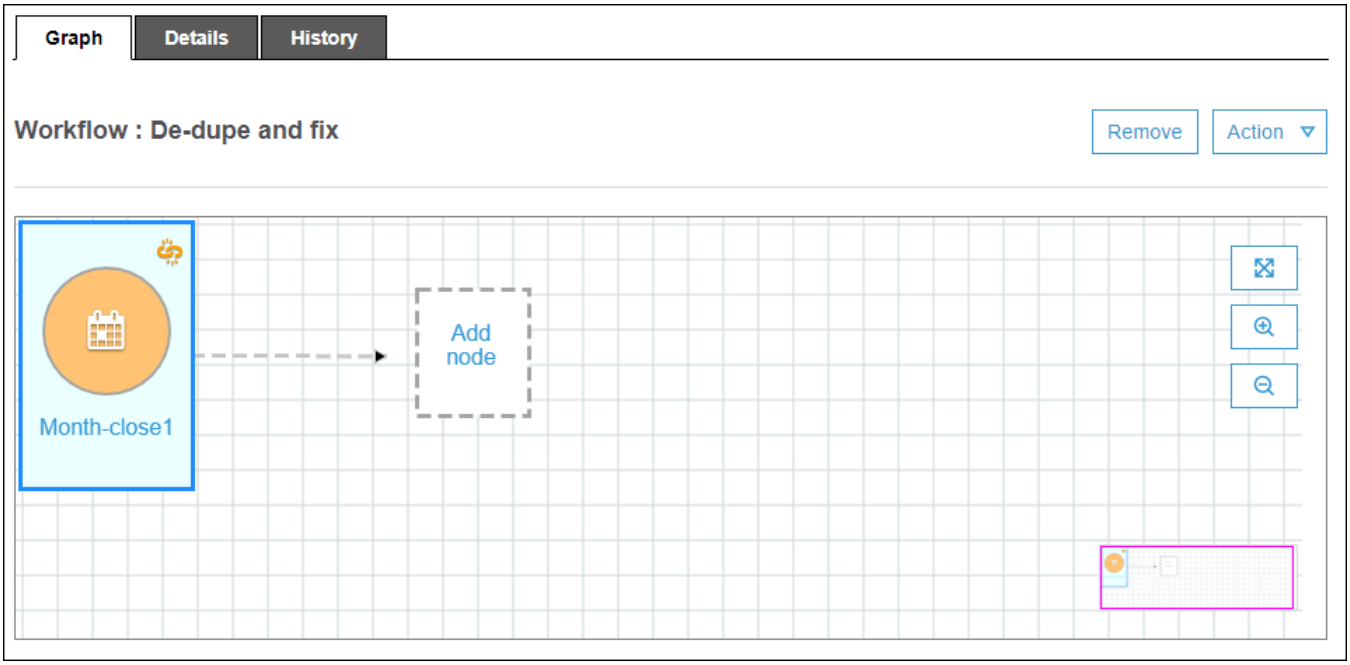
        트리거 유형 [일정(Schedule)]에서 [주파수(Frequency)] 옵션 중 하나를 선택합니다. [사용자 정의(Custom)]를 선택하여 `cron` 표현식을 입력합니다.

        트리거 유형 [EventBridge 이벤트(EventBridge event)]의 경우 [이벤트 수(Number of events)](배치 크기)를 입력하고 선택적으로 [시간 지연(Time delay)](배치 기간)을 입력합니다. [시간 지연(Time delay)]을 생략하면 배치 기간은 기본적으로 15분으로 설정됩니다. 자세한 내용은 AWS Glue의 워크플로 개요 단원을 참조하십시오.

     2. 추가(Add)를 선택합니다.

     자리 표시자 노드(Add node(노드 추가)라고 레이블이 지정됨)와 함께 트리거가 그래프에 나타납니다. 아래 예에서 시작 트리거는 `Month-close1`이라는 일정 트리거입니다.

     이때 트리거는 아직 저장되어 있지 않습니다.



3. 새 트리거를 추가한 경우 다음 단계를 완료합니다.

   a. 다음 중 하나를 수행합니다.

      - 자리 표시자 노드(Add node(노드 추가))를 선택합니다.
      - 시작 트리거가 선택되어 있는지 확인하고, 그래프 위의 Action(작업) 메뉴에서 Add jobs/crawlers to trigger(트리거할 작업/크롤러 추가)를 선택합니다.

   b. Add jobs(s) and crawler(s) to trigger(트리거할 작업 및 크롤러 추가) 대화 상자에서 작업 또는 크롤러를 하나 이상 선택한 후 Add(추가)를 선택합니다.

      트리거가 저장되고, 선택한 작업 또는 크롤러가 트리거의 커넥터와 함께 그래프에 나타납니다.

      실수로 잘못된 작업 또는 크롤러를 추가한 경우, 해당 트리거 또는 커넥터를 선택하고 Remove(제거)를 선택하면 됩니다.

3단계: 트리거 추가

[이벤트(Event)] 유형의 트리거를 더 추가하여 워크플로를 계속 구축합니다. 그래프 캔버스를 축소 또는 확대하려면 그래프 오른쪽의 아이콘을 사용합니다. 추가할 각 트리거에 대해 다음 단계를 완료합니다.

> **Note**
>
> 워크플로를 저장하는 작업은 없습니다. 마지막 트리거를 추가하고 트리거에 작업을 할당하면 워크플로가 완료되고 저장됩니다. 나중에 언제든지 다시 돌아와서 노드를 더 추가할 수 있습니다.

1. 다음 중 하나를 수행합니다.

   - 기존 트리거를 복제하려면 그래프의 노드가 선택되어 있지 않은지 확인하고, Action(작업) 메뉴에서 Add trigger(트리거 추가)를 선택합니다.
   - 그래프에서 특정 작업 또는 크롤러를 감시하는 새 트리거를 추가하려면 작업 또는 크롤러 노드를 선택한 후 Add trigger(트리거 추가) 자리 표시자 노드를 선택합니다.

     이후 단계에서 이 트리거가 감시할 작업 또는 크롤러를 더 추가할 수 있습니다.

2. Add Folder(폴더 추가) 대화 상자에서 다음 중 하나를 수행합니다.

   - Add new(새로 추가)를 선택하고 Add trigger(트리거 추가) 양식을 작성합니다. 그런 다음 추가를 선택합니다.

     트리거가 그래프에 나타납니다. 이후 단계에서 트리거를 완료합니다.
   - Clone existing(기존 복제)을 선택하고 복제할 트리거를 선택합니다. 그런 다음 추가를 선택합니다.

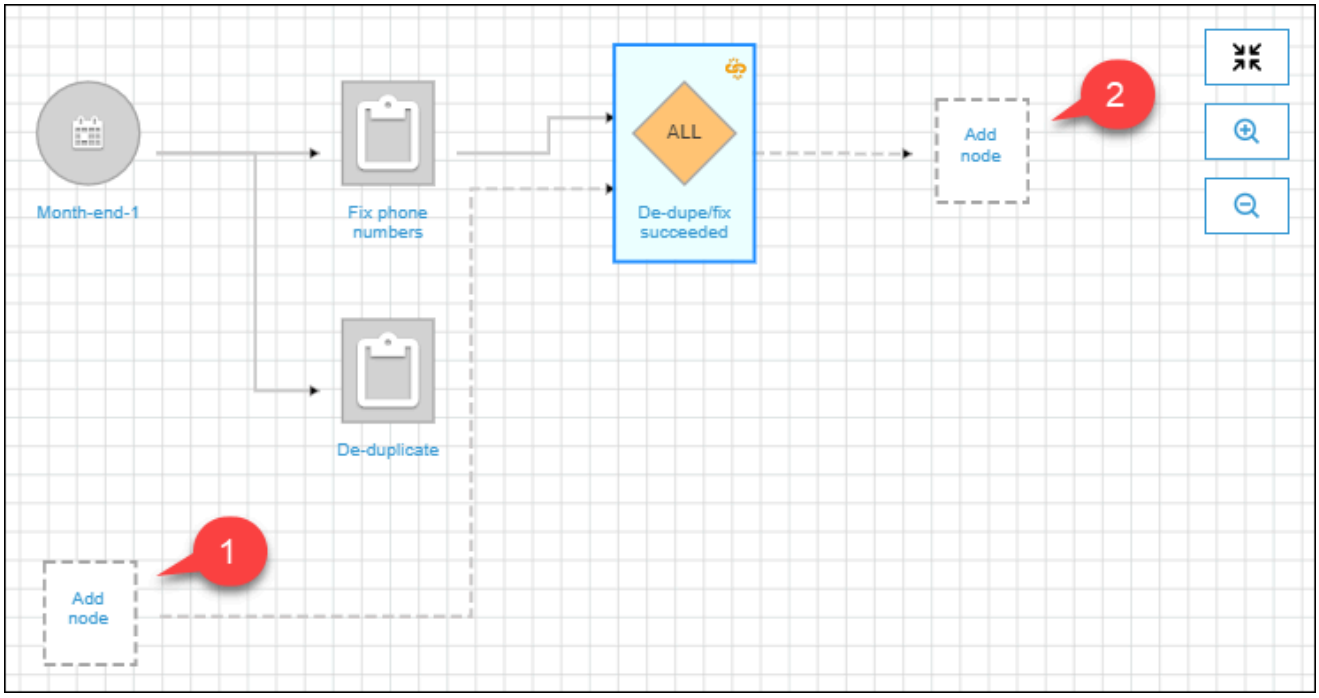
     트리거가 감시하는 작업 및 크롤러와 트리거가 시작하는 작업 및 크롤러와 함께 트리거가 그래프에 나타납니다.

     실수로 잘못된 트리거를 선택한 경우 그래프에서 해당 트리거를 선택한 후 Remove(제거)를 선택합니다.

3. 새 트리거를 추가한 경우 다음 단계를 완료합니다.

   a. 새 트리거를 선택합니다.

      다음 그래프에 나와 있듯이 트리거 `De-dupe/fix succeeded`가선택되고 감시할 (1) 이벤트와 (2) 작업에 대한 자리 표시자 노드가 나타납니다.



b. (트리거가 이미 이벤트를 감시하고 감시할 작업 또는 크롤러를 더 추가하려는 경우의 선택 사항) 감시할 이벤트 자리 표시자 노드를 선택하고, Add job(s) and crawler(s) to watch(감시할 작업 및 크롤러 추가) 대화 상자에서 작업 또는 크롤러를 하나 이상 선택합니다. 감시할 이벤트를 선택하고(SUCCEEDED, FAILED 등) Add(추가)를 선택합니다.

c. 트리거가 선택되었는지 확인하고, 작업 자리 표시자 노드를 선택합니다.

d. Add job(s) and crawler(s) to watch(감시할 작업 및 크롤러 추가) 대화 상자에서 작업 또는 크롤러를 하나 이상 선택하고 Add(추가)를 선택합니다.

   선택한 작업 및 크롤러가 트리거의 커넥터와 함께 그래프에 나타납니다.

워크플로와 블루프린트에 대한 자세한 내용은 다음 주제를 참조하세요.

- AWS Glue의 워크플로 개요
- AWS Glue에서 워크플로 실행 및 모니터링
- AWS Glue의 블루프린트에서 워크플로 생성

Amazon EventBridge 이벤트로 AWS Glue 워크플로 시작

CloudWatch Events라고도 하는 Amazon EventBridge를 사용하면 AWS 서비스를 자동화하고 애플리케이션 가용성 문제나 리소스 변경 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. AWS 서비스

의 이벤트는 거의 실시간으로 EventBridge로 전송됩니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다.

EventBridge 지원으로 AWS Glue는 이벤트 중심 아키텍처에서 이벤트 생산자 및 소비자 역할을 할 수 있습니다. 워크플로의 경우 AWS Glue는 모든 유형의 EventBridge 이벤트를 소비자로 지원합니다. 가장 일반적인 사용 사례는 Amazon S3 버킷에 새 객체 도착입니다. 데이터가 불규칙하거나 정의되지 않은 간격으로 도착하는 경우 이 데이터를 도착에 최대한 가깝게 처리할 수 있습니다.

> **Note**
>
> AWS Glue는 EventBridge 메시지의 배달을 보장하지 않습니다. AWS Glue는 EventBridge가 중복 메시지를 전달하는 경우 중복 제거를 수행하지 않습니다. 사용 사례에 따라 멱등성을 관리해야 합니다.
> 원하지 않는 이벤트가 전송되지 않도록 EventBridge 규칙을 올바르게 구성해야 합니다.

시작하기 전 준비 사항

Amazon S3 데이터 이벤트로 워크플로를 시작하려면 관심 있는 S3 버킷에 대한 이벤트가 AWS CloudTrail 및 EventBridge에 기록되었는지 확인해야 합니다. 그러려면 CloudTrail 추적을 생성해야 합니다. 자세한 내용은 AWS 계정에 대한 추적 생성을 참조하세요.

EventBridge 이벤트로 워크플로를 시작하려면

> **Note**
>
> 다음 명령에서 다음과 같이 바꿉니다.
>
> - *<workflow-name>*을 워크플로에 할당할 이름으로
> - *<trigger-name>*을 트리거에 할당할 이름으로
> - *<bucket-name>*을 Amazon S3 버킷의 이름으로
> - *<account-id>*를 유효한 AWS 계정 ID로
> - *<region>*을 리전 이름(예: `us-east-1`)으로
> - *<rule-name>*을 EventBridge 규칙에 할당할 이름으로

1. EventBridge 규칙 및 대상을 생성하고 볼 수 있는 AWS Identity and Access Management(IAM) 권한이 있는지 확인합니다. 다음은 연결할 수 있는 샘플 정책입니다. 작업 및 리소스에 대한 제한을 설정하기 위해 범위를 좁힐 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:DisableRule",
                "events:DeleteRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "events:EnableRule",
                "events:List*",
                "events:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

2. AWS Glue에 이벤트를 전달할 때 EventBridge 서비스가 수임할 수 있는 IAM 역할을 생성합니다.

   a. IAM 콘솔의 역할 생성 페이지에서 AWS 서비스를 선택합니다. 그런 다음 [CloudWatch Events] 서비스를 선택합니다.

   b. [역할 생성(Create role)] 마법사를 완료합니다. 마법사가 자동으로 `CloudWatchEventsBuiltInTargetExecutionAccess` 및 `CloudWatchEventsInvocationAccess` 정책을 연결합니다.

   c. 다음 인라인 정책을 역할에 연결합니다. 이 정책은 EventBridge 서비스가 이벤트를 AWS Glue로 보낼 수 있도록 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:notifyEvent"
```

```
      ],
      "Resource": [
        "arn:aws:glue:<region>:<account-id>:workflow/<workflow-name>"
      ]
    }
  ]
}
```

3. 다음 명령을 입력하여 워크플로를 생성합니다.

   선택적 추가 명령줄 파라미터에 대한 자세한 내용은 AWS CLI Command Reference의 create-workflow를 참조하세요.

   ```
   aws glue create-workflow --name <workflow-name>
   ```

4. 다음 명령을 입력하여 워크플로에 대한 EventBridge 이벤트 트리거를 생성합니다. 워크플로의 시작 트리거가 됩니다. *<actions>*를 수행할 작업(시작할 작업 및 크롤러)으로 바꿉니다.

   `actions` 인수를 코딩하는 방법에 대한 자세한 내용은 AWS CLI Command Reference의 create-trigger를 참조하세요.

   ```
   aws glue create-trigger --workflow-name <workflow-name> --type EVENT --
   name <trigger-name> --actions <actions>
   ```

   단일 EventBridge 이벤트 대신 이벤트 일괄 처리에 의해 워크플로가 트리거되도록 하려면 대신 다음 명령을 입력합니다.

   ```
   aws glue create-trigger --workflow-name <workflow-name> --type EVENT
    --name <trigger-name> --event-batching-condition BatchSize=<number-of-
   events>,BatchWindow=<seconds> --actions <actions>
   ```

   `event-batching-condition` 인수의 경우 `BatchSize`는 필수이고 `BatchWindow`는 선택 사항입니다. `BatchWindow`를 생략하면 기간의 기본값은 최대 기간 크기인 900초입니다.

   Example

   다음 예제에서는 3개의 EventBridge 이벤트가 도착한 후 또는 첫 번째 이벤트가 도착하고 5분 후 중 먼저 해당되는 시점에 `eventtest` 워크플로를 시작하는 트리거를 생성합니다.

```
aws glue create-trigger --workflow-name eventtest --type EVENT --name objectArrival
 --event-batching-condition BatchSize=3,BatchWindow=300 --actions JobName=test1
```

5. Amazon EventBridge에서 규칙을 생성합니다.

   a. 원하는 텍스트 편집기에서 규칙 세부 정보에 대한 JSON 객체를 생성합니다.

   다음 예에서는 Amazon S3를 이벤트 소스로, `PutObject`를 이벤트 이름으로, 버킷 이름을 요청 파라미터로 지정합니다. 이 규칙은 새 객체가 버킷에 도착하면 워크플로를 시작합니다.

```
{
  "source": [
    "aws.s3"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3.amazonaws.com"
    ],
    "eventName": [
      "PutObject"
    ],
    "requestParameters": {
      "bucketName": [
        "<bucket-name>"
      ]
    }
  }
}
```

   버킷 내의 폴더에 새 객체가 도착할 때 워크플로를 시작하려면 `requestParameters`를 다음 코드로 대체합니다.

```
    "requestParameters": {
      "bucketName": [
        "<bucket-name>"
      ]
      "key" : [{ "prefix" : "<folder1>/<folder2>/*"}}]
  }
```

b. 선호하는 도구를 사용하여 규칙 JSON 객체를 이스케이프 처리된 문자열로 변환합니다.

```
{\n  \"source\": [\n    \"aws.s3\"\n  ],\n  \"detail-type\": [\n    \"AWS API
 Call via CloudTrail\"\n  ],\n  \"detail\": {\n    \"eventSource\": [\n
 \"s3.amazonaws.com\"\n    ],\n    \"eventName\": [\n      \"PutObject\"\n
  ],\n    \"requestParameters\": {\n      \"bucketName\": [\n        \"<bucket-
name>\"\n      ]\n    }\n  }\n}
```

c. 다음 명령을 실행하여 후속 `put-rule` 명령에 대한 입력 파라미터를 지정하기 위해 편집할 수 있는 JSON 파라미터 템플릿을 생성합니다. 출력을 파일에 저장합니다. 이 예에서 파일 이름은 `ruleCommand`입니다.

```
aws events put-rule --name <rule-name> --generate-cli-skeleton >ruleCommand
```

`--generate-cli-skeleton` 파라미터에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 JSON 또는 YAML 입력 파일에서 AWS CLI 스켈레톤 및 입력 파라미터 생성을 참조하세요.

출력 파일은 다음과 같습니다.

```
{
    "Name": "",
    "ScheduleExpression": "",
    "EventPattern": "",
    "State": "ENABLED",
    "Description": "",
    "RoleArn": "",
    "Tags": [
        {
            "Key": "",
            "Value": ""
        }
    ],
    "EventBusName": ""
}
```

d. 파일을 편집하여 선택적으로 파라미터를 제거하고 최소한 `Name`, `EventPattern` 및 `State` 파라미터를 지정합니다. `EventPattern` 파라미터에 대해 이전 단계에서 생성한 규칙 세부 정보에 대한 이스케이프 처리된 문자열을 제공합니다.

```
{
```

```
    "Name": "<rule-name>",
    "EventPattern": "{\n  \"source\": [\n    \"aws.s3\"\n  ],\n  \"detail-
type\": [\n    \"AWS API Call via CloudTrail\"\n  ],\n  \"detail\": {\n
 \"eventSource\": [\n      \"s3.amazonaws.com\"\n    ],\n    \"eventName\": [\n
      \"PutObject\"\n    ],\n    \"requestParameters\": {\n      \"bucketName
\": [\n        \"<bucket-name>\"\n      ]\n    }\n  }\n}",
    "State": "DISABLED",
    "Description": "Start an AWS Glue workflow upon new file arrival in an
 Amazon S3 bucket"
}
```

> **Note**
>
> 워크플로 구축을 마칠 때까지 규칙을 사용 중지 상태로 두는 것이 가장 좋습니다.

e. 파일 ruleCommand에서 입력 파라미터를 읽는 다음 put-rule 명령을 입력합니다.

```
aws events put-rule --name <rule-name> --cli-input-json file://ruleCommand
```

다음 출력은 성공을 나타냅니다.

```
{
    "RuleArn": "<rule-arn>"
}
```

6. 다음 명령을 입력하여 대상에 규칙을 연결합니다. 대상은 AWS Glue의 워크플로입니다. *<role-name>*을 이 절차의 시작 부분에서 생성한 역할로 바꿉니다.

```
aws events put-targets --rule <rule-name> --targets
 "Id"="1","Arn"="arn:aws:glue:<region>:<account-id>:workflow/<workflow-
name>","RoleArn"="arn:aws:iam::<account-id>:role/<role-name>" --region <region>
```

다음 출력은 성공을 나타냅니다.

```
{
    "FailedEntryCount": 0,
    "FailedEntries": []
}
```

7. 다음 명령을 입력하여 규칙과 대상이 성공적으로 연결되었는지 확인합니다.

```
aws events list-rule-names-by-target --target-arn arn:aws:glue:<region>:<account-id>:workflow/<workflow-name>
```

다음 출력은 성공을 나타냅니다. 여기서 *<rule-name>*은 생성한 규칙의 이름입니다.

```
{
    "RuleNames": [
        "<rule-name>"
    ]
}
```

8. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
9. 워크플로를 선택하고 시작 트리거와 해당 작업(시작되는 작업 또는 크롤러) 이 워크플로 그래프에 나타나는지 확인합니다. 그리고 3단계: 트리거 추가의 절차로 진행합니다. 또는 AWS Glue API나 AWS Command Line Interface를 사용하여 워크플로에 구성 요소를 더 추가합니다.
10. 워크플로가 완전히 지정되면 규칙을 사용합니다.

```
aws events enable-rule --name <rule-name>
```

이제 EventBridge 이벤트 또는 이벤트 배치로 워크플로를 시작할 준비가 되었습니다.

다음 사항도 참조하세요.

- Amazon EventBridge 사용 설명서
- AWS Glue의 워크플로 개요
- AWS Glue에서 워크플로 수동 생성 및 구축

워크플로를 시작한 EventBridge 이벤트 보기

워크플로를 시작한 Amazon EventBridge 이벤트의 이벤트 ID를 볼 수 있습니다. 워크플로가 이벤트 배치로 시작된 경우 배치에 있는 모든 이벤트의 이벤트 ID를 볼 수 있습니다.

배치 크기가 1보다 큰 워크플로의 경우 어떤 배치 조건이 워크플로를 시작했는지 확인할 수도 있습니다(배치 크기의 이벤트 수 도착 또는 배치 기간 만료).

워크플로를 시작한 EventBridge 이벤트를 보려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 [워크플로(Workflows)]를 선택합니다.
3. 워크플로를 선택합니다. 그런 다음 하단에서 [기록(History)] 탭을 선택합니다.
4. 워크플로 실행을 선택한 다음 [실행 세부 정보 보기(View run details)]를 선택합니다.
5. 실행 세부 정보 페이지에서 [실행 속성(Run properties)] 필드를 찾아 [aws:eventIds] 키를 찾습니다.

   해당 키의 값은 EventBridge 이벤트 ID 목록입니다.

워크플로를 시작한 EventBridge 이벤트를 보려면(AWS API)

- Python 스크립트에 다음 코드를 포함합니다.

```
workflow_params =
 glue_client.get_workflow_run_properties(Name=workflow_name,RunId=workflow_run_id)
batched_events = workflow_params['aws:eventIds']
```

   `batched_events`는 문자열 목록이며 각 문자열은 이벤트 ID입니다.

> 다음 사항도 참조하세요.
>
> - Amazon EventBridge 사용 설명서
> - the section called “워크플로우 개요”

AWS Glue에서 워크플로 실행 및 모니터링

워크플로우의 시작 트리거가 온디맨드 트리거인 경우, AWS Glue 콘솔에서 워크플로우를 시작할 수 있습니다. 워크플로를 실행하고 모니터링하려면 다음 단계를 수행합니다. 워크플로가 실패하면 실행 그래프를 보고 실패한 노드를 확인할 수 있습니다. 블루프린트에서 워크플로가 생성된 경우 문제를 해결하기 위해 블루프린트 실행을 보고 워크플로를 생성하는 데 사용된 블루프린트 파라미터 값을 볼 수 있습니다. 자세한 내용은 the section called “블루프린트 실행 보기” 단원을 참조하십시오.

AWS Glue 콘솔, API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 워크플로를 실행하고 모니터링할 수 있습니다.

워크플로를 실행하고 모니터링하려면(콘솔)

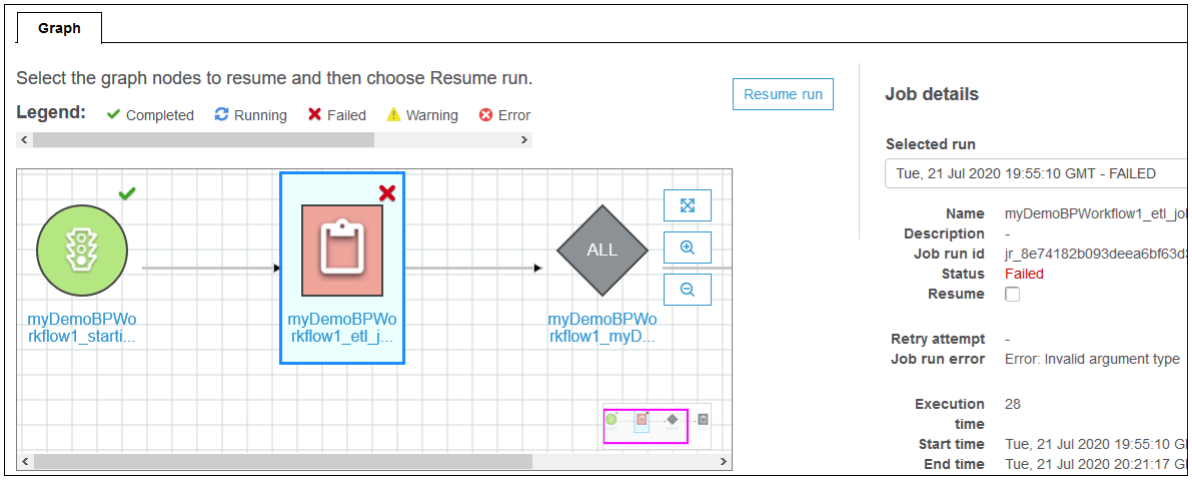
1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 ETL 아래에서 Workflows(워크플로우)를 선택합니다.
3. 워크플로우를 선택합니다. Actions(작업) 메뉴에서 Run(실행)을 선택합니다.
4. 워크플로 목록에서 [마지막 실행 상태(Last run status)] 열을 확인합니다. 새로 고침 버튼을 선택하여 진행 중인 워크플로 상태를 봅니다.
5. 워크플로가 실행 중이거나 완료된 후(또는 실패한 후) 다음 단계를 완료하여 실행 세부 정보를 확인합니다.

   a. 워크플로가 선택되어 있는지 확인하고 [기록(History)] 탭을 선택합니다.

   b. 현재 또는 가장 최근 워크플로 실행을 선택한 다음 [실행 세부 정보 보기(View run details)]를 선택합니다.

      워크플로 런타임 그래프는 현재 실행 상태를 보여줍니다.

   c. 그래프에서 노드를 선택하여 노드의 세부 정보와 상태를 봅니다.



워크플로를 실행하고 모니터링하려면(AWS CLI)

1. 다음 명령을 입력합니다. *<workflow-name>*을 실행할 워크플로우로 바꿉니다.

```
aws glue start-workflow-run --name <workflow-name>
```

워크플로우가 성공적으로 시작되면 명령은 실행 ID를 반환합니다.

2. get-workflow-run 명령을 사용하여 워크플로 실행 상태를 봅니다. 워크플로 이름과 실행 ID를 제공합니다.

```
aws glue get-workflow-run --name myWorkflow --run-id
 wr_d2af14217e8eae775ba7b1fc6fc7a42c795aed3cbcd8763f9415452e2dbc8705
```

다음은 샘플 명령 출력입니다.

```
{
    "Run": {
        "Name": "myWorkflow",
        "WorkflowRunId":
 "wr_d2af14217e8eae775ba7b1fc6fc7a42c795aed3cbcd8763f9415452e2dbc8705",
        "WorkflowRunProperties": {
            "run_state": "COMPLETED",
            "unique_id": "fee63f30-c512-4742-a9b1-7c8183bdaae2"
        },
        "StartedOn": 1578556843.049,
        "CompletedOn": 1578558649.928,
        "Status": "COMPLETED",
        "Statistics": {
            "TotalActions": 11,
            "TimeoutActions": 0,
            "FailedActions": 0,
            "StoppedActions": 0,
            "SucceededActions": 9,
            "RunningActions": 0,
            "ErroredActions": 0
        }
    }
}
```

다음 사항도 참조하세요.

- the section called “워크플로우 개요”
- the section called “블루프린트 개요”

워크플로 실행 중지

AWS Glue 콘솔, AWS Command Line Interface(AWS CLI) 또는 AWS Glue API를 사용하여 워크플로우 실행을 중지할 수 있습니다. 워크플로우 실행을 중지하면 실행 중인 모든 작업과 크롤러가 즉시 종료되고 아직 시작되지 않은 작업과 크롤러는 시작되지 않습니다. 실행 중인 모든 작업 및 크롤러가 중지되려면 최대 1분이 걸릴 수 있습니다. 워크플로우 실행 상태가 Running(실행 중)에서 Stopping(중지 중)으로 바뀌고, 워크플로우 실행이 완전히 중지되면 상태가 Stopped(중지됨)로 바뀝니다.

워크플로우 실행이 중지된 후 실행 그래프를 보고 완료된 작업 및 크롤러와 시작되지 않은 작업 및 크롤러를 확인할 수 있습니다. 그런 다음 데이터 무결성을 보장하기 위한 단계를 수행해야 하는지 여부를 결정할 수 있습니다. 워크플로우 실행을 중지하면 자동 롤백 작업이 수행되지 않습니다.

워크플로우 실행을 중지하려면(콘솔)

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 ETL 아래에서 Workflows(워크플로우)를 선택합니다.
3. 실행 중인 워크플로우를 선택한 다음 History(기록) 탭을 선택합니다.
4. 워크플로우 실행을 선택한 다음 Stop run(실행 중지)을 선택합니다.

   실행 상태가 Stopping(중지 중)으로 변경됩니다.
5. (선택 사항) 워크플로우 실행을 선택하고 View run details(실행 세부 정보 보기)를 선택한 다음 실행 그래프를 검토합니다.

워크플로우 실행을 중지하려면(AWS CLI)

- 다음 명령을 입력합니다. *<workflow-name>*을 워크플로우의 이름으로 바꾸고 *<run-id>*를 중지할 워크플로우 실행의 실행 ID로 바꿉니다.

```
aws glue stop-workflow-run --name <workflow-name> --run-id <run-id>
```

  다음은 stop-workflow-run 명령의 예입니다.

```
aws glue stop-workflow-run --name my-workflow --run-id
 wr_137b88917411d128081069901e4a80595d97f719282094b7f271d09576770354
```

워크플로 실행 복구 및 재개

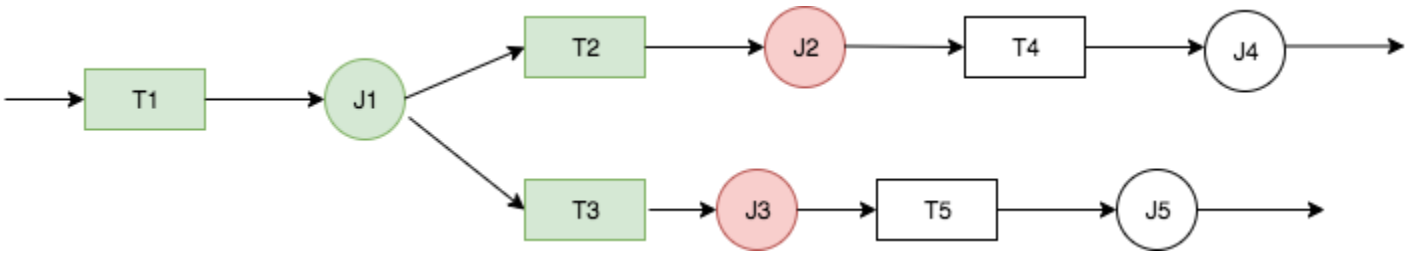
워크플로에서 하나 이상의 노드(작업 또는 크롤러)가 성공적으로 완료되지 않으면 워크플로가 부분적으로만 실행된 것입니다. 근본 원인을 찾아 수정한 후 워크플로 실행을 재개할 노드를 하나 이상 선택한 다음 워크플로 실행을 재개할 수 있습니다. 그런 다음 선택한 노드와 해당 노드에서 다운스트림인 모든 노드가 실행됩니다.

주제

- 워크플로 실행 재개: 작동 방식
- 워크플로 실행 재개
- 워크플로 실행 재개에 대한 참고 사항 및 제한 사항

워크플로 실행 재개: 작동 방식

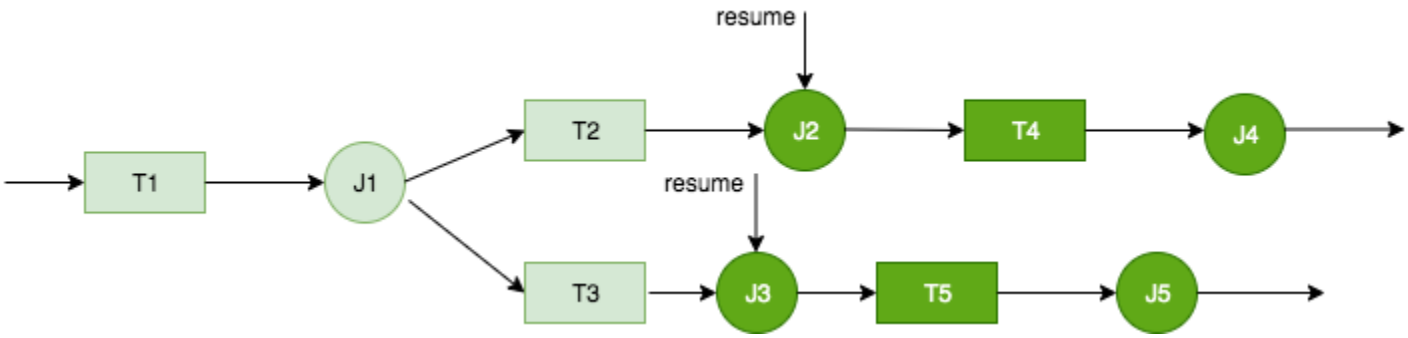
다음 다이어그램에서 워크플로 W1을 고려합니다.



워크플로 실행은 다음과 같이 진행됩니다.

1. 트리거 T1이 작업 J1을 시작합니다.
2. J1 발생이 성공적으로 완료되면 각각 J2 및 J3 작업을 실행하는 T2 및 T3이 트리거됩니다.
3. 작업 J2 및 J3이 실패합니다.
4. 트리거 T4 및 T5는 J2 및 J3의 성공적인 완료에 따라 달라지므로 발생하지 않고 작업 J4 및 J5가 실행되지 않습니다. 워크플로 W1이 부분적으로만 실행됩니다.

이제 J2 및 J3을 실패하게 만든 문제가 수정되었다고 가정합니다. J2 및 J3이 워크플로 실행을 재개할 시작점으로 선택됩니다.



워크플로 실행은 다음과 같이 재개됩니다.

1. 작업 J2 및 J3이 성공적으로 실행됩니다.
2. T4 및 T5 발생을 트리거합니다.
3. 작업 J4 및 J5가 성공적으로 실행됩니다.

재개된 워크플로 실행은 새 실행 ID를 가진 별도의 워크플로 실행으로 추적됩니다. 워크플로 기록을 볼 때 모든 워크플로 실행에 대한 이전 실행 ID를 볼 수 있습니다. 다음 스크린샷의 예에서 실행 ID `wr_c7a22...`(두 번째 행)로 실행된 워크플로에 완료되지 않은 노드가 있습니다. 사용자가 문제를 수정하고 워크플로 실행을 재개하여 실행 ID `wr_a07e55...`(첫 번째 행)가 생성되었습니다.

> **Note**
>
> 이 논의의 나머지 부분에서 "재개된 워크플로 실행"이라는 용어는 이전 워크플로 실행이 재개될 때 생성된 워크플로 실행을 나타냅니다. "원래 워크플로 실행"은 부분적으로만 실행되어 재개해야 했던 워크플로 실행을 나타냅니다.

재개된 워크플로 실행 그래프

재개된 워크플로 실행에서는 노드의 하위 집합만 실행되지만 실행 그래프는 전체 그래프입니다. 즉, 재개된 워크플로에서 실행되지 않은 노드는 원래 워크플로 실행의 실행 그래프에서 복사됩니다. 원래 워크플로 실행에서 실행된 복사된 작업 및 크롤러 노드에는 실행 세부 정보가 포함됩니다.

이전 다이어그램의 워크플로 W1을 다시 고려하세요. 워크플로 실행이 J2 및 J3부터 재개되면 재개된 워크플로 실행에 대한 실행 그래프에 모든 작업(J1~J5)과 모든 트리거(T1~T5)가 표시됩니다. J1에 대한 실행 세부 정보는 원래 워크플로 실행에서 복사됩니다.

워크플로 실행 스냅샷

워크플로 실행이 시작되면 AWS Glue는 해당 시점의 워크플로 설계 그래프의 스냅샷을 만듭니다. 해당 스냅샷은 워크플로 실행 기간 동안 사용됩니다. 실행이 시작된 후 트리거를 변경하는 경우 해당 변경 사항이 현재 워크플로 실행에 영향을 주지 않습니다. 스냅샷은 워크플로 실행이 일관된 방식으로 진행되도록 합니다.

스냅샷은 트리거만 변경 불가능하게 만듭니다. 워크플로 실행 중 다운스트림 작업 및 크롤러에 대한 변경 사항은 현재 실행에 적용됩니다.

워크플로 실행 재개

워크플로 실행을 재개하려면 다음 단계를 따릅니다. AWS Glue 콘솔, API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 워크플로 실행을 재개할 수 있습니다.

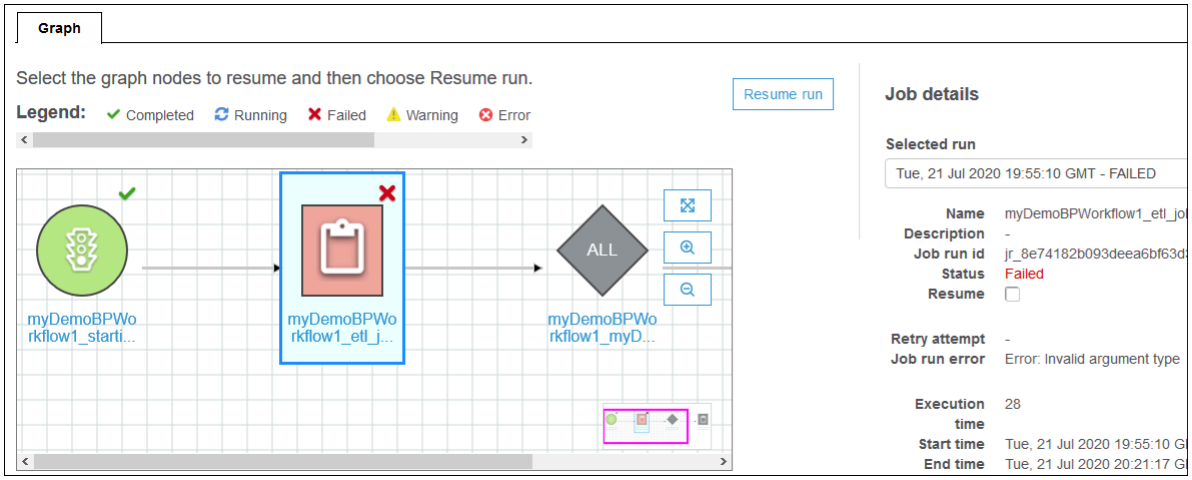
워크플로 실행을 재개하려면(콘솔)

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

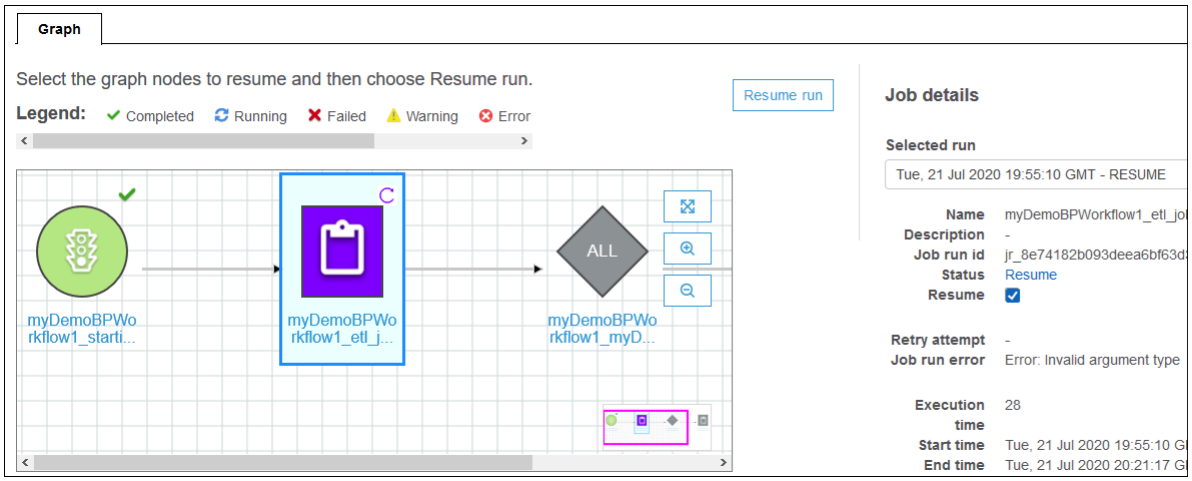
   워크플로를 보고 워크플로 실행을 재개할 수 있는 권한이 있는 사용자로 로그인합니다.

   > **Note**
   >
   > 워크플로 실행을 재개하려면 `glue:ResumeWorkflowRun` AWS Identity and Access Management(IAM) 권한이 필요합니다.

2. 탐색 창에서 [워크플로(Workflows)]를 선택합니다.
3. 워크플로를 선택한 다음 [기록(History)] 탭을 선택합니다.
4. 부분적으로만 실행된 워크플로 실행을 선택한 다음 [실행 세부 정보 보기(View run details)]를 선택합니다.
5. 실행 그래프에서 다시 시작하고 워크플로 실행을 재개할 첫 번째(또는 유일한) 노드를 선택합니다.
6. 그래프 오른쪽에 있는 세부 정보 창에서 [재개(Resume)] 확인란을 선택합니다.



노드의 색상이 변경되고 오른쪽 상단에 작은 이력서 아이콘이 표시됩니다.



7. 추가 노드를 다시 시작하려면 이전 두 단계를 완료합니다.
8. [실행 재개(Resume run)]를 선택합니다.

워크플로 실행을 재개하려면(AWS CLI)

1. `glue:ResumeWorkflowRun` IAM 권한이 있는지 확인합니다.
2. 다시 시작하려는 노드의 노드 ID를 검색합니다.

   a. 원래 워크플로 실행에 대해 `get-workflow-run` 명령을 실행합니다. 다음 예와 같이 워크플로 이름과 실행 ID를 제공하고 `--include-graph` 옵션을 추가합니다. 콘솔의 [기록(History)] 탭에서 또는 `get-workflow` 명령을 실행하여 실행 ID를 가져옵니다.

```
aws glue get-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --include-
graph
```

이 명령은 그래프의 노드와 가장자리를 큰 JSON 객체로 반환합니다.

b. 노드 객체의 Type 및 Name 속성으로 관심 노드를 찾습니다.

다음은 출력의 예제 노드 객체입니다.

```
{
    "Type": "JOB",
    "Name": "test1_post_failure_4592978",
    "UniqueId":
 "wnode_d1b2563c503078b153142ee76ce545fe5ceef66e053628a786ddd74a05da86fd",
    "JobDetails": {
        "JobRuns": [
            {
                "Id":
 "jr_690b9f7fc5cb399204bc542c6c956f39934496a5d665a42de891e5b01f59e613",
                "Attempt": 0,
                "TriggerName": "test1_aggregate_failure_649b2432",
                "JobName": "test1_post_failure_4592978",
                "StartedOn": 1595358275.375,
                "LastModifiedOn": 1595358298.785,
                "CompletedOn": 1595358298.785,
                "JobRunState": "FAILED",
                "PredecessorRuns": [],
                "AllocatedCapacity": 0,
                "ExecutionTime": 16,
                "Timeout": 2880,
                "MaxCapacity": 0.0625,
                "LogGroupName": "/aws-glue/python-jobs"
            }
        ]
    }
}
```

c. 노드 객체의 UniqueId 속성에서 노드 ID를 가져옵니다.

3. resume-workflow-run 명령을 실행합니다. 다음 예와 같이 워크플로 이름, 실행 ID 및 노드 ID 목록을 공백으로 구분하여 제공합니다.

```
aws glue resume-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --node-
ids wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3
 wnode_d1b2563c503078b153142ee76ce545fe5ceef66e053628a786ddd74a05da86fd
```

이 명령은 재개된(새) 워크플로 실행의 실행 ID와 시작될 노드 목록을 출력합니다.

```
{
    "RunId": "wr_2ada0d3209a262fc1156e4291134b3bd643491bcfb0ceead30bd3e4efac24de9",
    "NodeIds": [
        "wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3"
    ]
}
```

예제 `resume-workflow-run` 명령은 다시 시작할 2개의 노드를 나열했지만 예제 출력은 1개의 노드만 다시 시작됨을 나타냅니다. 이는 한 노드가 다른 노드의 다운스트림이었고, 다운스트림 노드는 워크플로의 정상적인 흐름에 따라 다시 시작되기 때문입니다.

워크플로 실행 재개에 대한 참고 사항 및 제한 사항

워크플로 실행을 재개할 때 다음 참고 사항과 제한 사항을 염두에 둡니다.

- `COMPLETED` 상태인 경우에만 워크플로 실행을 재개할 수 있습니다.

> **Note**
>
> 워크플로 실행에서 하나 이상의 노드가 완료되지 않은 경우에도 워크플로 실행 상태는 `COMPLETED`로 표시됩니다. 성공적으로 완료되지 않은 노드를 검색하려면 실행 그래프를 확인합니다.

- 원래 워크플로 실행이 실행을 시도한 모든 작업 또는 크롤러 노드에서 워크플로 실행을 재개할 수 있습니다. 트리거 노드에서 실행되는 워크플로를 재개할 수 없습니다.
- 노드를 다시 시작해도 상태가 재설정되지 않습니다. 부분적으로 처리된 데이터는 롤백되지 않습니다.
- 동일한 워크플로 실행을 여러 번 재개할 수 있습니다. 재개된 워크플로 실행이 부분적으로만 실행되는 경우 문제를 해결하고 재개된 실행을 재개할 수 있습니다.

- 다시 시작할 노드를 2개 선택하는 경우 이들이 서로 종속되어 있으면 업스트림 노드가 다운스트림 노드보다 먼저 실행됩니다. 실제로 다운스트림 노드를 선택하는 것은 일반적인 워크플로 흐름에 따라 실행되기 때문에 중복됩니다.

AWS Glue에서 워크플로 실행 속성 가져오기 및 설정

워크플로 실행 속성을 사용하여 AWS Glue 워크플로우의 작업 간에 상태를 공유 및 관리합니다. 워크플로를 생성할 때 기본 실행 속성을 설정할 수 있습니다. 그런 다음, 작업이 실행될 때 실행 속성 값을 검색하고 나중에 워크플로에 있는 작업에 대한 입력을 위해 필요에 따라 수정할 수 있습니다. 작업이 실행 속성을 수정할 때 새 값은 워크플로 실행에만 존재합니다. 기본 실행 속성은 영향을 받지 않습니다.

AWSGlue 작업이 워크플로의 일부가 아닌 경우 이러한 속성은 설정되지 않습니다.

ETL(추출, 변환 및 로드) 작업의 다음 샘플 Python 코드는 워크플로우 실행 속성을 가져오는 방법을 보여 줍니다.

```
import sys
import boto3
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from awsglue.context import GlueContext
from pyspark.context import SparkContext

glue_client = boto3.client("glue")
args = getResolvedOptions(sys.argv, ['JOB_NAME','WORKFLOW_NAME', 'WORKFLOW_RUN_ID'])
workflow_name = args['WORKFLOW_NAME']
workflow_run_id = args['WORKFLOW_RUN_ID']
workflow_params = glue_client.get_workflow_run_properties(Name=workflow_name,
                                       RunId=workflow_run_id)["RunProperties"]

target_database = workflow_params['target_database']
target_s3_location = workflow_params['target_s3_location']
```

다음 코드는 `target_format` 실행 속성을 `'csv'`로 설정하여 계속합니다.

```
workflow_params['target_format'] = 'csv'
glue_client.put_workflow_run_properties(Name=workflow_name, RunId=workflow_run_id,
 RunProperties=workflow_params)
```

자세한 내용은 다음을 참조하세요.

- GetWorkflowRunProperties 작업(Python: get_workflow_run_properties)
- PutWorkflowRunProperties 작업(Python: put_workflow_run_properties)

AWS Glue API를 사용하여 워크플로 쿼리

AWS Glue는 워크플로우를 관리하기 위한 다양한 API를 제공합니다. AWS Glue API를 사용하여 워크플로우의 정적 보기 또는 실행 중인 워크플로우의 동적 보기를 검색할 수 있습니다. 자세한 내용은 워크플로 단원을 참조하십시오.

주제

- 정적 보기 쿼리
- 동적 보기 쿼리

정적 보기 쿼리

`GetWorkflow` API 작업을 사용하여 워크플로우의 설계를 나타내는 정적 보기를 가져옵니다. 이 작업은 노드와 엣지로 구성된 방향성 그래프를 반환합니다. 여기에서 노드는 트리거, 작업 또는 크롤러를 나타냅니다. 엣지는 노드 간의 관계를 정의합니다. 엣지는 AWS Glue 콘솔의 그래프에서 커넥터(화살표)로 표현됩니다.

NetworkX, igraph, JGraphT, Java Universal Network/Graph(JUNG) Framework 등의 인기 있는 그래프 처리 라이브러리에서 이 작업을 사용할 수도 있습니다. 이러한 모든 라이브러리는 그래프를 비슷하게 나타내기 때문에 최소한의 변환이 필요합니다.

이 API가 반환하는 정적 보기는 워크플로우와 연결된 트리거의 최신 정의에 따라 최신 보기입니다.

그래프 정의

워크플로우 그래프 G는 정렬된 페어(N, E)입니다. 여기서 N은 노드 세트이고 E는 엣지 세트입니다. 노드는 고유한 숫자로 식별되는 그래프의 꼭짓점입니다. 노드는 트리거, 작업 또는 크롤러 유형일 수 있습니다. 예: `{name:T1, type:Trigger, uniqueId:1}, {name:J1, type:Job, uniqueId:2}`.

엣지는 2튜플의 형식(`src, dest`)입니다. 여기에서 `src`와 `dest`는 노드이고 `src`에서 `dest`로의 방향성 엣지가 있습니다.

정적 보기 쿼리의 예

작업 J1을 완료하면 작업 J2를 트리거하는 조건부 트리거 T를 생각해 보십시오.

```
J1 ---> T ---> J2
```

노드: J1, T, J2

엣지: (J1, T), (T, J2)

동적 보기 쿼리

GetWorkflowRun API 작업을 사용하여 실행 중인 워크플로우의 동적 보기를 가져옵니다. 이 작업은 워크플로우 실행과 관련된 메타데이터와 함께 그래프의 동일한 정적 보기를 반환합니다.

실행의 경우 GetWorkflowRun 호출의 작업을 나타내는 노드에는 최신 워크플로우 실행의 일부로 시작되는 작업 실행 목록이 있습니다. 이 목록을 사용하여 그래프 자체에 각 작업의 실행 상태를 표시할 수 있습니다. 아직 실행되지 않은 다운스트림 종속성의 경우 이 필드는 null로 설정됩니다. 그래프로 표시된 정보를 통해 언제든 워크플로우의 현재 상태를 알 수 있습니다.

이 API가 반환하는 동적 보기는 워크플로 실행이 시작될 때 있었던 정적 보기를 기반으로 합니다.

런타임 노드 예: {name:T1, type: Trigger, uniqueId:1}, {name:J1, type:Job, uniqueId:2, jobDetails:{jobRuns}}, {name:C1, type:Crawler, uniqueId:3, crawlerDetails:{crawls}}

예 1: 동적 보기

다음 예에서는 간단한 2 트리거 워크플로우를 보여 줍니다.

- 노드: t1, j1, t2, j2
- 엣지: (t1, j1), (j1, t2), (t2, j2)

GetWorkflow 응답에는 다음이 포함됩니다.

```
{
    Nodes : [
        {
            "type" : Trigger,
            "name" : "t1",
            "uniqueId" : 1
        },
        {
            "type" : Job,
```

```
            "name" : "j1",
            "uniqueId" : 2
        },
        {
            "type" : Trigger,
            "name" : "t2",
            "uniqueId" : 3
        },
        {
            "type" : Job,
            "name" : "j2",
            "uniqueId" : 4
        }
    ],
    Edges : [
        {
            "sourceId" : 1,
            "destinationId" : 2
        },
        {
            "sourceId" : 2,
            "destinationId" : 3
        },
        {
            "sourceId" : 3,
            "destinationId" : 4
        }
}
```

GetWorkflowRun 응답에는 다음이 포함됩니다.

```
{
    Nodes : [
        {
            "type" : Trigger,
            "name" : "t1",
            "uniqueId" : 1,
            "jobDetails" : null,
            "crawlerDetails" : null
        },
        {
            "type" : Job,
            "name" : "j1",
```

```
            "uniqueId" : 2,
            "jobDetails" : [
                {
                    "id" : "jr_12334",
                    "jobRunState" : "SUCCEEDED",
                    "errorMessage" : "error string"
                }
            ],
            "crawlerDetails" : null
        },
        {
            "type" : Trigger,
            "name" : "t2",
            "uniqueId" : 3,
            "jobDetails" : null,
            "crawlerDetails" : null
        },
        {
            "type" : Job,
            "name" : "j2",
            "uniqueId" : 4,
            "jobDetails" : [
                {
                    "id" : "jr_1233sdf4",
                    "jobRunState" : "SUCCEEDED",
                    "errorMessage" : "error string"
                }
            ],
            "crawlerDetails" : null
        }
    ],
    Edges : [
        {
            "sourceId" : 1,
            "destinationId" : 2
        },
        {
            "sourceId" : 2,
            "destinationId" : 3
        },
        {
            "sourceId" : 3,
            "destinationId" : 4
        }
```

```
}
```

예 2: 조건부 트리거가 있는 여러 작업

다음 예에서는 작업 여러 개와 조건부 트리거(t3) 하나가 있는 워크플로우를 보여 줍니다.

```
Consider Flow:
T(t1) ---> J(j1) ---> T(t2) ---> J(j2)
             |                     |
             |                     |
             >+------> T(t3) <-----+
                        |
                        |
                      J(j3)

Graph generated:
Nodes: t1, t2, t3, j1, j2, j3
Edges: (t1, j1), (j1, t2), (t2, j2), (j1, t3), (j2, t3), (t3, j3)
```

AWS Glue의 블루프린트 및 워크플로 제한 사항

다음은 블루프린트와 워크플로에 대한 제한 사항입니다.

블루프린트 제한 사항

다음 블루프린트 제한에 유의하세요.

- 블루프린트는 Amazon S3 버킷이 있는 동일한 AWS 리전에 등록되어야 합니다.
- AWS 계정 간에 블루프린트를 공유하려면 Amazon S3의 블루프린트 ZIP 아카이브에 대한 읽기 권한을 부여해야 합니다. 블루프린트 ZIP 아카이브에 대한 읽기 권한이 있는 고객은 AWS 계정에 블루프린트를 등록하여 사용할 수 있습니다.
- 블루프린트 파라미터 집합은 단일 JSON 객체로 저장됩니다. 이 객체의 최대 길이는 128KB입니다.
- 블루프린트 ZIP 아카이브의 압축되지 않은 최대 크기는 5MB입니다. 압축된 최대 크기는 1MB입니다.
- 워크플로 내 작업, 크롤러, 트리거의 총수를 100개 이하로 제한합니다. 100개가 넘게 포함될 경우 워크플로 실행을 재개하거나 중지하려고 할 때 오류가 발생할 수 있습니다.

워크플로 제한 사항

다음 워크플로 제한 사항에 유의하세요. 이러한 설명 중 일부는 수동으로 워크플로를 생성하는 사용자에 대한 것입니다.

- Amazon EventBridge 이벤트 트리거에 대한 최대 배치 크기는 100입니다. 최대 기간 크기는 900초(15분)입니다.
- 하나의 트리거는 하나의 워크플로우에만 연결할 수 있습니다.
- 시작 트리거(온디맨드 또는 일정)는 하나만 허용됩니다.
- 워크플로의 작업 또는 크롤러가 워크플로 외부의 트리거에 의해 시작된 경우, 작업 또는 크롤러 완료(성공 또는 기타)에 종속된 워크플로 내의 트리거는 실행되지 않습니다.
- 마찬가지로 워크플로의 작업 또는 크롤러에 워크플로 내부와 워크플로 외부에서 작업 또는 크롤러 완료(성공 또는 기타)에 종속된 트리거가 있는 경우에는 작업 또는 크롤러가 워크플로 내에서 시작되면 작업 또는 크롤러 완료 시 워크플로 내부의 트리거만 실행됩니다.

AWS Glue의 블루프린트 오류 해결

AWS Glue 블루프린트를 사용할 때 오류가 발생하면 다음 해결 방법에 따라 문제의 원인을 찾아 해결할 수 있습니다.

주제

- 오류: PySpark 모듈 누락
- 오류: 블루프린트 Config 파일 누락
- 오류: 가져온 파일 누락
- 오류: 리소스에서 iamPassRole을 수행할 권한이 없음
- 오류: 잘못된 cron 일정
- 오류: 같은 이름의 트리거가 이미 있음
- 오류: 이름이 foo인 워크플로가 이미 있습니다.
- 오류: 지정된 layoutGenerator 경로에서 모듈을 찾을 수 없음
- 오류: 연결 필드에 검증 오류

오류: PySpark 모듈 누락

AWS Glue에서 ["레이아웃 생성기 함수 ModuleNotFoundError를 실행하는 동안 알 수 없는 오류 발생: 'pyspark'라는 모듈 없음(Unknown error executing layout generator function ModuleNotFoundError: No module named 'pyspark')"]이라는 오류를 반환합니다.

블루프린트 아카이브의 압축을 풀면 다음 중 하나와 같을 수 있습니다.

```
$ unzip compaction.zip
Archive:  compaction.zip
   creating: compaction/
  inflating: compaction/blueprint.cfg
  inflating: compaction/layout.py
  inflating: compaction/README.md
  inflating: compaction/compaction.py

$ unzip compaction.zip
Archive:  compaction.zip
  inflating: blueprint.cfg
  inflating: compaction.py
  inflating: layout.py
  inflating: README.md
```

첫 번째 사례에서는 블루프린트와 관련된 모든 파일이 compaction이라는 폴더 아래에 배치된 다음 compaction.zip이라는 zip 파일로 변환되었습니다.

두 번째 사례에서는 블루프린트에 필요한 모든 파일이 폴더에 포함되지 않고 zip 파일인 compaction.zip 아래에 루트 파일로 추가되었습니다.

위 포맷 중 하나로 파일을 생성할 수 있습니다. 그러나 `blueprint.cfg`에 레이아웃을 생성하는 스크립트의 함수 이름에 대한 올바른 경로가 있는지 확인합니다.

예시

사례 1: 다음과 같이 `blueprint.cfg`에 `layoutGenerator`가 있어야 합니다.

```
layoutGenerator": "compaction.layout.generate_layout"
```

사례 2: 다음과 같이 `blueprint.cfg`에 `layoutGenerator`가 있어야 합니다.

```
layoutGenerator": "layout.generate_layout"
```

이 경로가 올바르게 포함되지 않은 경우 표시된 대로 오류가 발생할 수 있습니다. 예를 들어 사례 2와 같은 폴더 구조를 가지고 있고 사례 1과 같이 `layoutGenerator`가 표시되어 있다면 위의 오류가 발생할 수 있습니다.

오류: 블루프린트 Config 파일 누락

AWS Glue에서 ["레이아웃 생성기 함수 FileNotFoundError를 실행하는 동안 알 수 없는 오류 발생: [Errno 2] 해당 파일 또는 디렉터리 없음: '/tmp/compaction/blueprint.cfg'(Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory: '/tmp/compaction/blueprint.cfg')"]라는 오류를 반환합니다.

blueprint.cfg는 ZIP 아카이브의 루트 수준이나 ZIP 아카이브와 같은 이름의 폴더 내에 있어야 합니다.

블루프린트 ZIP 아카이브를 추출할 때 blueprint.cfg가 다음 경로 중 하나에 있어야 합니다. 다음 경로 중 하나에서 없으면 위의 오류가 발생할 수 있습니다.

```
$ unzip compaction.zip
Archive:  compaction.zip
   creating: compaction/
  inflating: compaction/blueprint.cfg

$ unzip compaction.zip
Archive:  compaction.zip
  inflating: blueprint.cfg
```

오류: 가져온 파일 누락

AWS Glue에서 ["레이아웃 생성기 함수 FileNotFoundError를 실행하는 동안 알 수 없는 오류 발생: [Errno 2] 해당 파일 또는 디렉터리 없음: * *'demo-project/foo.py'(Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory: * *'demo-project/foo.py')"]이라는 오류를 반환합니다.

레이아웃 생성 스크립트에 다른 파일을 읽을 수 있는 기능이 있는 경우 가져올 파일의 전체 경로를 지정해야 합니다. 예를 들어 Conversion.py 스크립트는 Layout.py에서 참조될 수 있습니다. 자세한 정보는 샘플 블루프린트 프로젝트를 참조하세요.

오류: 리소스에서 iamPassRole을 수행할 권한이 없음

AWS Glue에서 ["사용자: arn:aws:sts::123456789012:assumed-role/AWSGlueServiceRole/GlueSession에게 리소스 arn:aws:iam::123456789012:role/AWSGlueServiceRole에서 iam:PassRole을 수행할 권한이 부여되지 않음(User: arn:aws:sts::123456789012:assumed-role/

AWSGlueServiceRole/GlueSession is not authorized to perform: iam:PassRole on resource: arn:aws:iam::123456789012:role/AWSGlueServiceRole)"]이라는 오류를 반환합니다.

워크플로의 작업 및 크롤러가 블루프린트에서 워크플로를 생성하기 위해 전달된 역할과 동일한 역할을 수임하는 경우 블루프린트 역할 자체에 `iam:PassRole` 권한이 포함되어야 합니다.

워크플로의 작업 및 크롤러가 블루프린트에서 워크플로 엔터티를 생성하기 위해 전달된 역할과 다른 역할을 수임하는 경우 블루프린트 역할에 블루프린트 역할 대신 수임한 다른 역할에 대한 `iam:PassRole` 권한이 포함되어야 합니다.

자세한 정보는 블루프린트 역할에 대한 권한을 참조하세요.

오류: 잘못된 cron 일정

AWS Glue에서 ["일정 cron(0 0 * * * *)이 잘못되었습니다.(The schedule cron(0 0 * * * *) is invalid.)"]라는 오류를 반환합니다.

올바른 cron 표현식을 제공합니다. 자세한 내용은 크롤러와 작업을 위한 시간 기반 일정을 참조하십시오.

오류: 같은 이름의 트리거가 이미 있음

AWS Glue에서 ["오류: 같은 이름의 트리거가 이미 있음(Error: a trigger with the same name already exists)"]이라는 오류를 반환합니다.

블루프린트에서는 워크플로 생성을 위해 레이아웃 스크립트에서 트리거를 정의할 필요가 없습니다. 트리거 생성은 두 작업 간에 정의된 종속성을 기반으로 블루프린트 라이브러리에서 관리합니다.

트리거의 이름은 다음과 같습니다.

- 워크플로의 시작 트리거의 경우 이름은 <workflow_name>_starting_trigger입니다.
- 하나 또는 여러 업스트림 노드의 완료에 종속되는 워크플로의 노드(작업/크롤러)의 경우. AWS Glue는 이름이 <workflow_name>_<node_name>_trigger인 트리거를 정의합니다.

이 오류는 동일한 이름의 트리거가 이미 존재함을 의미합니다. 기존 트리거를 삭제하고 워크플로 생성을 다시 실행할 수 있습니다.

> **Note**
>
> 워크플로를 삭제해도 워크플로 내의 노드는 삭제되지 않습니다. 워크플로가 삭제되더라도 트리거가 남아 있을 수 있습니다. 이로 인해 '워크플로가 이미 존재함' 오류가 발생하지 않을 수

있지만 워크플로를 생성하고 삭제한 다음 동일한 블루프린트에서 동일한 이름으로 다시 생성하려고 하는 경우 '트리거가 이미 존재함' 오류가 발생할 수 있습니다.

오류: 이름이 foo인 워크플로가 이미 있습니다.

워크플로 이름은 고유해야 합니다. 다른 이름으로 시도합니다.

오류: 지정된 layoutGenerator 경로에서 모듈을 찾을 수 없음

AWS Glue에서 "레이아웃 생성기 함수 ModuleNotFoundError를 실행하는 동안 알 수 없는 오류 발생: 'crawl_s3_locations'라는 모듈 없음(Unknown error executing layout generator function ModuleNotFoundError: No module named 'crawl_s3_locations')"이라는 오류를 반환합니다.

```
layoutGenerator": "crawl_s3_locations.layout.generate_layout"
```

예를 들어 위의 layoutGenerator 경로가 있는 경우 블루프린트 아카이브의 압축을 풀면 다음과 같아야 합니다.

```
$ unzip crawl_s3_locations.zip
Archive:  crawl_s3_locations.zip
   creating: crawl_s3_locations/
  inflating: crawl_s3_locations/blueprint.cfg
  inflating: crawl_s3_locations/layout.py
  inflating: crawl_s3_locations/README.md
```

아카이브의 압축을 풀 때 블루프린트 아카이브가 다음과 같으면 위와 같은 오류가 발생할 수 있습니다.

```
$ unzip crawl_s3_locations.zip
Archive:  crawl_s3_locations.zip
  inflating: blueprint.cfg
  inflating: layout.py
  inflating: README.md
```

`crawl_s3_locations`라는 폴더가 없는 것을 볼 수 있으며 `layoutGenerator` 경로가 모듈 `crawl_s3_locations`를 통해 레이아웃 파일을 참조할 때 위의 오류가 발생할 수 있습니다.

오류: 연결 필드에 검증 오류

AWS Glue에서 "레이아웃 생성기 함수 TypeError를 실행하는 동안 알 수 없는 오류 발생: 키 연결에 대한 값 ['foo']는 <class 'dict'> 유형이어야 함!(Unknown error executing layout generator function

TypeError: Value ['foo'] for key Connections should be of type <class 'dict'>!)"라는 오류가 발생했습니다.

검증 오류입니다. `Job` 클래스의 `Connections` 필드에 딕셔너리가 필요하고 대신 값 목록이 제공되어 오류가 발생합니다.

```
User input was list of values
Connections= ['string']

Should be a dict like the following
Connections*=*{'Connections': ['string']}
```

블루프린트에서 워크플로를 생성하는 동안 이러한 런타임 오류를 방지하려면 블루프린트 테스트에 설명된 대로 워크플로, 작업 및 크롤러 정의를 검증할 수 있습니다.

레이아웃 스크립트에서 AWS Glue 작업, 크롤러 및 워크플로를 정의하려면 AWS Glue 블루프린트 클래스 참조의 구문을 참조하세요.

AWS Glue 블루프린트에 대한 페르소나 및 역할의 권한

다음은 일반적인 페르소나 및 AWS Glue 블루프린트에 대한 페르소나와 역할에 대해 제안된 AWS Identity and Access Management(IAM) 권한 정책입니다.

주제

- 블루프린트 페르소나
- 블루프린트 페르소나에 대한 권한
- 블루프린트 역할에 대한 권한

블루프린트 페르소나

다음은 일반적으로 AWS Glue 블루프린트의 수명 주기와 관련된 페르소나입니다.

페르소나	설명
AWS Glue 개발자	블루프린트를 개발, 테스트 및 게시합니다.
AWS Glue 관리자	블루프린트에 대한 권한을 등록, 유지 관리 및 부여합니다.
데이터 분석가	블루프린트를 실행하여 워크플로를 생성합니다.

자세한 내용은 the section called “블루프린트 개요” 단원을 참조하십시오.

블루프린트 페르소나에 대한 권한

다음은 각 블루프린트 페르소나에 대해 제안된 권한입니다.

블루프린트에 대한 AWS Glue 개발자 권한

AWS Glue 개발자는 블루프린트를 게시하는 데 사용되는 Amazon S3 버킷에 대한 쓰기 권한이 있어야 합니다. 종종 개발자는 블루프린트를 업로드한 후 등록합니다. 이 경우 개발자는 the section called “블루프린트에 대한 AWS Glue 관리자 권한”에 나열된 권한이 필요합니다. 또한 개발자가 블루프린트를 등록한 후 테스트하려면 the section called “블루프린트에 대한 데이터 분석가 권한”에 나열된 권한도 필요합니다.

블루프린트에 대한 AWS Glue 관리자 권한

다음 정책은 AWS Glue 블루프린트를 등록하고, 보고, 유지 관리할 수 있는 권한을 부여합니다.

> **Important**
>
> 다음 정책에서 *<s3-bucket-name>*과 *<prefix>*를 등록할 업로드된 블루프린트 ZIP 아카이브의 Amazon S3 경로로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateBlueprint",
                "glue:UpdateBlueprint",
                "glue:DeleteBlueprint",
                "glue:GetBlueprint",
                "glue:ListBlueprints",
                "glue:BatchGetBlueprints"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::<s3-bucket-name>/<prefix>/*"
        }
    ]
}
```

블루프린트에 대한 데이터 분석가 권한

다음 정책은 블루프린트를 실행하고 결과 워크플로 및 워크플로 구성 요소를 볼 수 있는 권한을 부여합니다. 또한 AWS Glue가 워크플로와 워크플로 구성 요소를 생성하기 위해 맡는 역할에 PassRole을 부여합니다.

이 정책은 모든 리소스에 대한 권한을 부여합니다. 개별 블루프린트에 대한 세분화된 액세스를 구성하려면 블루프린트 ARN에 다음 포맷을 사용합니다.

```
arn:aws:glue:<region>:<account-id>:blueprint/<blueprint-name>
```

> **Important**
>
> 다음 정책에서 *<account-id>*를 유효한 AWS 계정으로 바꾸고 *<role-name>*을 블루프린트를 실행하는 데 사용되는 역할의 이름으로 바꿉니다. 이 역할에 필요한 권한은 the section called “블루프린트 역할에 대한 권한” 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListBlueprints",
                "glue:GetBlueprint",
                "glue:StartBlueprintRun",
                "glue:GetBlueprintRun",
                "glue:GetBlueprintRuns",
                "glue:GetCrawler",
                "glue:ListTriggers",
```

```
                "glue:ListJobs",
                "glue:BatchGetCrawlers",
                "glue:GetTrigger",
                "glue:BatchGetWorkflows",
                "glue:BatchGetTriggers",
                "glue:BatchGetJobs",
                "glue:BatchGetBlueprints",
                "glue:GetWorkflowRun",
                "glue:GetWorkflowRuns",
                "glue:ListCrawlers",
                "glue:ListWorkflows",
                "glue:GetJob",
                "glue:GetWorkflow",
                "glue:StartWorkflowRun"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<account-id>:role/<role-name>"
        }
    ]
}
```

블루프린트 역할에 대한 권한

다음은 블루프린트에서 워크플로를 생성하는 데 사용되는 IAM 역할에 대해 제안된 권한입니다. 역할이 `glue.amazonaws.com`과 신뢰 관계를 맺고 있어야 합니다.

> **Important**
>
> 다음 정책에서 *<account-id>*를 유효한 AWS 계정으로 바꾸고 *<role-name>*을 역할의 이름으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "glue:CreateJob",
                "glue:GetCrawler",
                "glue:GetTrigger",
                "glue:DeleteCrawler",
                "glue:CreateTrigger",
                "glue:DeleteTrigger",
                "glue:DeleteJob",
                "glue:CreateWorkflow",
                "glue:DeleteWorkflow",
                "glue:GetJob",
                "glue:GetWorkflow",
                "glue:CreateCrawler"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<account-id>:role/<role-name>"
        }
    ]
}
```

> **Note**
>
> 워크플로의 작업 및 크롤러가 이 역할이 아닌 다른 역할을 수임하는 경우 이 정책에는 블루프린트 역할 대신 다른 역할에 대한 `iam:PassRole` 권한이 포함되어야 합니다.

AWS Glue에서 블루프린트 개발

조직에는 단일 워크플로를 파라미터화하여 모두 처리할 수 있는 이점을 얻을 수 있는 유사한 ETL 사용 사례 집합이 있을 수 있습니다. 이 요구를 해결하기 위해 AWS Glue에서 워크플로를 생성하는 데 사용할 수 있는 블루프린트를 정의할 수 있습니다. 블루프린트는 파라미터를 허용하므로 단일 블루프린트에서 데이터 분석가가 유사한 ETL 사용 사례를 처리하기 위해 다양한 워크플로를 생성할 수 있습니다. 블루프린트를 생성한 후에는 다른 부서, 팀 및 프로젝트에 재사용할 수 있습니다.

주제

- AWS Glue의 블루프린트 개요
- AWS Glue에서 블루프린트 개발

- AWS Glue의 블루프린트 등록
- AWS Glue에서 블루프린트 보기
- AWS Glue에서 블루프린트 업데이트
- AWS Glue의 블루프린트에서 워크플로 생성
- AWS Glue에서 블루프린트 실행 보기

AWS Glue의 블루프린트 개요

> **Note**
>
> 청사진 기능은 현재 아시아 태평양(자카르타) 리전 및 중동(UAE) 리전에서 AWS Glue 콘솔을 통해 사용할 수 없습니다.

AWS Glue 블루프린트는 AWS Glue 워크플로를 생성하고 공유하는 방법을 제공합니다. 유사한 사용 사례에 사용할 수 있는 복잡한 ETL 프로세스가 있는 경우 각 사용 사례에 대한 AWS Glue 워크플로를 생성하는 대신 단일 블루프린트를 생성할 수 있습니다.

블루프린트는 워크플로에 포함할 작업 및 크롤러를 지정하고 워크플로 사용자가 블루프린트를 실행하여 워크플로를 생성할 때 제공하는 파라미터를 지정합니다. 파라미터를 사용하면 단일 블루프린트에서 유사한 다양한 사용 사례에 대한 워크플로를 생성할 수 있습니다. 워크플로에 대한 자세한 내용은 AWS Glue의 워크플로 개요 섹션을 참조하세요.

다음은 블루프린트의 사용 사례 예입니다.

- 기존 데이터 집합을 분할하려고 합니다. 블루프린트의 입력 파라미터는 Amazon Simple Storage Service(Amazon S3) 소스 및 대상 경로와 파티션 열 목록입니다.
- Amazon DynamoDB 테이블의 스냅샷을 Amazon Redshift와 같은 SQL 데이터 스토어로 생성하려고 합니다. 블루프린트에 대한 입력 파라미터는 DynamoDB 테이블 이름과 Amazon Redshift 클러스터 및 대상 데이터베이스를 지정하는 AWS Glue 연결입니다.
- 여러 Amazon S3 경로의 CSV 데이터를 Parquet로 변환하려고 합니다. AWS Glue 워크플로에 각 경로에 대해 별도의 크롤러와 작업을 포함하려고 합니다. 입력 파라미터는 AWS Glue Data Catalog의 대상 데이터베이스와 쉼표로 구분된 Amazon S3 경로 목록입니다. 이 경우 워크플로가 생성하는 크롤러 및 작업의 수는 가변적입니다.

블루프린트 구성 요소

블루프린트는 다음과 같은 구성 요소를 포함하는 ZIP 아카이브 파일입니다.

- Python 레이아웃 생성기 스크립트

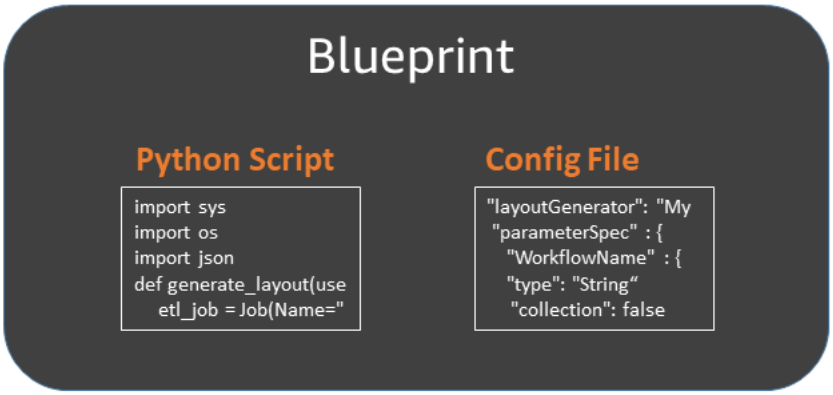
  워크플로 레이아웃, 즉 워크플로에 대해 생성할 크롤러 및 작업, 작업 및 크롤러 속성, 작업과 크롤러 간의 종속성을 지정하는 함수를 포함합니다. 이 함수는 블루프린트 파라미터를 허용하고 AWS Glue가 워크플로를 생성하는 데 사용하는 워크플로 구조(JSON 객체)를 반환합니다. Python 스크립트를 사용하여 워크플로를 생성하기 때문에 사용 사례에 적합한 고유한 논리를 추가할 수 있습니다.

- 구성 파일

  워크플로 레이아웃을 생성하는 Python 함수의 정규화된 이름을 지정합니다. 또한 스크립트에서 사용하는 모든 블루프린트 파라미터의 이름, 데이터 유형 및 기타 속성을 지정합니다.

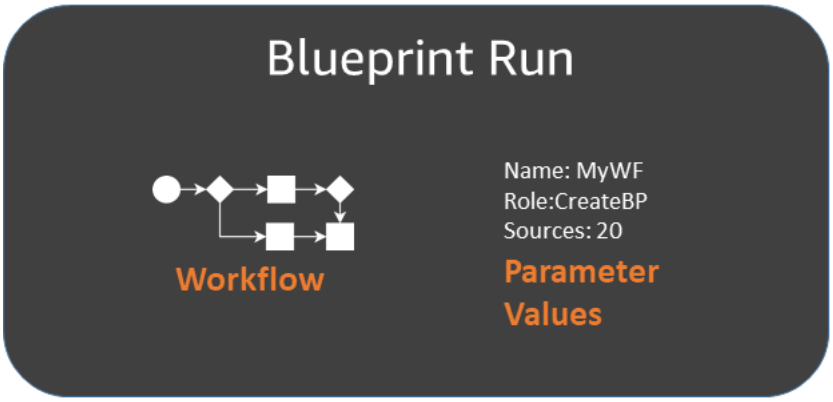
- (선택 사항) ETL 스크립트 및 지원 파일

  고급 사용 사례로 작업에서 사용하는 ETL 스크립트의 위치를 파라미터화할 수 있습니다. ZIP 아카이브에 작업 스크립트 파일을 포함하고 스크립트를 복사할 Amazon S3 위치에 대한 블루프린트 파라미터를 지정할 수 있습니다. 레이아웃 생성기 스크립트는 ETL 스크립트를 지정된 위치에 복사하고 해당 위치를 작업 스크립트 위치 속성으로 지정할 수 있습니다. 스크립트에서 처리하는 경우 라이브러리 또는 기타 지원 파일을 포함할 수도 있습니다.



블루프린트 실행

블루프린트에서 워크플로를 생성하는 경우 AWS Glue가 블루프린트를 실행하여 워크플로와 워크플로가 캡슐화하는 작업, 크롤러, 트리거를 생성하는 비동기 프로세스를 시작합니다. AWS Glue는 블루프린트 실행을 사용하여 워크플로 및 해당 구성 요소의 생성을 오케스트레이션합니다. 블루프린트 실행 상태를 확인하여 생성 프로세스의 상태를 봅니다. 블루프린트 실행에는 블루프린트 파라미터에 대해 제공한 값도 저장됩니다.



AWS Glue 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 블루프린트 실행을 볼 수 있습니다. 워크플로를 보거나 문제를 해결할 때 언제든지 블루프린트 실행으로 돌아가 워크플로를 생성하는 데 사용된 블루프린트 파라미터 값을 볼 수 있습니다.

블루프린트의 수명 주기

블루프린트는 AWS Glue로 개발, 테스트, 등록되고 워크플로를 생성하기 위해 실행됩니다. 일반적으로 블루프린트 수명 주기에는 세 가지 페르소나가 포함됩니다.

페르소나	Tasks
AWS Glue 개발자	• 워크플로 레이아웃 스크립트를 작성하고 구성 파일을 생성합니다. • AWS Glue 서비스에서 제공하는 라이브러리를 사용하여 로컬에서 블루프린트를 테스트합니다. • 스크립트, 구성 파일 및 지원 파일의 ZIP 아카이브를 생성하고 아카이브를 Amazon S3의 위치에 게시합니다. • AWS Glue 관리자의 AWS 계정에 버킷 객체에 대한 읽기 권한을 부여하는 버킷 정책을 Amazon S3 버킷에 추가합니다. • Amazon S3의 ZIP 아카이브에 대한 IAM 읽기 권한을 AWS Glue 관리자에게 부여합니다.
AWS Glue 관리자	• 블루프린트를 AWS Glue에 등록합니다. AWS Glue에서 ZIP 아카이브의 복사본을 예약된 Amazon S3 위치에 만듭니다.

페르소나	Tasks
	• 데이터 분석가에게 블루프린트에 대한 IAM 권한을 부여합니다.
데이터 분석가	• 블루프린트를 실행하여 워크플로를 생성하고 블루프린트 파라미터 값을 제공합니다. 블루프린트 실행 상태를 확인하여 워크플로 및 워크플로 구성 요소가 성공적으로 생성되었는지 확인합니다. • 워크플로를 실행하고 문제를 해결합니다. 워크플로를 실행하기 전에 AWS Glue 콘솔에서 워크플로 설계 그래프를 보고 워크플로를 확인할 수 있습니다.

다음 사항도 참조하세요.

- AWS Glue에서 블루프린트 개발
- AWS Glue의 블루프린트에서 워크플로 생성
- AWS Glue 블루프린트에 대한 페르소나 및 역할의 권한

AWS Glue에서 블루프린트 개발

AWS Glue 개발자는 데이터 분석가가 워크플로를 생성하는 데 사용할 수 있는 블루프린트를 생성하고 게시할 수 있습니다.

주제

- 블루프린트 개발 개요
- 블루프린트 개발을 위한 사전 조건
- 블루프린트 코드 작성
- 샘플 블루프린트 프로젝트
- 블루프린트 테스트
- 블루프린트 게시
- AWS Glue 블루프린트 클래스 참조
- 블루프린트 샘플

다음 사항도 참조하세요.

- AWS Glue의 블루프린트 개요

블루프린트 개발 개요

개발 프로세스의 첫 번째 단계는 블루프린트의 이점을 얻을 수 있는 일반적인 사용 사례를 파악하는 것입니다. 일반적인 사용 사례에는 일반적인 방식으로 해결해야 한다고 생각하는 반복적인 ETL 문제가 포함됩니다. 다음으로 일반화된 사용 사례를 구현하는 블루프린트를 설계하고 일반화된 사용 사례에서 특정 사용 사례를 정의할 수 있는 블루프린트 입력 파라미터를 정의합니다.

블루프린트는 블루프린트 파라미터 구성 파일이 포함된 프로젝트와 생성할 워크플로의 레이아웃을 정의하는 스크립트로 구성됩니다. 레이아웃은 생성할 작업 및 크롤러(또는 블루프린트 스크립트 용어의 엔터티)를 정의합니다.

레이아웃 스크립트에서 트리거를 직접 지정하지 않습니다. 대신 스크립트가 생성하는 작업과 크롤러 간의 종속성을 지정하는 코드를 작성합니다. AWS Glue에서는 종속성 사양에 따라 트리거를 생성합니다. 레이아웃 스크립트의 출력은 모든 워크플로 엔터티에 대한 사양을 포함하는 워크플로 객체입니다.

다음 AWS Glue 블루프린트 라이브러리를 사용하여 워크플로 객체를 구축합니다.

- `awsglue.blueprint.base_resource` - 라이브러리에서 사용하는 기본 리소스의 라이브러리입니다.
- `awsglue.blueprint.workflow` - `Workflow` 클래스를 정의하기 위한 라이브러리입니다.
- `awsglue.blueprint.job` - `Job` 클래스를 정의하기 위한 라이브러리입니다.
- `awsglue.blueprint.crawler` - `Crawler` 클래스를 정의하기 위한 라이브러리입니다.

레이아웃 생성을 위해 지원되는 유일한 다른 라이브러리는 Python 셸에 사용할 수 있는 라이브러리입니다.

블루프린트를 게시하기 전에 블루프린트 라이브러리에 정의된 방법을 사용하여 블루프린트를 로컬에서 테스트할 수 있습니다.

데이터 분석가에게 블루프린트를 제공할 준비가 되면 스크립트, 파라미터 구성 파일, 지원 파일(예: 추가 스크립트 및 라이브러리)을 배포 가능한 단일 자산으로 패키징합니다. 그런 다음 자산을 Amazon S3에 업로드하고 관리자에게 AWS Glue에 등록하도록 요청합니다.

추가 샘플 블루프린트 프로젝트에 대한 자세한 내용은 샘플 블루프린트 프로젝트 및 블루프린트 샘플 섹션을 참조하세요.

블루프린트 개발을 위한 사전 조건

블루프린트를 개발하려면 AWS Glue 사용과 Apache Spark ETL 작업 또는 Python 셸 작업을 위한 스크립트 작성에 익숙해야 합니다. 또한 다음 설정 태스크를 완료해야 합니다.

- 블루프린트 레이아웃 스크립트에 사용할 4개의 AWS Python 라이브러리를 다운로드합니다.
- AWS SDK를 설정합니다.
- AWS CLI를 설정합니다.

Python 라이브러리 다운로드

GitHub에서 다음 라이브러리를 다운로드하고 프로젝트에 설치합니다.

- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/base_resource.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/workflow.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/crawler.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/job.py

AWS Java SDK 설정

AWS Java SDK의 경우 블루프린트용 API가 `jar` 포함된 파일을 추가해야 합니다.

1. 아직 설정하지 않은 경우 AWS SDK for Java를 설정합니다.
   - Java 1.x의 경우 AWS SDK for Java Developer Guide의 Set up the AWS SDK for Java 지침을 따르세요.
   - Java 2.x의 경우 AWS SDK for Java 2.x Developer Guide의 Setting up the AWS SDK for Java 2.x 지침을 따르세요.
2. 블루프린트용 API에 대한 액세스 권한이 있는 클라이언트 `jar` 파일을 다운로드합니다.
   - Java 1.x의 경우: s3://awsglue-custom-blueprints-preview-artifacts/awsglue-java-sdk-preview/AWSGlueJavaClient-1.11.x.jar
   - Java 2.x의 경우: s3://awsglue-custom-blueprints-preview-artifacts/awsglue-java-sdk-v2-preview/AwsJavaSdk-Glue-2.0.jar

3. AWS Java SDK에서 제공하는 AWS Glue 클라이언트를 재정의하려면 Java 클래스 경로 앞에 클라이언트 jar를 추가합니다.

```
export CLASSPATH=<path-to-preview-client-jar>:$CLASSPATH
```

4. (선택 사항) 다음 Java 애플리케이션으로 SDK를 테스트합니다. 애플리케이션이 빈 목록을 출력해야 합니다.

   accessKey 및 secretKey를 자격 증명으로 바꾸고 us-east-1을 리전으로 바꿉니다.

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.glue.AWSGlue;
import com.amazonaws.services.glue.AWSGlueClientBuilder;
import com.amazonaws.services.glue.model.ListBlueprintsRequest;

public class App{
    public static void main(String[] args) {
        AWSCredentials credentials = new BasicAWSCredentials("accessKey",
 "secretKey");
        AWSCredentialsProvider provider = new
 AWSStaticCredentialsProvider(credentials);
        AWSGlue glue = AWSGlueClientBuilder.standard().withCredentials(provider)
                .withRegion("us-east-1").build();
        ListBlueprintsRequest request = new
 ListBlueprintsRequest().withMaxResults(2);
        System.out.println(glue.listBlueprints(request));
    }
}
```

AWS Python SDK 설정

다음 단계에서는 컴퓨터에 Python 버전 2.7 이상 또는 버전 3.9 이상이 설치되어 있다고 가정합니다.

1. 다음 boto3 휠 파일을 다운로드합니다. 열거나 저장하라는 메시지가 표시되면 s3://awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/boto3-1.17.31-py2.py3-none-any.whl 파일을 저장합니다.
2. botocore 휠 파일인 s3://awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/botocore-1.20.31-py2.py3-none-any.whl을 다운로드합니다.

3. Python 버전을 확인합니다.

```
python --version
```

4. Python 버전에 따라 다음 명령을 입력합니다(Linux의 경우).

   - Python 2.7 이상용입니다.

```
python3 -m pip install --user virtualenv
source env/bin/activate
```

   - Python 3.9 이상용입니다.

```
python3 -m venv python-sdk-test
source python-sdk-test/bin/activate
```

5. botocore 휠 파일을 설치합니다.

```
python3 -m pip install <download-directory>/botocore-1.20.31-py2.py3-none-any.whl
```

6. boto3 휠 파일을 설치합니다.

```
python3 -m pip install <download-directory>/boto3-1.17.31-py2.py3-none-any.whl
```

7. `~/.aws/credentials` 및 `~/.aws/config` 파일에서 자격 증명과 기본 리전을 구성합니다. 자세한 내용은 AWS Command Line Interface User Guide의 Configuring the AWS CLI를 참조하세요.

8. (선택 사항) 설정을 테스트합니다. 다음 명령은 빈 목록을 반환합니다.

   `us-east-1`을 해당 리전으로 바꿉니다.

```
$ python
>>> import boto3
>>> glue = boto3.client('glue', 'us-east-1')
>>> glue.list_blueprints()
```

미리 보기 AWS CLI 설정

1. 아직 수행하지 않았다면 컴퓨터에 AWS Command Line Interface(AWS CLI)를 설치 및/또는 업데이트합니다. 이를 수행하는 가장 쉬운 방법은 Python 설치 프로그램 유틸리티인 `pip`를 사용하는 것입니다.

   ```
   pip install awscli --upgrade --user
   ```

   AWS CLI에 대한 전체 설치 지침은 AWS Command Line Interface 설치에서 찾을 수 있습니다.
2. s3://awsglue-custom-blueprints-preview-artifacts/awscli-preview-build/awscli-1.19.31-py2.py3-none-any.whl에서 AWS CLI 휠 파일을 다운로드합니다.
3. AWS CLI 휠 파일을 설치합니다.

   ```
   python3 -m pip install awscli-1.19.31-py2.py3-none-any.whl
   ```
4. `aws configure` 명령을 실행합니다. AWS 자격 증명(액세스 키 및 보안 암호 키 포함) 및 AWS 리전을 구성합니다. AWS CLI 구성에 대한 자세한 내용은 AWS CLI 구성에서 찾을 수 있습니다.
5. AWS CLI를 테스트합니다. 다음 명령은 빈 목록을 반환합니다.

   `us-east-1`을 해당 리전으로 바꿉니다.

   ```
   aws glue list-blueprints --region us-east-1
   ```

블루프린트 코드 작성

생성하는 각 블루프린트 프로젝트에는 최소한 다음 파일이 포함되어야 합니다.

- 워크플로를 정의하는 Python 레이아웃 스크립트. 스크립트에는 워크플로의 엔터티(작업 및 크롤러)와 이들 간의 종속성을 정의하는 기능이 포함되어 있습니다.
- 다음을 정의하는 구성 파일, `blueprint.cfg`
  - 워크플로 레이아웃 정의 기능의 전체 경로.
  - 블루프린트에서 허용하는 파라미터.

주제

- 블루프린트 레이아웃 스크립트 생성
- 구성 파일 생성

- 블루프린트 파라미터 지정

블루프린트 레이아웃 스크립트 생성

블루프린트 레이아웃 스크립트에는 워크플로에서 엔터티를 생성하는 함수가 포함되어야 합니다. 원하는 대로 이 함수의 이름을 지정할 수 있습니다. AWS Glue에서는 구성 파일을 사용하여 함수의 정규화된 이름을 확인합니다.

레이아웃 함수는 다음을 수행합니다.

- (선택 사항) `Job` 클래스를 인스턴스화하여 `Job` 객체를 생성하고 `Command` 및 `Role` 등의 인수를 전달합니다. AWS Glue 콘솔 또는 API를 사용하여 작업을 생성하는 경우 지정할 작업 속성입니다.
- (선택 사항) `Crawler` 클래스를 인스턴스화하여 `Crawler` 객체를 생성하고 이름, 역할 및 대상 인수를 전달합니다.
- 객체(워크플로 엔터티) 간의 종속성을 나타내려면 `DependsOn` 및 `WaitForDependencies` 추가 인수를 `Job()` 및 `Crawler()`에 전달합니다. 이러한 인수는 이 섹션의 뒷부분에서 설명합니다.
- `Workflow` 클래스를 인스턴스화하여 AWS Glue에 반환되는 워크플로 객체를 생성하고 `Name` 인수, `Entities` 인수, 선택적 `OnSchedule` 인수를 전달합니다. 이 `Entities` 인수는 워크플로에 포함할 모든 작업 및 크롤러를 지정합니다. `Entities` 객체를 구성하는 방법을 보려면 이 섹션의 뒷부분에 나오는 샘플 프로젝트를 참조하세요.
- `Workflow` 객체를 반환합니다.

`Job`, `Crawler` 및 `Workflow` 클래스의 정의는 AWS Glue 블루프린트 클래스 참조 섹션을 참조하세요.

레이아웃 함수에 사용할 수 있는 입력 인수는 다음과 같습니다.

인수	설명
`user_params`	블루프린트 파라미터 이름 및 값의 Python 딕셔너리. 자세한 내용은 블루프린트 파라미터 지정 단원을 참조하십시오.
`system_params`	두 가지 속성(`region` 및 `accountId`)을 포함하는 Python 딕셔너리.

다음은 `Layout.py`라는 파일의 샘플 레이아웃 생성기 스크립트입니다.

```
import argparse
```

```
import sys
import os
import json
from awsglue.blueprint.workflow import *
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *


def generate_layout(user_params, system_params):

    etl_job = Job(Name="{}_etl_job".format(user_params['WorkflowName']),
                  Command={
                      "Name": "glueetl",
                      "ScriptLocation": user_params['ScriptLocation'],
                      "PythonVersion": "2"
                  },
                  Role=user_params['PassRole'])
    post_process_job = Job(Name="{}_post_process".format(user_params['WorkflowName']),
                           Command={
                               "Name": "pythonshell",
                               "ScriptLocation": user_params['ScriptLocation'],
                               "PythonVersion": "2"
                           },
                           Role=user_params['PassRole'],
                           DependsOn={
                               etl_job: "SUCCEEDED"
                           },
                           WaitForDependencies="AND")
    sample_workflow = Workflow(Name=user_params['WorkflowName'],
                           Entities=Entities(Jobs=[etl_job, post_process_job]))
    return sample_workflow
```

샘플 스크립트는 필요한 Blueprint 라이브러리를 가져오고 2개의 작업으로 워크플로를 생성하는 `generate_layout` 함수를 포함합니다. 이것은 매우 간단한 스크립트입니다. 보다 복잡한 스크립트는 추가 로직 및 파라미터를 사용하여 많은 작업과 크롤러 또는 다양한 수의 작업과 크롤러가 있는 워크플로를 생성할 수 있습니다.

DependsOn 인수 사용

`DependsOn` 인수는 이 엔터티가 워크플로 내의 다른 엔터티에 대해 갖는 종속성의 딕셔너리 표현입니다. 형식은 다음과 같습니다.

```
DependsOn = {dependency1 : state, dependency2 : state, ...}
```

이 딕셔너리의 키는 엔터티의 이름이 아닌 객체 참조를 나타내는 반면 값은 감시할 상태에 해당하는 문자열입니다. AWS Glue는 적절한 트리거를 유추합니다. 유효한 상태는 조건 구조를 참조하세요.

예를 들어 작업은 크롤러의 성공적인 완료에 따라 달라질 수 있습니다. 다음과 같이 이름이 `crawler2`인 크롤러 객체를 정의하는 경우

```
crawler2 = Crawler(Name="my_crawler", ...)
```

그러면 `crawler2`에 종속된 객체에는 다음과 같은 생성자 인수가 포함됩니다.

```
DependsOn = {crawler2 : "SUCCEEDED"}
```

예:

```
job1 = Job(Name="Job1", ..., DependsOn = {crawler2 : "SUCCEEDED", ...})
```

엔터티에 대해 `DependsOn`이 생략된 경우 해당 엔터티는 워크플로 시작 트리거에 따라 달라집니다.

WaitForDependencies 인수 사용

`WaitForDependencies` 인수는 작업 또는 크롤러 엔터티가 종속된 모든 엔터티가 완료될 때까지 또는 하나라도완료될 때까지 기다려야 하는지 여부를 정의합니다.

허용되는 값은 "AND" 또는 "ANY"입니다.

OnSchedule 인수 사용

`Workflow` 클래스 생성자의 `OnSchedule` 인수는 워크플로에 대한 시작 트리거 정의를 정의하는 `cron` 표현식입니다.

이 인수가 지정되면 AWS Glue는 해당 일정으로 일정 트리거를 생성합니다. 지정되지 않은 경우 워크플로의 시작 트리거는 온디맨드 트리거입니다.

구성 파일 생성

블루프린트 구성 파일은 워크플로 생성을 위한 스크립트 진입점과 블루프린트에서 허용하는 파라미터를 정의하는 필수 파일입니다. 파일의 이름은 `blueprint.cfg`여야 합니다.

다음은 샘플 구성 파일입니다.

```
{
    "layoutGenerator": "DemoBlueprintProject.Layout.generate_layout",
    "parameterSpec" : {
           "WorkflowName" : {
                "type": "String",
                "collection": false
           },
           "WorkerType" : {
                "type": "String",
                "collection": false,
                "allowedValues": ["G1.X", "G2.X"],
                "defaultValue": "G1.X"
           },
           "Dpu" : {
                "type" : "Integer",
                "allowedValues" : [2, 4, 6],
                "defaultValue" : 2
           },
           "DynamoDBTableName": {
                "type": "String",
                "collection" : false
           },
           "ScriptLocation" : {
                "type": "String",
                "collection": false
     }
    }
}
```

`layoutGenerator` 속성은 레이아웃을 생성하는 스크립트에서 함수의 정규화된 이름을 지정합니다.

`parameterSpec` 속성은 이 블루프린트에서 허용하는 파라미터를 지정합니다. 자세한 내용은 블루프린트 파라미터 지정 단원을 참조하십시오.

> **Important**
>
> 구성 파일에 워크플로 이름이 블루프린트 파라미터로 포함되거나 레이아웃 스크립트에서 고유한 워크플로 이름을 생성해야 합니다.

블루프린트 파라미터 지정

구성 파일에는 parameterSpec JSON 객체의 블루프린트 파라미터 사양이 포함되어 있습니다. parameterSpec은 하나 이상의 파라미터 객체를 포함합니다.

```
"parameterSpec": {
    "<parameter_name>": {
      "type": "<parameter-type>",
      "collection": true|false,
      "description": "<parameter-description>",
      "defaultValue": "<default value for the parameter if value not specified>"
      "allowedValues": "<list of allowed values>"
    },
    "<parameter_name>": {
       ...
    }
  }
```

다음은 각 파라미터 객체를 코딩하는 규칙입니다.

- 파라미터 이름과 type은 필수입니다. 다른 모든 속성은 선택 사항입니다.
- defaultValue 속성을 지정하는 경우 파라미터는 선택 사항입니다. 그렇지 않으면 파라미터가 필수이며 블루프린트에서 워크플로를 생성하는 데이터 분석가가 해당 값을 제공해야 합니다.
- collection 속성을 true로 설정하면 파라미터가 값 컬렉션을 사용할 수 있습니다. 컬렉션은 모든 데이터 유형이 될 수 있습니다.
- allowedValues를 지정하면 AWS Glue 콘솔에 데이터 분석가가 블루프린트에서 워크플로를 생성할 때 선택할 수 있는 값의 드롭다운 목록이 표시됩니다.

다음은 type에 허용되는 값입니다.

파라미터 데이터 유형	참고
String	-
Integer	-
Double	-

파라미터 데이터 유형	참고
`Boolean`	가능한 값은 `true` 및 `false`입니다. AWS Glue 콘솔의 <블루프린트> 페이지에서 워크플로 생성(Create a workflow from <blueprint>) 페이지에 확인란을 생성합니다.
`S3Uri`	`s3://`로 시작하여 Amazon S3 경로를 완성합니다. [<블루프린트>에서 워크플로 생성(Create a workflow from <blueprint>)]에서 텍스트 필드와 [찾아보기(Browse)] 버튼을 생성합니다.
`S3Bucket`	Amazon S3 버킷 이름만입니다. <블루프린트>에서 워크플로 생성(Create a workflow from <blueprint>) 페이지에서 버킷 선택기를 생성합니다.
`IAMRoleArn`	AWS Identity and Access Management(IAM) 역할의 Amazon 리소스 이름(ARN)입니다. <블루프린트>에서 워크플로 생성(Create a workflow from <blueprint>) 페이지에서 역할 선택기를 생성합니다.
`IAMRoleName`	IAM 역할의 이름입니다. <블루프린트>에서 워크플로 생성(Create a workflow from <blueprint>) 페이지에서 역할 선택기를 생성합니다.

샘플 블루프린트 프로젝트

데이터 포맷 변환은 빈번한 추출, 변환, 로드 사용 사례입니다. 일반적인 분석 워크로드에서는 Parquet 또는 ORC와 같은 열 기반 파일 포맷이 CSV 또는 JSON과 같은 텍스트 포맷보다 선호됩니다. 이 샘플 블루프린트를 사용하면 CSV/JSON 등의 데이터를 Amazon S3의 파일용 Parquet로 변환할 수 있습니다.

이 블루프린트는 블루프린트 파라미터로 정의된 S3 경로 목록을 가져와 데이터를 Parquet 포맷으로 변환한 다음 다른 블루프린트 파라미터로 지정된 S3 위치에 씁니다. 레이아웃 스크립트는 각 경로에 대한 크롤러 및 작업을 생성합니다. 또한 레이아웃 스크립트는 `Conversion.py`의 ETL 스크립트를 다른 블루프린트 파라미터로 지정된 S3 버킷에 업로드합니다. 그런 다음 레이아웃 스크립트는 업로드된 스크립트를 각 작업에 대한 ETL 스크립트로 지정합니다. 프로젝트의 ZIP 아카이브에는 레이아웃 스크립트, ETL 스크립트 및 Blueprint 구성 파일이 포함되어 있습니다.

추가 샘플 블루프린트 프로젝트에 대한 자세한 내용은 블루프린트 샘플 섹션을 참조하세요.

다음은 `Layout.py` 파일에 있는 레이아웃 스크립트입니다.

```
from awsglue.blueprint.workflow import *
```

```
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *
import boto3

s3_client = boto3.client('s3')

# Ingesting all the S3 paths as Glue table in parquet format
def generate_layout(user_params, system_params):
    #Always give the full path for the file
    with open("ConversionBlueprint/Conversion.py", "rb") as f:
        s3_client.upload_fileobj(f, user_params['ScriptsBucket'], "Conversion.py")
    etlScriptLocation = "s3://{}/Conversion.py".format(user_params['ScriptsBucket'])

    crawlers = []
    jobs = []
    workflowName = user_params['WorkflowName']
    for path in user_params['S3Paths']:
      tablePrefix = "source_"
      crawler = Crawler(Name="{}_crawler".format(workflowName),
                        Role=user_params['PassRole'],
                        DatabaseName=user_params['TargetDatabase'],
                        TablePrefix=tablePrefix,
                        Targets= {"S3Targets": [{"Path": path}]})
      crawlers.append(crawler)
      transform_job = Job(Name="{}_transform_job".format(workflowName),
                          Command={"Name": "glueetl",
                                   "ScriptLocation": etlScriptLocation,
                                   "PythonVersion": "3"},
                          Role=user_params['PassRole'],
                          DefaultArguments={"--database_name":
 user_params['TargetDatabase'],
                                            "--table_prefix": tablePrefix,
                                            "--region_name": system_params['region'],
                                            "--output_path":
 user_params['TargetS3Location']},
                          DependsOn={crawler: "SUCCEEDED"},
                          WaitForDependencies="AND")
      jobs.append(transform_job)
    conversion_workflow = Workflow(Name=workflowName, Entities=Entities(Jobs=jobs,
 Crawlers=crawlers))
    return conversion_workflow
```

다음은 해당 블루프린트 구성 파일인 `blueprint.cfg`입니다.

```
{
    "layoutGenerator": "ConversionBlueprint.Layout.generate_layout",
    "parameterSpec" : {
        "WorkflowName" : {
            "type": "String",
            "collection": false,
            "description": "Name for the workflow."
        },
        "S3Paths" : {
            "type": "S3Uri",
            "collection": true,
            "description": "List of Amazon S3 paths for data ingestion."
        },
        "PassRole" : {
            "type": "IAMRoleName",
            "collection": false,
            "description": "Choose an IAM role to be used in running the job/crawler"
        },
        "TargetDatabase": {
            "type": "String",
            "collection" : false,
            "description": "Choose a database in the Data Catalog."
        },
        "TargetS3Location": {
            "type": "S3Uri",
            "collection" : false,
            "description": "Choose an Amazon S3 output path: ex:s3://<target_path>/."
        },
        "ScriptsBucket": {
            "type": "S3Bucket",
            "collection": false,
            "description": "Provide an S3 bucket name(in the same AWS Region) to store
 the scripts."
        }
    }
}
```

Conversion.py 파일의 다음 스크립트는 업로드된 ETL 스크립트입니다. 변환하는 동안 파티션 구성 표를 유지합니다.

```
import sys
from pyspark.sql.functions import *
from pyspark.context import SparkContext
```

```
from awsglue.transforms import *
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions
import boto3

args = getResolvedOptions(sys.argv, [
    'JOB_NAME',
    'region_name',
    'database_name',
    'table_prefix',
    'output_path'])
databaseName = args['database_name']
tablePrefix = args['table_prefix']
outputPath = args['output_path']

glue = boto3.client('glue', region_name=args['region_name'])

glue_context = GlueContext(SparkContext.getOrCreate())
spark = glue_context.spark_session
job = Job(glue_context)
job.init(args['JOB_NAME'], args)

def get_tables(database_name, table_prefix):
    tables = []
    paginator = glue.get_paginator('get_tables')
    for page in paginator.paginate(DatabaseName=database_name, Expression=table_prefix
+"*"):
        tables.extend(page['TableList'])
    return tables

for table in get_tables(databaseName, tablePrefix):
    tableName = table['Name']
    partitionList = table['PartitionKeys']
    partitionKeys = []
    for partition in partitionList:
        partitionKeys.append(partition['Name'])

    # Create DynamicFrame from Catalog
    dyf = glue_context.create_dynamic_frame.from_catalog(
        name_space=databaseName,
        table_name=tableName,
        additional_options={
            'useS3ListImplementation': True
```

```
        },
        transformation_ctx='dyf'
    )

    # Resolve choice type with make_struct
    dyf = ResolveChoice.apply(
        frame=dyf,
        choice='make_struct',
        transformation_ctx='resolvechoice_' + tableName
    )

    # Drop null fields
    dyf = DropNullFields.apply(
        frame=dyf,
        transformation_ctx="dropnullfields_" + tableName
    )

    # Write DynamicFrame to S3 in glueparquet
    sink = glue_context.getSink(
        connection_type="s3",
        path=outputPath,
        enableUpdateCatalog=True,
        partitionKeys=partitionKeys
    )
    sink.setFormat("glueparquet")

    sink.setCatalogInfo(
        catalogDatabase=databaseName,
        catalogTableName=tableName[len(tablePrefix):]
    )
    sink.writeFrame(dyf)

job.commit()
```

> **Note**
>
> 2개의 Amazon S3 경로만 샘플 블루프린트에 대한 입력으로 제공될 수 있습니다. 이는 AWS Glue 트리거가 2개의 크롤러 작업만 호출하도록 제한되기 때문입니다.

블루프린트 테스트

코드를 개발하는 동안 워크플로 레이아웃이 올바른지 확인하기 위해 로컬 테스트를 수행해야 합니다.

로컬 테스트는 AWS Glue 작업, 크롤러 또는 트리거를 생성하지 않습니다. 대신 레이아웃 스크립트를 로컬에서 실행하고 `to_json()` 및 `validate()` 메서드를 사용하여 객체를 인쇄하고 오류를 찾습니다. 이러한 방법은 라이브러리에 정의 된 3가지 클래스 모두에서 사용할 수 있습니다.

AWS Glue가 레이아웃 함수에 전달하는 `user_params` 및 `system_params` 인수를 처리하는 두 가지 방법이 있습니다. 테스트 벤치 코드는 샘플 블루프린트 파라미터 값의 딕셔너리를 생성하고 이를 `user_params` 인수로 레이아웃 함수에 전달할 수 있습니다. 또는 `user_params`에 대한 참조를 제거하고 하드코딩된 문자열로 바꿀 수 있습니다.

코드에서 `system_params` 인수의 `region` 및 `accountId` 속성을 사용하는 경우 `system_params`에 대한 딕셔너리를 전달할 수 있습니다.

블루프린트를 테스트하려면

1. 라이브러리가 있는 디렉터리에서 Python 인터프리터를 시작하거나 블루프린트 파일과 제공된 라이브러리를 원하는 IDE(통합 개발 환경)로 로드합니다.
2. 코드가 제공된 라이브러리를 가져 오는지 확인합니다.
3. 레이아웃 기능에 코드를 추가하여 엔터티 또는 `Workflow` 객체에서 `validate()` 또는 `to_json()`을 호출합니다. 예를 들어 코드가 `mycrawler`라는 `Crawler` 객체를 생성하는 경우 다음과 같이 `validate()`를 호출할 수 있습니다.

   ```
   mycrawler.validate()
   ```

   다음과 같이 `mycrawler`를 인쇄 할 수 있습니다.

   ```
   print(mycrawler.to_json())
   ```

   객체에서 `to_json`을 호출하면 `to_json()`이 `validate()`를 호출하기 때문에 `validate()`도 호출할 필요가 없습니다.

   워크플로 객체에서 이러한 메서드를 호출하는 것이 가장 유용합니다. 스크립트가 워크플로 객체의 이름을 `my_workflow`라고 가정하고 다음과 같이 워크플로 객체를 검증하고 인쇄합니다.

   ```
   print(my_workflow.to_json())
   ```

to_json() 및 validate()에 대한 자세한 내용은 클래스 메서드섹션을 참조하세요.

이 섹션의 뒷부분에 나오는 예제와 같이 pprint를 가져와서 워크플로 객체를 예쁘게 인쇄할 수도 있습니다.

4. 코드를 실행하고, 오류를 수정하고, 마지막으로 validate() 또는 to_json()에 대한 호출을 제거합니다.

Example

다음 예제는 샘플 블루프린트 파라미터의 딕셔너리를 구성하고 레이아웃 함수 generate_compaction_workflow에 user_params 인수로 전달하는 방법을 보여줍니다. 또한 생성된 워크플로 객체를 예쁘게 인쇄하는 방법을 보여줍니다.

```
from pprint import pprint
from awsglue.blueprint.workflow import *
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *

USER_PARAMS = {"WorkflowName": "compaction_workflow",
               "ScriptLocation": "s3://amzn-s3-demo-bucket/scripts/threaded-
compaction.py",
               "PassRole": "arn:aws:iam::111122223333:role/GlueRole-ETL",
               "DatabaseName": "cloudtrial",
               "TableName": "ct_cloudtrail",
               "CoalesceFactor": 4,
               "MaxThreadWorkers": 200}


def generate_compaction_workflow(user_params: dict, system_params: dict) -> Workflow:
    compaction_job = Job(Name=f"{user_params['WorkflowName']}_etl_job",
                         Command={"Name": "glueetl",
                                  "ScriptLocation": user_params['ScriptLocation'],
                                  "PythonVersion": "3"},
                         Role="arn:aws:iam::111122223333:role/
AWSGlueServiceRoleDefault",
                         DefaultArguments={"DatabaseName": user_params['DatabaseName'],
                                           "TableName": user_params['TableName'],
                                           "CoalesceFactor":
 user_params['CoalesceFactor'],
                                           "max_thread_workers":
 user_params['MaxThreadWorkers']})
```

```
    catalog_target = {"CatalogTargets": [{"DatabaseName": user_params['DatabaseName'],
 "Tables": [user_params['TableName']]}]}

    compacted_files_crawler = Crawler(Name=f"{user_params['WorkflowName']}_post_crawl",
                                      Targets = catalog_target,
                                      Role=user_params['PassRole'],
                                      DependsOn={compaction_job: "SUCCEEDED"},
                                      WaitForDependencies="AND",
                                      SchemaChangePolicy={"DeleteBehavior": "LOG"})

    compaction_workflow = Workflow(Name=user_params['WorkflowName'],
                                   Entities=Entities(Jobs=[compaction_job],

 Crawlers=[compacted_files_crawler]))
    return compaction_workflow

generated = generate_compaction_workflow(user_params=USER_PARAMS, system_params={})
gen_dict = generated.to_json()

pprint(gen_dict)
```

블루프린트 게시

블루프린트를 개발한 후에는 Amazon S3에 업로드해야 합니다. 블루프린트를 게시하는 데 사용하는 Amazon S3 버킷에 대한 쓰기 권한이 있어야 합니다. 또한 블루프린트를 등록할 AWS Glue 관리자에게 Amazon S3 버킷에 대한 읽기 액세스 권한이 있는지 확인해야 합니다. AWS Glue 블루프린트의 페르소나 및 역할에 대해 제안된 AWS Identity and Access Management(IAM) 권한 정책은 AWS Glue 블루프린트에 대한 페르소나 및 역할의 권한 섹션을 참조하세요.

블루프린트를 게시하려면

1. 필요한 스크립트, 리소스 및 블루프린트 구성 파일을 생성합니다.
2. 모든 파일을 ZIP 아카이브에 추가하고 ZIP 파일을 Amazon S3에 업로드합니다. 사용자가 블루프린트를 등록하고 실행할 리전과 동일한 리전에 있는 S3 버킷을 사용합니다.

   다음 명령을 사용하여 명령줄에서 ZIP 파일을 생성할 수 있습니다.

   ```
   zip -r folder.zip folder
   ```

3. 원하는 AWS 계정에 읽기 권한을 부여하는 버킷 정책을 추가합니다. 다음은 샘플 정책입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-blueprints/*"
        }
    ]
}
```

4. Amazon S3 버킷에 대한 IAM `s3:GetObject` 권한을 AWS Glue 관리자 또는 블루프린트를 등록할 사람에게 부여합니다. 관리자에게 부여할 샘플 정책은 블루프린트에 대한 AWS Glue 관리자 권한 섹션을 참조하세요.

블루프린트의 로컬 테스트를 완료한 후 AWS Glue에서 블루프린트를 테스트할 수도 있습니다. AWS Glue에서 블루프린트를 테스트하려면 등록이 필요합니다. IAM 권한 부여를 사용하거나 별도의 테스트 계정을 사용하여 등록된 블루프린트를 볼 수 있는 사람을 제한할 수 있습니다.

> 다음 사항도 참조하세요.
>
> - AWS Glue의 블루프린트 등록

AWS Glue 블루프린트 클래스 참조

AWS Glue 블루프린트용 라이브러리는 워크플로 레이아웃 스크립트에서 사용하는 세 가지 클래스인 `Job`, `Crawler` 및 `Workflow`를 정의합니다.

주제

- 작업 클래스
- 크롤러 클래스
- 워크플로 클래스
- 클래스 메서드

작업 클래스

Job 클래스는 AWS Glue ETL 작업을 나타냅니다.

필수 생성자 인수

다음은 Job 클래스에 대한 필수 생성자 인수입니다.

인수 이름	유형	설명
Name	str	작업에 할당할 이름입니다. AWS Glue는 작업을 다른 블루프린트 실행에서 생성된 작업과 구별하기 위해 이름에 임의로 생성된 접미사를 추가합니다.
Role	str	실행 중 작업이 수임해야 하는 역할의 Amazon 리소스 이름(ARN)입니다.
Command	dict	API 문서의 JobCommand 구조에 지정된 작업 명령.

선택적 생성자 인수

다음은 Job 클래스에 대한 선택적 생성자 인수입니다.

인수 이름	유형	설명
DependsOn	dict	작업이 종속된 워크플로 엔터티의 목록입니다. 자세한 내용은 DependsOn 인수 사용 단원을 참조하십시오.
WaitForDependencies	str	작업이 종속된 모든 엔터티가 실행되기 전에 완료될 때까지 기다릴지 아니면 하나라도 완료될 때까지 기다릴지 나타냅니다. 자세한 내용은 WaitForDependencies 인수 사용 단원을 참조하십시오. 작업이 하나의 엔티티에만 종속되는 경우 생략합니다.

인수 이름	유형	설명
(작업 속성)	-	AWS Glue API 설명서의 작업 구조에 나열된 모든 작업 속성(CreatedOn 및 LastModifiedOn 제외)입니다.

크롤러 클래스

Crawler 클래스는 AWS Glue 크롤러를 나타냅니다.

필수 생성자 인수

다음은 Crawler 클래스에 대한 필수 생성자 인수입니다.

인수 이름	유형	설명
Name	str	크롤러에 할당할 이름입니다. AWS Glue는 크롤러를 다른 블루프린트 실행에서 생성된 크롤러와 구별하기 위해 이름에 임의로 생성된 접미사를 추가합니다.
Role	str	크롤러가 실행 중 수임해야 하는 역할의 ARN입니다.
Targets	dict	크롤링할 대상 컬렉션입니다. Targets 클래스 생성자 인수는 API 문서의 CrawlerTargets 구조에 정의되어 있습니다. 모두 Targets 생성자 인수는 선택 사항이지만 적어도 하나는 전달해야 합니다.

선택적 생성자 인수

다음은 Crawler 클래스에 대한 선택적 생성자 인수입니다.

인수 이름	유형	설명
DependsOn	dict	크롤러가 종속된 워크플로 엔터티의 목록입니다. 자세한 내용은 DependsOn 인수 사용 단원을 참조하십시오.
WaitForDependencies	str	크롤러가 종속된 모든 엔터티가 실행되기 전에 완료될 때까지 기다릴지 아니면 하나라도 완료될 때까지 기다릴지 나타냅니다. 자세한 내용은 WaitForDependencies 인수 사용 단원을 참조하십시오. 크롤러가 하나의 엔티티에만 종속된 경우에는 생략합니다.
(크롤러 속성)	-	다음 예외사항과 함께 AWS Glue API 설명서의 크롤러 구조에 나열된 모든 크롤러 속성: • State • CrawlElapsedTime • CreationTime • LastUpdated • LastCrawl • Version

워크플로 클래스

Workflow 클래스는 AWS Glue 워크플로를 나타냅니다. 워크플로 레이아웃 스크립트는 Workflow 객체를 반환합니다. AWS Glue에서는 이 객체를 기반으로 하여 워크플로를 생성합니다.

필수 생성자 인수

다음은 Workflow 클래스에 대한 필수 생성자 인수입니다.

인수 이름	유형	설명
Name	str	워크플로에 할당할 이름입니다.

인수 이름	유형	설명
`Entities`	`Entities`	워크플로에 포함할 엔터티 (작업 및 크롤러)의 컬렉션입니다. `Entities` 클래스 생성자는 `Crawler` 객체의 목록인 `Crawlers` 인수와 `Job` 객체의 목록인 `Jobs` 인수를 허용합니다.

선택적 생성자 인수

다음은 `Workflow` 클래스에 대한 선택적 생성자 인수입니다.

인수 이름	유형	설명
`Description`	`str`	워크플루 구조 섹션을 참조하세요.
`DefaultRunProperties`	`dict`	워크플루 구조 섹션을 참조하세요.
`OnSchedule`	`str`	`cron` 표현식입니다.

클래스 메서드

세 클래스 모두 다음 메서드를 포함합니다.

validate()

객체의 속성을 검증하고 오류가 발견되면 메시지를 출력하고 종료합니다. 오류가 없으면 출력을 생성하지 않습니다. `Workflow` 클래스의 경우 워크플로의 모든 엔터티에서 자신을 호출합니다.

to_json()

객체를 JSON으로 직렬화합니다. 또한 `validate()`를 호출합니다. `Workflow` 클래스의 경우 JSON 객체에는 작업 및 크롤러 목록과 작업 및 크롤러 종속성 사양에 의해 생성된 트리거 목록이 포함됩니다.

블루프린트 샘플

AWS Glue 블루프린트 Github 리포지토리에서 사용할 수 있는 샘플 블루프린트 프로젝트가 많이 있습니다. 이 샘플은 참조용이며 프로덕션 용도로 사용되지 않습니다.

샘플 프로젝트의 제목은 다음과 같습니다.

- 압축: 이 블루프린트는 원하는 파일 크기에 따라 입력 파일을 더 큰 청크로 압축하는 작업을 생성합니다.
- 변환: 이 블루프린트은 다양한 표준 파일 포맷의 입력 파일을 분석 워크로드에 최적화된 Apache Parkore 포맷으로 변환합니다.
- Amazon S3 위치 크롤링: 이 블루프린트는 여러 Amazon S3 위치를 크롤링하여 Data Catalog에 메타데이터 테이블을 추가합니다.
- Data Catalog에 대한 사용자 지정 연결: 이 블루프린트는 AWS Glue 사용자 지정 커넥터를 사용하여 데이터 스토어에 액세스하고, 레코드를 읽고, 레코드 스키마를 기준으로 AWS Glue Data Catalog에 테이블 정의를 채웁니다.
- 인코딩: 이 블루프린트은 UTF가 아닌 파일을 UTF로 인코딩된 파일로 변환합니다.
- 분할: 이 블루프린트는 특정 파티션 키를 기반으로 출력 파일을 파티션에 배치하는 분할 작업을 생성합니다.
- Amazon S3 데이터를 DynamoDB 테이블로 가져오기: 이 블루프린트는 Amazon S3에서 DynamoDB 테이블로 데이터를 가져옵니다.
- 관리 대상 표준 테이블: 이 블루프린트는 AWS Glue Data Catalog 테이블을 Lake Formation 테이블로 가져옵니다.

AWS Glue의 블루프린트 등록

AWS Glue 개발자가 블루프린트를 코딩하고 ZIP 아카이브를 Amazon Simple Storage Service(Amazon S3)에 업로드한 후 AWS Glue 관리자가 블루프린트를 등록해야 합니다. 블루프린트를 등록하면 사용할 수 있습니다.

블루프린트를 등록하면 AWS Glue는 블루프린트 아카이브를 예약된 Amazon S3 위치에 복사합니다. 그런 다음 업로드 위치에서 아카이브를 삭제할 수 있습니다.

블루프린트를 등록하려면 업로드된 아카이브가 포함된 Amazon S3 위치에 대한 읽기 권한이 필요합니다. AWS Identity and Access Management(IAM) 권한 `glue:CreateBlueprint`도 필요합니다. 블루프린트를 등록하고, 보고, 유지 관리해야 하는 AWS Glue 관리자에 대해 제안된 권한은 블루프린트에 대한 AWS Glue 관리자 권한 섹션을 참조하세요.

AWS Glue 콘솔, AWS Glue API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 블루프린트를 등록할 수 있습니다.

블루프린트를 등록하려면(콘솔)

1. Amazon S3의 블루프린트 ZIP 아카이브에 대한 읽기 권한(`s3:GetObject`)이 있는지 확인합니다.
2. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

   블루프린트를 등록할 권한이 있는 사용자로 로그인합니다. 블루프린트 ZIP 아카이브가 포함된 Amazon S3 버킷과 동일한 AWS 리전으로 전환합니다.
3. 탐색 창에서 블루프린트(Blueprints)를 선택합니다. 그런 다음 블루프린트 페이지에서 블루프린트 추가(Add blueprint)를 선택합니다.
4. 블루프린트 이름과 선택적 설명을 입력합니다.
5. [ZIP 아카이브 위치(S3)(ZIP archive location (S3))]에 업로드된 블루프린트 ZIP 아카이브의 Amazon S3 경로를 입력합니다. 경로에 아카이브 파일 이름을 포함하고 경로를 `s3://`로 시작합니다.
6. (선택 사항) 태그를 하나 이상 추가합니다.
7. [블루프린트 추가(Add blueprint)]를 선택합니다.

   블루프린트 페이지가 반환되고 블루프린트 상태가 `CREATING`임을 표시합니다. 상태가 `ACTIVE` 또는 `FAILED`로 변경될 때까지 새로 고침 버튼을 선택합니다.
8. 상태가 `FAILED`인 경우 블루프린트를 선택하고 [작업(Actions)] 메뉴에서 [보기(View)]를 선택합니다.

   세부 정보 페이지에 실패 이유가 표시됩니다. 오류 메시지가 “위치의 객체에 액세스할 수 없습니다...(Unable to access object at location...)” 또는 “위치의 객체에 대한 액세스가 거부되었습니다...(Access denied on object at location...)”인 경우 다음 요구 사항을 검토합니다.

   - 로그인한 사용자는 Amazon S3의 블루프린트 ZIP 아카이브에 대한 읽기 권한이 있어야 합니다.
   - ZIP 아카이브가 포함된 Amazon S3 버킷에는 AWS 계정 ID에 객체에 대한 읽기 권한을 부여하는 버킷 정책이 있어야 합니다. 자세한 내용은 AWS Glue에서 블루프린트 개발 단원을 참조하십시오.
   - 사용 중인 Amazon S3 버킷은 콘솔에서 로그인한 리전과 동일한 리전에 있어야 합니다.
9. 데이터 분석가에게 블루프린트에 대한 권한이 있는지 확인합니다.

   데이터 분석가를 위한 제안된 IAM 정책은 블루프린트에 대한 데이터 분석가 권한에 나와 있습니다. 이 정책은 모든 리소스에 `glue:GetBlueprint`를 부여합니다. 정책이 리소스 수준에서 보다 세분화된 경우 데이터 분석가에게 이 새로 생성된 리소스에 대한 권한을 부여합니다.

블루프린트를 등록하려면(AWS CLI)

1. 다음 명령을 입력합니다.

```
aws glue create-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. 다음 명령을 입력하여 블루프린트 상태를 확인합니다. 상태가 ACTIVE 또는 FAILED가 될 때까지 명령을 반복합니다.

```
aws glue get-blueprint --name <blueprint-name>
```

상태가 FAILED이고 오류 메시지가 “위치의 객체에 액세스할 수 없습니다...(Unable to access object at location...)” 또는 “위치의 객체에 대한 액세스가 거부되었습니다...(Access denied on object at location...)”인 경우 다음 요구 사항을 검토합니다.

- 로그인한 사용자는 Amazon S3의 블루프린트 ZIP 아카이브에 대한 읽기 권한이 있어야 합니다.
- ZIP 아카이브가 포함된 Amazon S3 버킷에는 AWS 계정 ID에 객체에 대한 읽기 권한을 부여하는 버킷 정책이 있어야 합니다. 자세한 내용은 블루프린트 게시 단원을 참조하십시오.
- 사용 중인 Amazon S3 버킷은 콘솔에서 로그인한 리전과 동일한 리전에 있어야 합니다.

> 다음 사항도 참조하세요.
>
> - AWS Glue의 블루프린트 개요

AWS Glue에서 블루프린트 보기

블루프린트를 보고 블루프린트 설명, 상태 및 파라미터 사양을 검토하고 블루프린트 ZIP 아카이브를 다운로드합니다.

AWS Glue 콘솔, AWS Glue API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 블루프린트를 볼 수 있습니다.

블루프린트를 보려면(콘솔)

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 블루프린트(Blueprints)를 선택합니다.

3. 블루프린트 페이지에서 블루프린트를 선택합니다. [작업(Actions)] 메뉴에서 [보기(View)]을 선택합니다.

블루프린트를 보려면(AWS CLI)

- 블루프린트 이름, 설명 및 상태만 보려면 다음 명령을 입력합니다. *<blueprint-name>*을 보려는 블루프린트의 이름으로 바꿉니다.

```
aws glue get-blueprint --name <blueprint-name>
```

명령 출력은 다음과 같습니다.

```
{
    "Blueprint": {
        "Name": "myDemoBP",
        "CreatedOn": 1587414516.92,
        "LastModifiedOn": 1587428838.671,
        "BlueprintLocation": "s3://amzn-s3-demo-bucket1/demo/
DemoBlueprintProject.zip",
        "Status": "ACTIVE"
    }
}
```

파라미터 사양도 보려면 다음 명령을 입력합니다.

```
aws glue get-blueprint --name <blueprint-name>  --include-parameter-spec
```

명령 출력은 다음과 같습니다.

```
{
    "Blueprint": {
        "Name": "myDemoBP",
        "CreatedOn": 1587414516.92,
        "LastModifiedOn": 1587428838.671,
        "ParameterSpec": "{\"WorkflowName\":{\"type\":\"String\",\"collection
\":false,\"description\":null,\"defaultValue\":null,\"allowedValues\":null},
\"PassRole\":{\"type\":\"String\",\"collection\":false,\"description\":null,
\"defaultValue\":null,\"allowedValues\":null},\"DynamoDBTableName\":{\"type
\":\"String\",\"collection\":false,\"description\":null,\"defaultValue\":null,
```

```
\"allowedValues\":null},\"ScriptLocation\":{\"type\":\"String\",\"collection
\":false,\"description\":null,\"defaultValue\":null,\"allowedValues\":null}}",
        "BlueprintLocation": "s3://awsexamplebucket1/demo/
DemoBlueprintProject.zip",
        "Status": "ACTIVE"
    }
}
```

`--include-blueprint` 인수를 추가하여 AWS Glue가 저장한 블루프린트 ZIP 아카이브를 다운로드하기 위해 브라우저에 붙여넣을 수 있는 URL을 출력에 포함합니다.

다음 사항도 참조하세요.

- AWS Glue의 블루프린트 개요

AWS Glue에서 블루프린트 업데이트

수정된 레이아웃 스크립트, 수정된 블루프린트 파라미터 집합 또는 수정된 지원 파일이 있는 경우 블루프린트를 업데이트할 수 있습니다. 블루프린트를 업데이트하면 새 버전이 생성됩니다.

블루프린트를 업데이트해도 블루프린트에서 생성된 기존 워크플로에는 영향이 없습니다.

AWS Glue 콘솔, AWS Glue API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 블루프린트를 업데이트할 수 있습니다.

다음 절차에서는 AWS Glue 개발자가 업데이트된 블루프린트 ZIP 아카이브를 생성하여 Amazon S3에 업로드했다고 가정합니다.

블루프린트를 업데이트하려면(콘솔)

1. Amazon S3의 블루프린트 ZIP 아카이브에 대한 읽기 권한(`s3:GetObject`)이 있는지 확인합니다.
2. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

   블루프린트를 업데이트할 권한이 있는 사용자로 로그인합니다. 블루프린트 ZIP 아카이브가 포함된 Amazon S3 버킷과 동일한 AWS 리전으로 전환합니다.
3. 탐색 창에서 블루프린트(Blueprints)를 선택합니다.

4. 블루프린트 페이지에서 블루프린트를 선택하고 작업(Actions) 메뉴에서 편집(Edit)을 선택합니다.
5. [블루프린터 편집(Edit a blueprint)] 페이지에서 블루프린트 [설명(Description)] 또는 [ZIP 아카이브 위치(S3)(ZIP archive location (S3))]를 업데이트합니다. 경로에 아카이브 파일 이름을 포함해야 합니다.
6. Save(저장)를 선택합니다.

   블루프린트 페이지가 반환되고 블루프린트 상태가 `UPDATING`임을 표시합니다. 상태가 `ACTIVE` 또는 `FAILED`로 변경될 때까지 새로 고침 버튼을 선택합니다.
7. 상태가 `FAILED`인 경우 블루프린트를 선택하고 [작업(Actions)] 메뉴에서 [보기(View)]를 선택합니다.

   세부 정보 페이지에 실패 이유가 표시됩니다. 오류 메시지가 “위치의 객체에 액세스할 수 없습니다...(Unable to access object at location...)” 또는 “위치의 객체에 대한 액세스가 거부되었습니다...(Access denied on object at location...)”인 경우 다음 요구 사항을 검토합니다.

   - 로그인한 사용자는 Amazon S3의 블루프린트 ZIP 아카이브에 대한 읽기 권한이 있어야 합니다.
   - ZIP 아카이브가 포함된 Amazon S3 버킷에는 AWS 계정 ID에 객체에 대한 읽기 권한을 부여하는 버킷 정책이 있어야 합니다. 자세한 내용은 블루프린트 게시 단원을 참조하십시오.
   - 사용 중인 Amazon S3 버킷은 콘솔에서 로그인한 리전과 동일한 리전에 있어야 합니다.

> **Note**
>
> 업데이트가 실패하면 성공적으로 등록되거나 업데이트된 블루프린트의 최신 버전이 다음 블루프린트 실행에 사용됩니다.

블루프린트를 업데이트하려면(AWS CLI)

1. 다음 명령을 입력합니다.

```
aws glue update-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. 다음 명령을 입력하여 블루프린트 상태를 확인합니다. 상태가 `ACTIVE` 또는 `FAILED`가 될 때까지 명령을 반복합니다.

```
aws glue get-blueprint --name <blueprint-name>
```

상태가 FAILED이고 오류 메시지가 “위치의 객체에 액세스할 수 없습니다...(Unable to access object at location...)” 또는 “위치의 객체에 대한 액세스가 거부되었습니다...(Access denied on object at location...)”인 경우 다음 요구 사항을 검토합니다.

- 로그인한 사용자는 Amazon S3의 블루프린트 ZIP 아카이브에 대한 읽기 권한이 있어야 합니다.
- ZIP 아카이브가 포함된 Amazon S3 버킷에는 AWS 계정 ID에 객체에 대한 읽기 권한을 부여하는 버킷 정책이 있어야 합니다. 자세한 내용은 블루프린트 게시 단원을 참조하십시오.
- 사용 중인 Amazon S3 버킷은 콘솔에서 로그인한 리전과 동일한 리전에 있어야 합니다.

다음 사항도 참조하세요.

- AWS Glue의 블루프린트 개요

AWS Glue의 블루프린트에서 워크플로 생성

AWS Glue 워크플로를 수동으로 생성하여 한 번에 하나의 구성 요소를 추가하거나 AWS Glue 블루프린트에서 워크플로를 생성할 수 있습니다. AWS Glue에는 일반적인 사용 사례에 대한 블루프린트가 포함되어 있습니다. AWS Glue 개발자가 추가 블루프린트를 생성할 수 있습니다.

Important

워크플로 내 작업, 크롤러, 트리거의 총수를 100개 이하로 제한합니다. 100개가 넘게 포함될 경우 워크플로 실행을 재개하거나 중지하려고 할 때 오류가 발생할 수 있습니다.

블루프린트를 사용하면 블루프린트에서 정의한 일반화된 사용 사례를 기반으로 특정 사용 사례에 대한 워크플로를 빠르게 생성할 수 있습니다. 블루프린트 파라미터에 대한 값을 제공하여 특정 사용 사례를 정의합니다. 예를 들어 데이터 집합을 분할하는 블루프린트에는 Amazon S3 소스 및 대상 경로가 파라미터로 포함될 수 있습니다.

AWS Glue는 블루프린트를 실행하여 블루프린트에서 워크플로를 생성합니다. 블루프린트 실행은 제공한 파라미터 값을 저장하고 워크플로 및 해당 구성 요소 생성의 진행률과 결과를 추적하는 데 사용됩니다. 워크플로 문제를 해결할 때 블루프린트 실행을 보고 워크플로 생성에 사용된 블루프린트 파라미터 값을 확인할 수 있습니다.

워크플로를 생성하고 보려면 특정 IAM 권한이 필요합니다. 제안된 IAM 정책은 블루프린트에 대한 데이터 분석가 권한 섹션을 참조하세요.

AWS Glue 콘솔, AWS Glue API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 블루프린트에서 워크플로를 생성할 수 있습니다.

블루프린트에서 워크플로를 생성하려면(콘솔)

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

   워크플로를 생성할 권한이 있는 사용자로 로그인합니다.
2. 탐색 창에서 블루프린트(Blueprints)를 선택합니다.
3. 블루프린트를 선택하고 [작업(Actions)] 메뉴에서 [워크플로 생성(Create workflow)]을 선택합니다.
4. [<blueprint-name>에서 워크플로 생성(Create a workflow from <blueprint-name>)] 페이지에서 다음 정보를 입력합니다.

   블루프린트 파라미터

   블루프린트 파라미터는 블루프린트 설계에 따라 다릅니다. 파라미터에 대한 질문은 개발자에게 문의하세요. 블루프린트에는 일반적으로 워크플로 이름에 대한 파라미터가 포함됩니다.

   IAM 역할

   AWS Glue가 워크플로 및 해당 구성 요소를 생성하기 위해 맡는 역할입니다. 역할에는 워크플로, 작업, 크롤러 및 트리거를 생성하고 삭제할 수 있는 권한이 있어야 합니다. 역할에 제안되는 정책은 블루프린트 역할에 대한 권한 섹션을 참조하세요.
5. 제출을 선택합니다.

   [블루프린트 세부 정보(Blueprint Details)] 페이지가 나타나고 아래쪽에 블루프린트 실행 목록이 표시됩니다.
6. 블루프린트 실행 목록의 최상위 블루프린트 실행에서 워크플로 생성 상태를 확인합니다.

   초기 상태는 `RUNNING`입니다. 상태가 `SUCCEEDED` 또는 `FAILED`로 바뀔 때까지 새로 고침 버튼을 선택합니다.
7. 다음 중 하나를 수행합니다.

   - 완료 상태가 `SUCCEEDED`이면 [워크플로(Workflows)] 페이지로 이동하여 새로 생성된 워크플로를 선택하여 실행할 수 있습니다. 워크플로를 실행하기 전에 설계 그래프를 검토할 수 있습니다.

- 완료 상태가 FAILED인 경우 블루프린트를 선택하고 [작업(Actions)] 메뉴에서 [보기(View)]를 선택하여 오류 메시지를 봅니다.

워크플로와 블루프린트에 대한 자세한 내용은 다음 주제를 참조하세요.

- AWS Glue의 워크플로 개요
- AWS Glue에서 블루프린트 업데이트
- AWS Glue에서 워크플로 수동 생성 및 구축

AWS Glue에서 블루프린트 실행 보기

블루프린트 실행을 보고 다음 정보를 확인합니다.

- 생성된 워크플로의 이름입니다.
- 워크플로를 생성하는 데 사용된 블루프린트 파라미터 값입니다.
- 워크플로 생성 작업의 상태입니다.

AWS Glue 콘솔, AWS Glue API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 블루프린트 실행을 볼 수 있습니다.

블루프린트 실행을 보려면(콘솔)

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 블루프린트(Blueprints)를 선택합니다.
3. 블루프린트 페이지에서 블루프린트를 선택합니다. [작업(Actions)] 메뉴에서 [보기(View)]을 선택합니다.
4. [블루프린트 세부 정보(Blueprint Details)] 페이지 하단에서 블루프린트 실행을 선택하고 [작업(Actions)] 메뉴에서 [보기(View)]를 선택합니다.

블루프린트 실행을 보려면(AWS CLI)

- 다음 명령을 입력합니다. *<blueprint-name>*을 블루프린트의 이름으로 바꿉니다. *<blueprint-run-id>*를 블루프린트 실행 ID로 바꿉니다.

```
aws glue get-blueprint-run --blueprint-name <blueprint-name> --run-id <blueprint-run-id>
```

다음 사항도 참조하세요.

- AWS Glue의 블루프린트 개요

AWS Glue용 AWS CloudFormation

AWS CloudFormation는 여러 AWS 리소스를 생성할 수 있는 서비스입니다. AWS Glue는 AWS Glue Data Catalog에 객체를 생성하는 API 작업을 제공합니다. 그러나 AWS CloudFormation 템플릿 파일에 AWS Glue 객체 및 기타 AWS 리소스 객체를 정의하고 생성할 때 용이할 수 있습니다. 그런 다음 객체 생성 작업을 자동화할 수 있습니다.

AWS CloudFormation은 단순화된 구문((JSON(JavaScript Object Notation) 또는 YAML(YAML Ain't Markup Language))을 제공하여 AWS 리소스의 생성을 보여줍니다. AWS CloudFormation 템플릿을 사용하여 데이터베이스, 테이블, 파티션, 크롤러, 분류자 및 연결과 같은 Data Catalog 객체를 정의할 수 있습니다. 작업, 트리거 및 개발 엔드포인트와 같은 ETL 객체를 정의할 수도 있습니다. 필요한 모든 AWS 리소스를 설명하는 템플릿을 생성하면 AWS CloudFormation이 해당 리소스의 프로비저닝과 구성을 담당합니다.

자세한 내용은 AWS CloudFormation User Guide의 What Is AWS CloudFormation? 및 Working with AWS CloudFormation Templates를 참조하세요.

AWS Glue와 호환되는 AWS CloudFormation 템플릿을 사용하려는 경우 관리자는 AWS CloudFormation과 AWS 서비스 및 의존 대상 작업에 액세스 권한을 부여해야 합니다. AWS CloudFormation 리소스를 생성하는 데 필요한 권한을 부여하려면 다음 정책을 AWS CloudFormation으로 작업하는 사용자에게 연결합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

다음 테이블은 작업을 포함하여 AWS CloudFormation 템플릿을 수행하도록 허용합니다. AWS 리소스 유형 및 속성 유형에 대한 정보 링크를 포함하고 AWS CloudFormation 템플릿에 추가할 수 있습니다.

AWS Glue 리소스	AWS CloudFormation 템플릿	AWS Glue 샘플
분류자	AWS::Glue::Classifier	Grok 분류자, JSON 분류자, XML 분류자
연결	AWS::Glue::Connection	MySQL 연결
크롤러	AWS::Glue::Crawler	Amazon S3 크롤러, MySQL 크롤러
데이터베이스	AWS::Glue::Database	데이터베이스 비우기, 데이터베이스 및 테이블
개발 엔드포인트	AWS::Glue::DevEndpoint	개발 엔드포인트
작업	AWS::Glue::Job	Amazon S3 작업, JDBC 작업
기계 학습 변환	AWS::Glue::MLTransform	기계 학습 변환
데이터 품질 규칙 세트	AWS::Glue::DataQualityRuleset	데이터 품질 규칙 세트, EventBridge 스케줄러에서 데이터 품질 규칙 세트
Partition	AWS::Glue::Partition	테이블 파티션
표	AWS::Glue::Table	데이터베이스의 테이블
트리거	AWS::Glue::Trigger	온디맨드 트리커, 일정이 정해진 트리거, 조건부 트리거

시작하려면 다음 샘플 템플릿을 사용하고 메타데이터로 사용자 지정합니다. AWS CloudFormation 콘솔을 사용하여 AWS CloudFormation 스택을 생성한 다음 객체를 AWS Glue 및 기타 관련 서비스에 추가합니다. AWS Glue 객체의 많은 필드는 선택 사항입니다. 이 템플릿은 필요한 필드 또는 작업 및 기능적 AWS Glue객체에 필요한 필드를 보여줍니다.

AWS CloudFormation 템플릿은 JSON 또는 YAML 형식일 수 있습니다. 이런 예제의 YAML은 더 쉽게 읽게 사용됩니다. 예제는 설명문(#)을 포함하여 템플릿에서 정의된 값을 설명합니다.

AWS CloudFormation 템플릿에 Parameters 섹션을 포함시킬 수 있습니다. 이 섹션은 샘플 텍스트 또는 YAML 파일이 AWS CloudFormation 콘솔에 입력되어 스택을 생성할 경우 변경될 수 있습니다. 템플릿의 Resources 섹션에는 AWS Glue 구문 정의와 관련 객체가 포함됩니다. AWS CloudFormation 템플릿 구문 정의에는 더 세분화된 속성 구문을 포함한 속성이 포함될 수 있습니다. AWS Glue 객체는 모든 속성이 없어도 생성할 수 있습니다. 다음 샘플은 AWS Glue 객체를 생성할 수 있는 일반 속성 예제 값을 보여줍니다.

AWS Glue 데이터베이스의 샘플 AWS CloudFormation 템플릿

Data Catalog의 AWS Glue 데이터베이스는 메타데이터 테이블을 포함합니다. 데이터베이스는 적은 속성으로 구성되어 있고 AWS CloudFormation 템플릿으로 Data Catalog에 생성될 수 있습니다. 다음 샘플 템플릿을 제공하여 시작을 돕고 AWS Glue로 AWS CloudFormation 스택의 사용에 대하여 보여줍니다. 샘플 템플릿이 생성한 유일한 리소스는 cfn-mysampledatabase라는 데이터베이스입니다. YAML을 입력한 후, 샘플 텍스트를 편집하거나 AWS CloudFormation 콘솔 값을 변경하면 리소스를 변경할 수 있습니다.

다음은 AWS Glue 데이터베이스를 생성할 수 있는 일반 속성 예제 값을 보여줍니다. AWS Glue용 AWS CloudFormation 데이터베이스 템플릿에 대한 자세한 내용은 AWS::Glue::Database를 참조하세요.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CloudFormation template in YAML to demonstrate creating a database named
 mysampledatabase
# The metadata created in the Data Catalog points to the flights public S3 bucket
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  CFNDatabaseName:
    Type: String
    Default: cfn-mysampledatabse

# Resources section defines metadata for the Data Catalog
Resources:
# Create an AWS Glue database
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      # The database is created in the Data Catalog for your account
```

```
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        # The name of the database is defined in the Parameters section above
        Name: !Ref CFNDatabaseName
        Description: Database to hold tables for flights data
        LocationUri: s3://crawler-public-us-east-1/flight/2016/csv/
        #Parameters: Leave AWS database parameters blank
```

AWS Glue 데이터베이스. 테이블 및 파티션의 샘플 AWS CloudFormation 템플릿

AWS Glue 테이블은 메타데이터를 포함하여 ETL 스크립트에서 진행하고자 하는 데이터의 구조 및 위치를 정의할 수 있습니다. 테이블 내에서 파티션을 정의하여 데이터를 병렬로 진행할 수 있습니다. 파티션은 키로 정의한 데이터 덩어리입니다. 예를 들어, 키로써 월을 사용하여 모든 1월 데이터는 동일한 파티션을 포함합니다. AWS Glue의 경우, 데이터베이스는 테이블을 포함하고 테이블은 파티션을 포함합니다.

다음 샘플에서는 AWS CloudFormation 템플릿을 사용하여 데이터베이스, 테이블 및 파티션을 채우는 방법을 보여줍니다. 기본 데이터 포맷은 `csv`이고 콤마(,)로 범위가 제한됩니다. 데이터베이스는 테이블을 포함하기 전에 존재하고 테이블은 파티션이 생성되기 전에 존재해야 하기 때문에 템플릿은 `DependsOn`을 사용하여 생성될 경우, 객체의 종속성을 정의합니다.

샘플 값은 공개적으로 사용 가능한 Amazon S3 버킷에서 항공 데이터를 포함하는 테이블을 정의합니다. 설명하자면 오직 몇몇 데이터 열과 하나의 파티션 키가 정의됩니다. 4개의 파티션은 Data Catalog에도 정의됩니다. 기본 데이터의 스토리지를 설명하는 어떤 필드는 `StorageDescriptor` 필드에서 보여줍니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CloudFormation template in YAML to demonstrate creating a database, a table,
 and partitions
# The metadata created in the Data Catalog points to the flights public S3 bucket
#
# Parameters substituted in the Resources section
# These parameters are names of the resources created in the Data Catalog
Parameters:
  CFNDatabaseName:
    Type: String
    Default: cfn-database-flights-1
```

```
  CFNTableName1:
    Type: String
    Default: cfn-manual-table-flights-1
# Resources to create metadata in the Data Catalog
Resources:
###
# Create an AWS Glue database
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: Database to hold tables for flights data
###
# Create an AWS Glue table
  CFNTableFlights:
    # Creating the table waits for the database to be created
    DependsOn: CFNDatabaseFlights
    Type: AWS::Glue::Table
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableInput:
        Name: !Ref CFNTableName1
        Description: Define the first few columns of the flights table
        TableType: EXTERNAL_TABLE
        Parameters: {
    "classification": "csv"
  }
#       ViewExpandedText: String
        PartitionKeys:
        # Data is partitioned by month
        - Name: mon
          Type: bigint
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: year
            Type: bigint
          - Name: quarter
            Type: bigint
          - Name: month
            Type: bigint
```

```
          - Name: day_of_month
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 1
# Create an AWS Glue partition
  CFNPartitionMon1:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 1
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=1/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 2
# Create an AWS Glue partition
  CFNPartitionMon2:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 2
        StorageDescriptor:
```

```
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=2/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 3
# Create an AWS Glue partition
  CFNPartitionMon3:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 3
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=3/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 4
# Create an AWS Glue partition
  CFNPartitionMon4:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
```

```
        - 4
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=4/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
```

AWS Glue Grok 분류자의 샘플 AWS CloudFormation 템플릿

AWS Glue 분류자는 데이터의 스키마를 결정합니다. 사용자 분류자의 한 유형은 grok 패턴을 사용하여 데이터와 매치합니다. 패턴이 맞으면 사용자 분류자는 테이블 스키마를 생성하고 `classification`을 분류자 정의의 값 세트를 설정합니다.

이 샘플은 분류자를 생성하여 `message`라는 하나의 열과 함께 스키마를 생성하고 `greedy`의 분류를 설정합니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-grok-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses grok pattern to put all data in one column and classifies
 it as "greedy".
```

```
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      GrokClassifier:
        #Grok classifier that puts all data in one column
        Name: !Ref CFNClassifierName
        Classification: greedy

        GrokPattern: "%{GREEDYDATA:message}"
        #CustomPatterns: none
```

AWS Glue JSON 분류자의 샘플 AWS CloudFormation 템플릿

AWS Glue 분류자는 데이터의 스키마를 결정합니다. 분류자가 분류해야 하는 JSON 데이터를 정의하는 `JsonPath` 문자열을 사용하는 유형의 사용자 지정 분류자입니다. AWS Glue은(는) `JsonPath`에 대한 연산자 하위 집단을 지원합니다. 자세한 설명은 Writing JsonPath Custom Classifiers(JsonPath 사용자 지정 분류자 작성)에서 확인할 수 있습니다.

패턴이 맞으면 사용자 분류자는 테이블 스키마를 생성하는 데 사용됩니다.

이 샘플은 객체 내의 `Records3` 어레이에 있는 개별 레코드로 스키마를 생성하는 분류자를 생성합니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a JSON classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-json-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses a JSON pattern.
```

```
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      JSONClassifier:
        #JSON classifier
        Name: !Ref CFNClassifierName
        JsonPath: $.Records3[*]
```

AWS Glue XML 분류자의 샘플 AWS CloudFormation 템플릿

AWS Glue 분류자는 데이터의 스키마를 결정합니다. 분석 중인 XML 문서에 각각의 레코드를 포함하는 요소를 지정하도록 XML 태그를 지정하는 유형의 사용자 지정 분류자입니다. 패턴이 맞으면 사용자 분류자는 테이블 스키마를 생성하고 `classification`을 분류자 정의의 값 세트를 설정합니다.

이 샘플은 분류자를 생성하여 `Record` 태그에 있는 각 레코드와 함께 스키마를 생성하고 `XML`의 분류를 설정합니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating an XML classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-xml-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses the XML pattern and classifies it as "XML".
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      XMLClassifier:
        #XML classifier
```

```
        Name: !Ref CFNClassifierName
        Classification: XML
        RowTag: <Records>
```

Amazon S3용 AWS Glue 크롤러의 샘플 AWS CloudFormation 템플릿

AWS Glue 크롤러는 데이터에 대응하는 Data Catalog에 메타데이터 테이블을 생성합니다. 그런 다음 이 테이블 정의를 ETL 작업의 원본 및 대상으로써 사용할 수 있습니다.

이 샘플은 크롤러, 필요한 IAM 역할 및 AWS Glue 데이터베이스를 Data Catalog에 생성합니다. 크롤러가 실행되면 IAM 역할을 가정하고 퍼블릭 항공 데이터의 데이터베이스에 테이블을 생성합니다. 테이블은 접두사 "cfn_sample_1_"로 생성합니다. 템플릿에 의해 생성된 IAM 역할은 글로벌 권한을 허용하여 사용자 역할을 생성하고자 합니다. 이 분류자는 사용자 지정 분류자를 정의하지 않습니다. 기본적으로 AWS Glue 내장 분류자가 사용됩니다.

이 샘플을 AWS CloudFormation 콘솔에 제출하면 IAM 역할을 생성하고자 하는 확신이 있어야 합니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a crawler
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNCrawlerName:
    Type: String
    Default: cfn-crawler-flights-1
  CFNDatabaseName:
    Type: String
    Default: cfn-database-flights-1
  CFNTablePrefixName:
    Type: String
    Default: cfn_sample_1_
#
#
# Resources section defines metadata for the Data Catalog
Resources:
```

```
#Create IAM Role assumed by the crawler. For demonstration, this role is given all
 permissions.
  CFNRoleFlights:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
            Principal:
              Service:
                - "glue.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        -
          PolicyName: "root"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              -
                Effect: "Allow"
                Action: "*"
                Resource: "*"
 # Create a database to contain tables created by the crawler
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: "AWS Glue container to hold metadata tables for the flights
crawler"
 #Create a crawler to crawl the flights data on a public S3 bucket
  CFNCrawlerFlights:
    Type: AWS::Glue::Crawler
    Properties:
      Name: !Ref CFNCrawlerName
      Role: !GetAtt CFNRoleFlights.Arn
      #Classifiers: none, use the default classifier
      Description: AWS Glue crawler to crawl flights data
      #Schedule: none, use default run-on-demand
```

```
      DatabaseName: !Ref CFNDatabaseName
      Targets:
        S3Targets:
          # Public S3 bucket with the flights data
          - Path: "s3://crawler-public-us-east-1/flight/2016/csv"
      TablePrefix: !Ref CFNTablePrefixName
      SchemaChangePolicy:
        UpdateBehavior: "UPDATE_IN_DATABASE"
        DeleteBehavior: "LOG"
      Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":
{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"AddOrUpdateBehavior\":
\"MergeNewColumns\"}}}"
```

AWS Glue 연결의 샘플 AWS CloudFormation 템플릿

Data Catalog의 AWS Glue 연결은 JDBC 데이터베이스 연결에 필요한 JDBC 및 네트워크 정보를 포함합니다. 이 정보는 JDBC 데이터베이스에 연결하여 ETL 작업을 크롤하거나 실행할 때 사용됩니다.

이 샘플은 devdb라는 Amazon RDS MySQL 데이터베이스의 연결을 생성합니다. 이 연결이 사용되면 IAM 역할, 데이터베이스 자격 및 네트워크 연결 값도 제공되어야 합니다. 템플릿에 필요한 필드의 상세 정보를 참조하십시오.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a connection
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the connection to be created
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
  CFNJDBCString:
    Type: String
    Default: "jdbc:mysql://xxx-mysql.yyyyyyyyyyyyyy.us-east-1.rds.amazonaws.com:3306/
devdb"
  CFNJDBCUser:
    Type: String
    Default: "master"
```

```
  CFNJDBCPassword:
    Type: String
    Default: "12345678"
    NoEcho: true
#
#
# Resources section defines metadata for the Data Catalog
Resources:
  CFNConnectionMySQL:
    Type: AWS::Glue::Connection
    Properties:
      CatalogId: !Ref AWS::AccountId
      ConnectionInput:
        Description: "Connect to MySQL database."
        ConnectionType: "JDBC"
        #MatchCriteria: none
        PhysicalConnectionRequirements:
          AvailabilityZone: "us-east-1d"
          SecurityGroupIdList:
           - "sg-7d52b812"
          SubnetId: "subnet-84f326ee"
        ConnectionProperties: {
          "JDBC_CONNECTION_URL": !Ref CFNJDBCString,
          "USERNAME": !Ref CFNJDBCUser,
          "PASSWORD": !Ref CFNJDBCPassword
        }
        Name: !Ref CFNConnectionName
```

JDBC을 위한 AWS Glue 크롤러의 샘플 AWS CloudFormation 템플릿

AWS Glue 크롤러는 데이터에 대응하는 Data Catalog에 메타데이터 테이블을 생성합니다. 그런 다음 이 테이블 정의를 ETL 작업의 원본 및 대상으로써 사용할 수 있습니다.

이 샘플은 크롤러, 필요한 IAM 역할 및 AWS Glue 데이터베이스를 Data Catalog에 생성합니다. 크롤러가 실행되면 IAM 역할을 가정하고 MySQL 데이터베이스에 저장된 퍼블릭 항공 데이터의 데이터베이스에 테이블을 생성합니다. 테이블은 접두사 "`cfn_jdbc_1_`"로 생성합니다. 템플릿에 의해 생성된 IAM 역할은 글로벌 권한을 허용하여 사용자 역할을 생성하고자 합니다. JDBC 데이터는 사용자 지정 분류자를 정의할 수 없습니다. 기본적으로 AWS Glue 내장 분류자가 사용됩니다.

이 샘플을 AWS CloudFormation 콘솔에 제출하면 IAM 역할을 생성하고자 하는 확신이 있어야 합니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a crawler
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNCrawlerName:
    Type: String
    Default: cfn-crawler-jdbc-flights-1
# The name of the database to be created to contain tables
  CFNDatabaseName:
    Type: String
    Default: cfn-database-jdbc-flights-1
# The prefix for all tables crawled and created
  CFNTablePrefixName:
    Type: String
    Default: cfn_jdbc_1_
# The name of the existing connection to the MySQL database
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
# The name of the JDBC path (database/schema/table) with wildcard (%) to crawl
  CFNJDBCPath:
    Type: String
    Default: saldev/%
#
#
# Resources section defines metadata for the Data Catalog
Resources:
#Create IAM Role assumed by the crawler. For demonstration, this role is given all
 permissions.
  CFNRoleFlights:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
```

```
            Principal:
              Service:
                - "glue.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        -
          PolicyName: "root"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              -
                Effect: "Allow"
                Action: "*"
                Resource: "*"
# Create a database to contain tables created by the crawler
 CFNDatabaseFlights:
   Type: AWS::Glue::Database
   Properties:
     CatalogId: !Ref AWS::AccountId
     DatabaseInput:
       Name: !Ref CFNDatabaseName
       Description: "AWS Glue container to hold metadata tables for the flights
crawler"
#Create a crawler to crawl the flights data in MySQL database
 CFNCrawlerFlights:
   Type: AWS::Glue::Crawler
   Properties:
     Name: !Ref CFNCrawlerName
     Role: !GetAtt CFNRoleFlights.Arn
     #Classifiers: none, use the default classifier
     Description: AWS Glue crawler to crawl flights data
     #Schedule: none, use default run-on-demand
     DatabaseName: !Ref CFNDatabaseName
     Targets:
       JdbcTargets:
         # JDBC MySQL database with the flights data
         - ConnectionName: !Ref CFNConnectionName
           Path: !Ref CFNJDBCPath
         #Exclusions: none
     TablePrefix: !Ref CFNTablePrefixName
     SchemaChangePolicy:
       UpdateBehavior: "UPDATE_IN_DATABASE"
```

```
        DeleteBehavior: "LOG"
   Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":
{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"AddOrUpdateBehavior\":
\"MergeNewColumns\"}}}"
```

Amazon S3에서 Amazon S3로의 AWS Glue 작업을 위한 샘플 AWS CloudFormation 템플릿

Data Catalog의 AWS Glue 작업은 AWS Glue의 스크립트를 실행하는 데 필요한 파라미터 값을 포함합니다.

이 샘플은 csv 포맷의 Amazon S3 버킷에서 항공 데이터를 읽고 Amazon S3 Parquet 파일에 작성하는 작업을 생성합니다. 이 작업에 실행되는 스크립트는 먼저 존재해야 합니다. AWS Glue 콘솔을 통해 환경의 ETL 스트립트를 생성할 수 있습니다. 이 작업이 실행되면 올바른 권한과 함께 IAM 역할도 제공되어야 합니다.

범용 파라미터 값은 템플릿에서 보여줍니다. 예를 들어 AllocatedCapacity(DPU)는 기본값 5로 돌아갑니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a job using the public flights S3 table in a
 public bucket
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the job to be created
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-2
# The name of the IAM role that the job assumes. It must have access to data, script,
 temporary directory
  CFNIAMRoleName:
    Type: String
    Default: AWSGlueServiceRoleGA
# The S3 path where the script for this job is located
  CFNScriptLocation:
    Type: String
```

```
    Default: s3://aws-glue-scripts-123456789012-us-east-1/myid/sal-job-test2
#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create job to run script which accesses flightscsv table and write to S3 file as
 parquet.
# The script already exists and is called by this job
  CFNJobFlights:
    Type: AWS::Glue::Job
    Properties:
      Role: !Ref CFNIAMRoleName
      #DefaultArguments: JSON object
      # If script written in Scala, then set DefaultArguments={'--job-language';
 'scala', '--class': 'your scala class'}
      #Connections:  No connection needed for S3 to S3 job
      #  ConnectionsList
      #MaxRetries: Double
      Description: Job created with CloudFormation
      #LogUri: String
      Command:
        Name: glueetl
        ScriptLocation: !Ref CFNScriptLocation
             # for access to directories use proper IAM role with permission to buckets
 and folders that begin with "aws-glue-"
             # script uses temp directory from job definition if required (temp
 directory not used S3 to S3)
             # script defines target for output as s3://aws-glue-target/sal
      AllocatedCapacity: 5
      ExecutionProperty:
        MaxConcurrentRuns: 1
      Name: !Ref CFNJobName
```

JDBC에서 Amazon S3로의 AWS Glue 작업용 샘플 AWS CloudFormation 템플릿

Data Catalog의 AWS Glue 작업은 AWS Glue의 스크립트를 실행하는 데 필요한 파라미터 값을 포함합니다.

이 샘플은 `cfn-connection-mysql-flights-1`이라는 연결에 의해 정의되어 MySQL JDBC 데이터베이스로부터 항공 데이터를 읽고 Amazon S3 Parquet 파일에 작성하는 작업을 생성합니다. 이 작업

에 실행되는 스크립트는 먼저 존재해야 합니다. AWS Glue 콘솔을 통해 환경의 ETL 스트립트를 생성할 수 있습니다. 이 작업이 실행되면 올바른 권한과 함께 IAM 역할도 제공되어야 합니다.

범용 파라미터 값은 템플릿에서 보여줍니다. 예를 들어 `AllocatedCapacity`(DPU)는 기본값 5로 돌아갑니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a job using a MySQL JDBC DB with the flights
 data to an S3 file
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the job to be created
  CFNJobName:
    Type: String
    Default: cfn-job-JDBC-to-S3-1
# The name of the IAM role that the job assumes. It must have access to data, script,
 temporary directory
  CFNIAMRoleName:
    Type: String
    Default: AWSGlueServiceRoleGA
# The S3 path where the script for this job is located
  CFNScriptLocation:
    Type: String
    Default: s3://aws-glue-scripts-123456789012-us-east-1/myid/sal-job-dec4a
# The name of the connection used for JDBC data source
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create job to run script which accesses JDBC flights table via a connection and write
 to S3 file as parquet.
# The script already exists and is called by this job
  CFNJobFlights:
    Type: AWS::Glue::Job
    Properties:
```

```
      Role: !Ref CFNIAMRoleName
      #DefaultArguments: JSON object
      # For example, if required by script, set temporary directory as
 DefaultArguments={'--TempDir'; 's3://aws-glue-temporary-xyc/sal'}
      Connections:
        Connections:
        - !Ref CFNConnectionName
      #MaxRetries: Double
      Description: Job created with CloudFormation using existing script
      #LogUri: String
      Command:
        Name: glueetl
        ScriptLocation: !Ref CFNScriptLocation
             # for access to directories use proper IAM role with permission to buckets
 and folders that begin with "aws-glue-"
             # if required, script defines temp directory as argument TempDir and used
 in script like redshift_tmp_dir = args["TempDir"]
             # script defines target for output as s3://aws-glue-target/sal
      AllocatedCapacity: 5
      ExecutionProperty:
        MaxConcurrentRuns: 1
      Name: !Ref CFNJobName
```

AWS Glue 온디맨드 트리거의 샘플 AWS CloudFormation 템플릿

Data Catalog의 AWS Glue 트리거는 트리거가 실행되어 작업이 시작될 때 필요한 파라미터 값을 포함합니다. 온디맨드 트리거는 작동 시 시작합니다.

이 예제에서는 `cfn-job-S3-to-S3-1`이라는 하나의 작업을 시작하는 온디맨드 트리거를 만듭니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating an on-demand trigger
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-1
```

```
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-ondemand-flights-1
#
# Resources section defines metadata for the Data Catalog
# Sample CFN YAML to demonstrate creating an on-demand trigger for a job
Resources:
# Create trigger to run an existing job (CFNJobName) on an on-demand schedule.
  CFNTriggerSample:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: ON_DEMAND
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      #Schedule:
      #Predicate:
```

AWS Glue 일정이 정해진 트리거를 위한 샘플 AWS CloudFormation 템플릿

Data Catalog의 AWS Glue 트리거는 트리거가 실행되어 작업이 시작될 때 필요한 파라미터 값을 포함합니다. 일정이 정해진 트리거는 활성화되고 cron 타이머가 뜨면 시작합니다.

이 예제에서는 `cfn-job-S3-to-S3-1`이라는 하나의 작업을 시작하는 일정이 정해진 트리거를 만듭니다. 이 타이머는 매 10분마다 주중에 작업을 실행하는 cron 표현식입니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a scheduled trigger
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
```

```
    Type: String
    Default: cfn-job-S3-to-S3-1
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-scheduled-flights-1
#
# Resources section defines metadata for the Data Catalog
# Sample CFN YAML to demonstrate creating a scheduled trigger for a job
#
Resources:
# Create trigger to run an existing job (CFNJobName) on a cron schedule.
  TriggerSample1CFN:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: SCHEDULED
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      # # Run the trigger every 10 minutes on Monday to Friday
      Schedule: cron(0/10 * ? * MON-FRI *)
      #Predicate:
```

AWS Glue 조건부 트리거의 샘플 AWS CloudFormation 템플릿

Data Catalog의 AWS Glue 트리거는 트리거가 실행되어 작업이 시작될 때 필요한 파라미터 값을 포함합니다. 성공적으로 작업을 완료하는 것처럼 활성화되고 조건이 맞으면 조건부 트리거는 시작됩니다.

이 예제에서는 `cfn-job-S3-to-S3-1`이라는 하나의 작업을 시작하는 조건부 트리거를 만듭니다. 이 작업은 `cfn-job-S3-to-S3-2` 라는 작업을 성공적으로 완료하면 시작합니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a conditional trigger for a job, which starts
 when another job completes
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
```

```
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-1
  # The existing job that when it finishes causes trigger to fire
  CFNJobName2:
    Type: String
    Default: cfn-job-S3-to-S3-2
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-conditional-1
#
Resources:
# Create trigger to run an existing job (CFNJobName) when another job completes
 (CFNJobName2).
  CFNTriggerSample:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: CONDITIONAL
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      #Schedule: none
      Predicate:
        #Value for Logical is required if more than 1 job listed in Conditions
        Logical: AND
        Conditions:
          - LogicalOperator: EQUALS
            JobName: !Ref CFNJobName2
            State: SUCCEEDED
```

AWS Glue 개발 엔드포인트의 샘플 AWS CloudFormation 템플릿

AWS Glue 기계 학습 변환은 데이터를 정리하기 위한 사용자 지정 변환입니다. 여기에는 FindMatches로 명명된 사용 가능 변환이 하나 있습니다. FindMatches 변환으로는 레코드에 공통된 고유 식별자가 없고 정확히 일치되는 필드 또한 없을 경우에도 데이터 세트에서 중복 레코드나 일치 레코드를 식별할 수 있습니다.

이 샘플에서는 기계 학습 변환을 생성합니다. 기계 학습 변환을 생성하는 데 필요한 파라미터에 대한 자세한 내용을 알아보려면 AWS Lake Formation FindMatches로 레코드 매칭 섹션을 참조하세요.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a machine learning transform
#
# Resources section defines metadata for the machine learning transform
Resources:
  MyMLTransform:
    Type: "AWS::Glue::MLTransform"
    Condition: "isGlueMLGARegion"
    Properties:
      Name: !Sub "MyTransform"
      Description: "The bestest transform ever"
      Role: !ImportValue MyMLTransformUserRole
      GlueVersion: "1.0"
      WorkerType: "Standard"
      NumberOfWorkers: 5
      Timeout: 120
      MaxRetries: 1
      InputRecordTables:
        GlueTables:
          - DatabaseName: !ImportValue MyMLTransformDatabase
            TableName: !ImportValue MyMLTransformTable
      TransformParameters:
        TransformType: "FIND_MATCHES"
        FindMatchesParameters:
          PrimaryKeyColumnName: "testcolumn"
          PrecisionRecallTradeoff: 0.5
          AccuracyCostTradeoff: 0.5
          EnforceProvidedLabels: True
      Tags:
        key1: "value1"
        key2: "value2"
      TransformEncryption:
        TaskRunSecurityConfigurationName: !ImportValue
 MyMLTransformSecurityConfiguration
        MLUserDataEncryption:
          MLUserDataEncryptionMode: "SSE-KMS"
          KmsKeyId: !ImportValue MyMLTransformEncryptionKey
```

AWS Glue Data Quality 규칙 세트에 대한 샘플 AWS CloudFormation 템플릿

AWS Glue 데이터 품질 규칙 세트에는 데이터 카탈로그의 테이블에서 평가할 수 있는 규칙이 포함되어 있습니다. 규칙 세트를 대상 테이블에 배치한 후에는 데이터 카탈로그로 이동하여 규칙 세트 내 해당 규칙에 대해 데이터를 실행하는 평가를 실행할 수 있습니다. 이러한 규칙은 행 수 평가에서 데이터에 대한 참조 무결성 평가에 이르기까지 다양할 수 있습니다.

다음 샘플은 지정된 대상 테이블에서 다양한 규칙이 포함된 규칙 세트를 생성하는 CloudFormation 템플릿입니다.

```
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a DataQualityRuleset
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

  # The name of the ruleset to be created
  RulesetName:
    Type: String
    Default: "CFNRulesetName"
  RulesetDescription:
    Type: String
    Default: "CFN DataQualityRuleset"
  # Rules that will be associated with this ruleset
  Rules:
    Type: String
    Default: 'Rules = [
        RowCount > 100,
        IsUnique "id",
        IsComplete "nametype"
        ]'
  # Name of database and table within Data Catalog which the ruleset will
  # be applied too
  DatabaseName:
    Type: String
    Default: "ExampleDatabaseName"
  TableName:
    Type: String
    Default: "ExampleTableName"
```

```
# Resources section defines metadata for the Data Catalog
Resources:
  # Creates a Data Quality ruleset under specified rules
  DQRuleset:
    Type: AWS::Glue::DataQualityRuleset
    Properties:
      Name: !Ref RulesetName
      Description: !Ref RulesetDescription
      # The String within rules must be formatted in DQDL, a language
      # used specifically to make rules
      Ruleset: !Ref Rules
      # The targeted table must exist within Data Catalog alongside
      # the correct database
      TargetTable:
        DatabaseName: !Ref DatabaseName
        TableName: !Ref TableName
```

EventBridge 스케줄러에서 AWS Glue Data Quality 규칙 세트에 대한 샘플 AWS CloudFormation 템플릿

AWS Glue 데이터 품질 규칙 세트에는 데이터 카탈로그의 테이블에서 평가할 수 있는 규칙이 포함되어 있습니다. 규칙 세트를 대상 테이블에 배치한 후에는 데이터 카탈로그로 이동하여 규칙 세트 내 해당 규칙에 대해 데이터를 실행하는 평가를 실행할 수 있습니다. 규칙 세트를 평가하기 위해 수동으로 데이터 카탈로그로 이동할 필요 없이, CloudFormation 템플릿 내에 EventBridge 스케줄러를 추가하여 일정 간격으로 이러한 규칙 세트 평가를 예약할 수도 있습니다.

다음 샘플은 데이터 품질 규칙 세트와 EventBridge 스케줄러를 생성하여 앞서 언급한 규칙 세트를 5분마다 평가하는 CloudFormation 템플릿입니다.

```
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a DataQualityRuleset
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

  # The name of the ruleset to be created
  RulesetName:
    Type: String
    Default: "CFNRulesetName"
  # Rules that will be associated with this Ruleset
```

```
  Rules:
    Type: String
    Default: 'Rules = [
        RowCount > 100,
        IsUnique "id",
        IsComplete "nametype"
        ]'
  # The name of the Schedule to be created
  ScheduleName:
    Type: String
    Default: "ScheduleDQRulsetEvaluation"
  # This expression determines the rate at which the Schedule will evaluate
  # your data using the above ruleset
  ScheduleRate:
    Type: String
    Default: "rate(5 minutes)"
  # The Request that being sent must match the details of the Data Quality Ruleset
  ScheduleRequest:
    Type: String
    Default: '
        { "DataSource": { "GlueTable": { "DatabaseName": "ExampleDatabaseName",
         "TableName": "ExampleTableName" } },
         "Role": "role/AWSGlueServiceRoleDefault",
          "RulesetNames": [ ""CFNRulesetName"" ] }
        '

# Resources section defines metadata for the Data Catalog
Resources:
  # Creates a Data Quality ruleset under specified rules
  DQRuleset:
    Type: AWS::Glue::DataQualityRuleset
    Properties:
      Name: !Ref RulesetName
      Description: "CFN DataQualityRuleset"
      # The String within rules must be formatted in DQDL, a language
      # used specifically to make rules
      Ruleset: !Ref Rules
      # The targeted table must exist within Data Catalog alongside
      # the correct database
      TargetTable:
        DatabaseName: "ExampleDatabaseName"
        TableName: "ExampleTableName"
  # Create a Scheduler to schedule evaluation runs on the above ruleset
  ScheduleDQEval:
```

```
    Type: AWS::Scheduler::Schedule
    Properties:
      Name: !Ref ScheduleName
      Description: "Schedule DataQualityRuleset Evaluations"
      FlexibleTimeWindow:
        Mode: "OFF"
      ScheduleExpression: !Ref ScheduleRate
      ScheduleExpressionTimezone: "America/New_York"
      State: "ENABLED"
      Target:
        # The ARN is the API that will be run, since we want to evaluate our ruleset
        # we want this specific ARN
        Arn: "arn:aws:scheduler:::aws-sdk:glue:startDataQualityRulesetEvaluationRun"
        # Your RoleArn must have approval to schedule
        RoleArn: "arn:aws:iam::123456789012:role/AWSGlueServiceRoleDefault"
        # This is the Request that is being sent to the Arn
        Input: '
        { "DataSource": { "GlueTable": { "DatabaseName": "sampledb", "TableName":
"meteorite" } },
         "Role": "role/AWSGlueServiceRoleDefault",
          "RulesetNames": [ "TestCFN" ] }
        '
```

AWS Glue 개발 엔드포인트의 샘플 AWS CloudFormation 템플릿

AWS Glue 개발 엔드포인트는 AWS Glue 스크립트를 개발하고 테스트하는 데 사용되는 환경입니다.

이 샘플은 성공적으로 샘플을 생성할 때 필요한 최소한의 네트워크 파라미터 값으로 개발 엔드포인트를 생성합니다. 개발 엔드포인터를 설정할 필요가 있는 파라미터에 대한 자세한 내용은 AWS Glue의 개발에 대한 네트워킹 설정를 참조하십시오.

기존 IAM 역할 ARN(Amazon Resource Name)을 제공하여 개발 엔드포인트를 생성합니다. 노트북 서버를 개발 엔드포인터에 생성하고자 한다면 유효한 RSA 퍼블릭 키를 제공하고 대응하는 프라이빗 키를 사용 가능한 상태로 유지합니다.

> **Note**
>
> 개발 엔드포인트와 연결되어 생성한 어떤 노트북 서버도 관리합니다. 따라서 엔드포인트를 삭제하고자 한다면 AWS CloudFormation 콘솔에서 AWS CloudFormation 스택을 삭제해야 노트북 서버를 삭제할 수 있습니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a development endpoint
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNEndpointName:
    Type: String
    Default: cfn-devendpoint-1
  CFNIAMRoleArn:
    Type: String
    Default: arn:aws:iam::123456789012/role/AWSGlueServiceRoleGA
#
#
# Resources section defines metadata for the Data Catalog
Resources:
  CFNDevEndpoint:
    Type: AWS::Glue::DevEndpoint
    Properties:
      EndpointName: !Ref CFNEndpointName
      #ExtraJarsS3Path: String
      #ExtraPythonLibsS3Path: String
      NumberOfNodes: 5
      PublicKey: ssh-rsa public.....key myuserid-key
      RoleArn: !Ref CFNIAMRoleArn
      SecurityGroupIds:
        - sg-64986c0b
      SubnetId: subnet-c67cccac
```

AWS Glue 프로그래밍 안내서

스크립트는 소스에서 데이터를 추출하고 데이터를 변환한 다음 대상으로 로드하는 스크립트를 실행하는 코드를 포함합니다. AWS Glue는 작업 시작 시 스크립트를 실행합니다.

AWS Glue ETL 스크립트는 Python 또는 Scala에 코딩될 수 있습니다. 모든 작업 유형을 Python으로 작성할 수 있지만 AWS Glue for Spark 작업은 Scala로도 작성할 수 있습니다. AWS Glue Studio에서 작업의 소스 코드 논리를 자동으로 생성하면 스크립트가 생성됩니다. 스크립트를 편집하거나 자체 스크립트를 제공하여 ETL 작업을 실행할 수 있습니다.

사용자 지정 스크립트 제공

스크립트는 AWS Glue에서 추출, 변환 및 로드(ETL) 작업을 수행합니다. 자동적으로 작업의 소스 코드 로직을 생성할 때 스크립트가 생성됩니다. 스크립트를 편집하거나 사용자 지정 스크립트를 제공할 수 있습니다.

AWS Glue의 자체 사용자 지정 스크립트를 제공하려면 다음 일반 절차를 따르십시오.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. ETL 작업 탭을 선택한 다음 작업 생성 섹션을 확인합니다. 스크립트 편집기 옵션을 선택합니다.
3. This job runs(이 작업이 실행됩니다) 아래에서 다음 중 하나를 선택합니다.

   - 표준 문안 코드로 새 스크립트 생성
   - 기존 스크립트 업로드 및 편집
4. 작업 속성 페이지에서 사용자 지정 스크립트를 실행하는 데 필요한 IAM 역할을 선택합니다. 자세한 내용은 AWS Glue의 Identity and Access Management 단원을 참조하십시오.
5. 스크립트가 참조하는 모든 연결을 선택합니다. 이런 객체는 필요한 JDBC 데이터 스토어로 연결이 필요합니다.

   탄력적 네트워크 인터페이스는 Virtual Private Cloud(VPC)에서 인스턴스에 장착할 수 있는 가상 네트워크 인터페이스입니다. 스크립트에 사용된 데이터 스토어로 연결할 때 필요한 탄력적 네트워크 인터페이스를 선택합니다.
6. 작업 유형별 파라미터를 비롯한 추가 구성을 제공합니다. 작업 유형의 구성에 대한 자세한 내용은 AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 섹션을 참조하세요.
7. 스크립트 탭에서 사용자 지정 스크립트를 붙여넣거나 작성합니다.

이 섹션의 콘텐츠를 사용하여 사용자 지정 스크립트를 작성하는 프로세스를 안내합니다.

AWS Glue의 작업 추가에 대한 자세한 내용은 AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 단원을 참조하십시오.

단계별 지침은 AWS Glue 콘솔의 [작업 추가(Add job)] 튜토리얼을 참조하세요.

Spark 스크립트 프로그래밍

AWS Glue를 사용하면 테스트 및 실행 외에도 추출, 변환, 로드 스크립트를 쉽게 작성하거나 자동 생성할 수 있습니다. AWS Glue가 도입한 아파치 스파크의 확장성을 보여주고 Python 혹은 Scala에서 ETL 스크립트를 코딩하고 실행하는 방법의 예를 제공합니다.

> **Important**
>
> AWS Glue 버전에 따라 지원되는 Apache Spark 버전이 다릅니다. 사용자 지정 스크립트는 지원되는 Apache Spark 버전과 호환되어야 합니다. AWS Glue 버전에 대한 자세한 내용은 Glue version job property 섹션을 참조하세요.

주제

- 자습서: AWS Glue for Spark 스크립트 작성
- PySpark의 AWS Glue ETL 스크립트 프로그래밍
- Scala의 AWS Glue ETL 스크립트 프로그래밍
- AWS Glue for Spark ETL 스크립트 프로그래밍을 위한 기능 및 최적화

자습서: AWS Glue for Spark 스크립트 작성

이 자습서에서는 AWS Glue 스크립트 작성 과정을 소개합니다. 작업을 사용하여 일정에 따라 스크립트를 실행하거나, 대화형 세션을 사용하여 대화형으로 실행할 수 있습니다. 작업에 대한 자세한 내용은 AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 섹션을 참조하세요. 대화형 세션에 대한 자세한 내용을 알아보려면 the section called “AWS Glue 대화형 세션 개요”을(를) 참조하세요.

AWS Glue Studio 시각적 편집기는 AWS Glue 작업 빌드를 위한 그래픽의 코드가 없는 인터페이스를 제공합니다. AWS Glue 스크립트는 시각적 작업을 뒷받침합니다. 이를 통해 Apache Spark 프로그램과 함께 작동하는 데 사용할 수 있는 확장된 도구 세트에 액세스할 수 있습니다. 기본 Spark API와 AWS

Glue 스크립트 내에서 추출, 전환, 적재(ETL) 워크플로를 용이하게 하는 AWS Glue 라이브러리에 액세스할 수 있습니다.

이 자습서에서는 주차 티켓 데이터 세트를 추출, 변환 및 로드합니다. 이 작업을 수행하는 스크립트는 AWS Glue Studio 시각적 편집기를 소개하는 AWS Big Data Blog의 Making ETL easier with AWS Glue Studio에서 생성한 스크립트와 형태와 기능이 동일합니다. 작업에서 이 스크립트를 실행하여 시각적 작업과 비교하고 AWS Glue ETL 스크립트가 어떻게 작동하는지 확인할 수 있습니다. 이렇게 하면 시각적 작업에서 아직 사용할 수 없는 추가 기능을 사용할 수 있습니다.

이 자습서에서는 Python 언어 및 라이브러리를 사용합니다. Scala에서도 비슷한 기능을 사용할 수 있습니다. 이 자습서를 진행한 후에는 샘플 Scala 스크립트를 생성하고 검사하여 Scala AWS Glue ETL 스크립트 작성 프로세스를 수행하는 방법을 파악할 수 있습니다.

사전 조건

이 자습서의 사전 요구 사항은 다음과 같습니다.

- AWS CloudFormation 템플릿을 실행하도록 안내하는 AWS Glue Studio 블로그 게시물과 동일한 사전 조건입니다.

  이 템플릿은 AWS Glue 데이터 카탈로그를 사용하여 `s3://aws-bigdata-blog/artifacts/gluestudio/`에서 제공되는 주차 티켓 데이터 세트를 관리합니다. 참조할 다음 리소스를 생성합니다.
- AWS Glue Studio역할 — AWS Glue 작업을 실행할 IAM 역할
- AWS Glue StudioAmazon S3Bucket — 블로그 관련 파일을 저장할 Amazon S3 버킷의 이름
- AWS Glue StudioTicketsYYZDB – AWS Glue 데이터 카탈로그 데이터베이스
- AWS Glue StudioTableTickets — 소스로 사용할 데이터 카탈로그 테이블
- AWS Glue StudioTableTrials — 소스로 사용할 데이터 카탈로그 테이블
- AWS Glue StudioParkingTicketCount – 대상으로 사용할 데이터 카탈로그 테이블
- AWS Glue Studio 블로그 게시물에서 생성된 스크립트입니다. 블로그 게시물이 변경되면 다음 텍스트에서도 스크립트를 사용할 수 있습니다.

샘플 스크립트 생성

AWS Glue Studio 시각적 편집기는 작성하려는 스크립트에 대한 스캐폴드를 생성하는 강력한 코드 생성 도구로 사용할 수 있습니다. 이 도구를 사용하여 샘플 스크립트를 생성합니다.

이 단계를 건너뛰려는 경우, 스크립트가 제공됩니다.

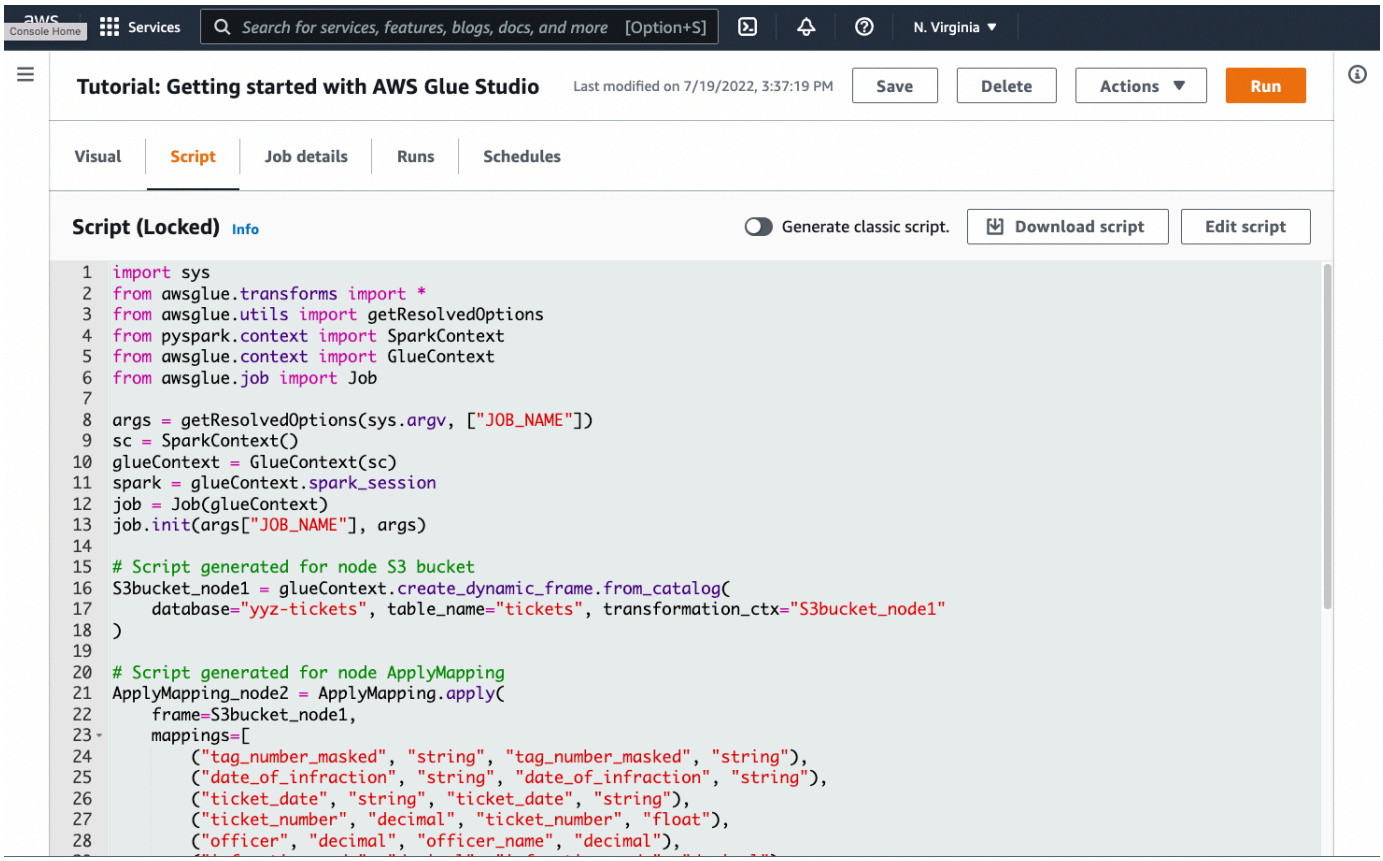
자습서 샘플 스크립트

```
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_catalog(
    database="yyz-tickets", table_name="tickets", transformation_ctx="S3bucket_node1"
)

# Script generated for node ApplyMapping
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3bucket_node1,
    mappings=[
        ("tag_number_masked", "string", "tag_number_masked", "string"),
        ("date_of_infraction", "string", "date_of_infraction", "string"),
        ("ticket_date", "string", "ticket_date", "string"),
        ("ticket_number", "decimal", "ticket_number", "float"),
        ("officer", "decimal", "officer_name", "decimal"),
        ("infraction_code", "decimal", "infraction_code", "decimal"),
        ("infraction_description", "string", "infraction_description", "string"),
        ("set_fine_amount", "decimal", "set_fine_amount", "float"),
        ("time_of_infraction", "decimal", "time_of_infraction", "decimal"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node S3 bucket
S3bucket_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="glueparquet",
    connection_options={"path": "s3://DOC-EXAMPLE-BUCKET", "partitionKeys": []},
```

```
    format_options={"compression": "gzip"},
    transformation_ctx="S3bucket_node3",
)

job.commit()
```

샘플 스크립트 생성

1. AWS Glue Studio 자습서를 완료합니다. 이 자습서를 완료하려면 예제 작업을 통해 AWS Glue Studio에서 작업 생성을 참조하세요.
2. 다음 스크린샷에 나와 있는 것처럼 작업 페이지의 Script(스크립트) 탭으로 이동합니다.



3. Script(스크립트) 탭의 전체 내용을 복사합니다. Job details(작업 세부 정보)에서 스크립트 언어를 설정하여 Python 또는 Scala 코드 생성 간을 전환할 수 있습니다.

1단계. 작업 생성 및 스크립트 붙여넣기

이 단계에서는 AWS Management Console에서 AWS Glue 작업을 생성합니다. 이렇게 하면 AWS Glue에서 스크립트를 실행할 수 있는 구성이 설정됩니다. 또한 스크립트를 저장하고 편집할 수 있는 장소가 생성됩니다.

작업 생성

1. AWS Management Console에서 AWS Glue 랜딩 페이지로 이동합니다.
2. 왼쪽 탐색 창에서 Jobs(작업)을 선택합니다.
3. Create job(작업 생성)에서 Spark script editor(Spark 스크립트 편집기)를 선택한 다음 Create(생성)을 선택합니다.
4. 선택 사항 - 스크립트의 전체 텍스트를 Script(스크립트) 창에 붙여 넣습니다. 자습서를 따라 할 수도 있습니다.

2단계. AWS Glue 라이브러리 가져오기

스크립트 외부에 정의된 코드 및 구성과 상호 작용하도록 스크립트를 설정해야 합니다. 이 작업은 AWS Glue Studio에서 백그라운드로 수행됩니다.

이 단계에서 다음 작업을 수행합니다.

- `GlueContext` 객체를 가져오고 초기화합니다. 이는 스크립트 작성 관점에서 가장 중요한 가져오기입니다. 이를 통해 모든 ETL 스크립트의 시작점인 소스 및 대상 데이터 세트를 정의하기 위한 표준 메서드가 노출됩니다. `GlueContext` 클래스에 대한 자세한 내용을 알아보려면 GlueContext 클래스 섹션을 참조하세요.
- `SparkContext`와 `SparkSession`을 초기화합니다. 이를 통해 AWS Glue 작업 내에서 사용 가능한 Spark 엔진을 구성할 수 있습니다. 입문용 AWS Glue 스크립트 내에서 이러한 Spark 엔진을 직접 사용할 필요는 없습니다.
- `getResolvedOptions`를 호출하여 스크립트 내에서 사용할 작업 인수를 준비합니다. 작업 파라미터 확인에 대한 자세한 내용을 알아보려면 the section called “getResolvedOptions 옵션 ” 섹션을 참조하세요.
- `Job`을 초기화합니다. `Job` 객체는 다양한 선택적 AWS Glue 기능에 대한 구성을 설정하고 상태를 추적합니다. 스크립트는 `Job` 객체 없이 실행될 수 있지만, 이러한 기능이 나중에 통합되는 경우 혼동을 피하기 위해 작업을 초기화하는 것이 좋습니다.

  이러한 기능 중 하나는 이 자습서에서 선택적으로 구성할 수 있는 작업 북마크입니다. the section called “선택 사항 - 작업 북마크 활성화” 섹션에서 작업 북마크에 대해 알아볼 수 있습니다.

이 절차에서는 다음 코드를 작성합니다. 이 코드는 생성된 샘플 스크립트의 일부입니다.

```
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)
```

AWS Glue 라이브러리 가져오기

- 코드의 이 섹션을 복사하여 Script(스크립트) 편집기에 붙여 넣습니다.

> **Note**
>
> 코드 복사는 잘못된 엔지니어링 관행으로 간주될 수도 있습니다. 이 자습서에서는 모든 AWS Glue ETL 스크립트에서 핵심 변수의 이름을 일관되게 지정하기 위해 이를 권장합니다.

3단계. 소스에서 데이터 추출

모든 ETL 프로세스에서 먼저 변경하려는 소스 데이터 세트를 정의해야 합니다. AWS Glue Studio 시각적 편집기에서 Source(소스) 노드를 생성하여 이 정보를 제공합니다.

이 단계에서는 AWS Glue Data Catalog에 구성된 소스에서 데이터를 추출할 `database` 및 `table_name`을(를) `create_dynamic_frame.from_catalog` 메서드에 제공합니다.

전 단계에서 `GlueContext` 객체를 초기화했습니다. 이 객체를 사용하여 소스를 구성하는 데 사용되는 `create_dynamic_frame.from_catalog`와(과) 같은 메서드를 찾습니다.

이 절차에서는 `create_dynamic_frame.from_catalog`를 사용하여 다음 코드를 작성합니다. 이 코드는 생성된 샘플 스크립트의 일부입니다.

```
S3bucket_node1 = glueContext.create_dynamic_frame.from_catalog(
    database="yyz-tickets", table_name="tickets", transformation_ctx="S3bucket_node1"
    )
```

소스에서 데이터 추출

1. 설명서를 검토하여 GlueContext에서 AWS Glue 데이터 카탈로그에 정의된 소스에서 데이터를 추출하는 메서드를 찾습니다. 이러한 메서드는 the section called “GlueContext”에 문서화되어 있습니다. create_dynamic_frame.from_catalog 메서드를 선택합니다. glueContext에서 이 메서드를 호출합니다.
2. 설명서에서 create_dynamic_frame.from_catalog를 검토합니다. 이 메서드에는 database 및 table_name 파라미터가 필요합니다. create_dynamic_frame.from_catalog에 필요한 파라미터를 제공합니다.

   AWS Glue 데이터 카탈로그는 소스 데이터의 위치와 형식에 대한 정보를 저장하며 사전 조건 섹션에서 설정되었습니다. 스크립트에 해당 정보를 직접 제공할 필요는 없습니다.
3. 선택 사항 – 작업 책갈피를 지원하려면 메서드에 transformation_ctx 파라미터를 제공합니다. the section called “선택 사항 - 작업 북마크 활성화” 섹션에서 작업 북마크에 대해 알아볼 수 있습니다.

> **Note**
>
> 데이터를 추출하는 데 일반적으로 사용되는 메서드 the section called “create_dynamic_frame_from_catalog”는 AWS Glue 데이터 카탈로그의 테이블에 연결하는 데 사용됩니다.
> 소스의 구조와 위치를 설명하는 구성을 작업에 직접 제공해야 하는 경우 the section called “create_dynamic_frame_from_options” 메서드를 참조하세요.
> create_dynamic_frame.from_catalog를 사용할 때보다 데이터를 설명하는 더 자세한 파라미터를 제공해야 합니다.
> 필요한 파라미터를 식별하려면 format_options 및 connection_parameters에 대한 보충 문서를 참조하세요. 소스 데이터 형식에 대한 스크립트 정보를 제공하는 방법에 대한 설명을 보려면 the section called “데이터 포맷 옵션” 섹션을 참조하세요. 소스 데이터 위치에 대한 스크립트 정보를 제공하는 방법에 대한 설명을 보려면 the section called “연결 파라미터” 섹션을 참조하세요.
> 스트리밍 소스에서 정보를 읽는 경우 the section called “create_data_frame_from_catalog” 또는 the section called “create_data_frame_from_options” 메서드를 통해 소스 정보를 작업에 제공합니다. 이러한 메서드는 Apache Spark DataFrames를 반환합니다.
> 생성된 코드는 create_dynamic_frame.from_catalog를 호출하는 반면 참조 문서는 create_dynamic_frame_from_catalog를 참조합니다. 이러한 메서드는 최종적으로 동일

한 코드를 호출하며 더 깔끔한 코드를 작성할 수 있도록 포함되어 있습니다. aws-glue-libs에서 제공되는 Python 래퍼의 소스를 보고 이를 확인할 수 있습니다.

4단계. AWS Glue로 데이터 변환

ETL 프로세스에서 소스 데이터를 추출한 후 데이터를 어떻게 변경할지 설명해야 합니다. AWS Glue Studio 시각적 편집기에서 Transform(변환) 노드를 생성하여 이 정보를 제공합니다.

이 단계에서는 `ApplyMapping` 메서드에 현재 필드 이름 및 유형과 원하는 필드 이름 및 유형의 맵을 제공하여 `DynamicFrame`을(를) 변환합니다.

다음 변환을 수행할 수 있습니다.

- `4`개의 `location` 및 `province` 키를 삭제합니다.
- `officer`의 이름을 `officer_name`으로 변경합니다.
- `ticket_number`와 `set_fine_amount`의 유형을 `float`로 변경합니다.

`create_dynamic_frame.from_catalog`가 `DynamicFrame` 객체를 제공합니다. `DynamicFrame`은 AWS Glue의 데이터 세트를 나타냅니다. AWS Glue 변환은 `DynamicFrames`를 변경하는 작업입니다.

> **Note**
>
> `DynamicFrame`란 무엇입니까?
> `DynamicFrame`은 데이터에 있는 항목의 이름 및 유형에 대한 설명과 함께 데이터 세트를 연결할 수 있는 추상화입니다. Apache Spark에는 DataFrame이라는 유사한 추상화가 있습니다. DataFrames에 대한 설명을 보려면 Spark SQL Guide(Spark SQL 가이드)를 참조하세요. `DynamicFrames`을(를) 사용하면 데이터 세트 스키마를 동적으로 설명할 수 있습니다. 일부 항목은 가격을 문자열로 저장하고 다른 항목은 가격을 실수(double)로 저장하는 가격 열이 있는 데이터세트를 예로 들어 보겠습니다. AWS Glue는 즉석에서 스키마를 계산하여 각 행에 대한 자체 설명 레코드를 생성합니다.
> 가격과 같은 불일치 필드는 프레임 스키마에서 유형(`ChoiceType`)으로 명시적으로 표시됩니다. 불일치 필드는 `DropFields`(으)로 삭제하거나 `ResolveChoice`(으)로 확인하여 해결할 수 있습니다. 다음은 `DynamicFrame`에서 사용할 수 있는 변환입니다. 그런 다음 `writeDynamicFrame`을 사용하여 데이터를 데이터 레이크에 다시 쓸 수 있습니다.

DynamicFrame 클래스의 메서드에서 동일한 변환을 많이 호출할 수 있으므로 스크립트를 더 쉽게 읽을 수 있습니다. DynamicFrame에 대한 자세한 정보는 the section called "DynamicFrame" 섹션을 참조하세요.

이 절차에서는 ApplyMapping를 사용하여 다음 코드를 작성합니다. 이 코드는 생성된 샘플 스크립트의 일부입니다.

```
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3bucket_node1,
    mappings=[
        ("tag_number_masked", "string", "tag_number_masked", "string"),
        ("date_of_infraction", "string", "date_of_infraction", "string"),
        ("ticket_date", "string", "ticket_date", "string"),
        ("ticket_number", "decimal", "ticket_number", "float"),
        ("officer", "decimal", "officer_name", "decimal"),
        ("infraction_code", "decimal", "infraction_code", "decimal"),
        ("infraction_description", "string", "infraction_description", "string"),
        ("set_fine_amount", "decimal", "set_fine_amount", "float"),
        ("time_of_infraction", "decimal", "time_of_infraction", "decimal"),
    ],
    transformation_ctx="ApplyMapping_node2",
)
```

AWS Glue로 데이터 변환

1. 설명서를 검토하여 필드를 변경하고 삭제하는 변환을 식별합니다. 자세한 내용은 the section called "GlueTransform"을 참조하세요. ApplyMapping 변환을 선택합니다. ApplyMapping에 대한 자세한 정보는 the section called "ApplyMapping" 섹션을 참조하세요. ApplyMapping 변환 객체에서 apply를 호출합니다.

   > **Note**
   >
   > ApplyMapping란 무엇인가요?
   > ApplyMapping은 DynamicFrame을 가져와서 변환합니다. 이는 필드의 변환을 나타내는 튜플 목록인 '매핑'을 가져옵니다. 처음 두 튜플 요소인 필드 이름과 유형은 프레임에서 필드를 식별하는 데 사용됩니다. 두 번째 두 파라미터는 필드 이름과 유형이기도 합니다. ApplyMapping은 소스 필드를 대상 이름으로 변환하고 반환할 새 DynamicFrame을(를) 입력합니다. 제공되지 않은 필드는 반환 값에서 삭제됩니다.

apply를 호출하는 대신 DynamicFrame에서 apply_mapping 메서드로 동일한 변환을 호출하여 보다 매끄럽고 읽기 쉬운 코드를 생성할 수 있습니다. 자세한 내용은 the section called “apply_mapping” 섹션을 참조하세요.

2. 설명서에서 ApplyMapping을 검토하여 필수 파라미터를 식별합니다. the section called “ApplyMapping”을(를) 참조하세요. 이 메서드에는 frame 및 mappings 파라미터가 필요합니다. ApplyMapping에 필요한 파라미터를 제공합니다.
3. 선택 사항 – 작업 책갈피를 지원하려면 메서드에 transformation_ctx을(를) 제공합니다. the section called “선택 사항 - 작업 북마크 활성화” 섹션에서 작업 북마크에 대해 알아볼 수 있습니다.

> **Note**
>
> Apache Spark 기능
> 작업 내에서 ETL 워크플로를 간소화하기 위한 변환을 제공합니다. 또한 작업 중에 보다 일반적인 용도로 구축된 Spark 프로그램에서 사용할 수 있는 라이브러리에 액세스할 수 있습니다. 이러한 라이브러리를 사용하기 위해 DynamicFrame와(과) DataFrame 간을 변환합니다. the section called “toDF”(으)로 DataFrame을(를) 생성할 수 있습니다. 그런 다음 DataFrame에서 사용 가능한 메서드를 사용하여 데이터 세트를 변환할 수 있습니다. 이러한 메서드에 대한 자세한 내용을 알아보려면 DataFrame을 참조하세요. 그런 다음 the section called “fromDF”(으)로 역방향으로 변환하여 대상에 프레임을 로드하는 데 AWS Glue 작업을 사용할 수 있습니다.

5단계. 대상에 데이터 로드

데이터를 변환한 후 일반적으로 변환된 데이터를 소스와 다른 장소에 저장합니다. AWS Glue Studio 시각적 편집기에서 target(대상) 노드를 생성하여 이 작업을 수행합니다.

이 단계에서는 write_dynamic_frame.from_options 메서드에 connection_type, connection_options, format 및 format_options을(를) 제공하여 Amazon S3의 대상 버킷에 데이터를 로드합니다.

1단계에서 GlueContext 객체를 초기화했습니다. AWS Glue의 경우 여기에서 소스와 마찬가지로 대상을 구성하는 데 사용되는 메서드를 찾을 수 있습니다.

이 절차에서는 write_dynamic_frame.from_options를 사용하여 다음 코드를 작성합니다. 이 코드는 생성된 샘플 스크립트의 일부입니다.

```
S3bucket_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="glueparquet",
    connection_options={"path": "s3://amzn-s3-demo-bucket", "partitionKeys": []},
    format_options={"compression": "gzip"},
    transformation_ctx="S3bucket_node3",
    )
```

대상에 데이터 로드

1. 설명서를 검토하여 대상 Amazon S3 버킷에 데이터를 로드하는 방법을 찾습니다. 이러한 메서드는 the section called “GlueContext”에 문서화되어 있습니다. the section called “write_dynamic_frame_from_options” 메서드를 선택합니다. `glueContext`에서 이 메서드를 호출합니다.

   > **Note**
   >
   > 데이터를 로드하는 데 일반적으로 사용되는 메서드 `write_dynamic_frame.from_options`는 데이터를 로드하는 데 가장 일반적으로 사용되는 메서드로서, AWS Glue에서 사용 가능한 모든 대상을 지원합니다.
   > AWS Glue 연결에 정의된 JDBC 대상에 쓰는 경우 the section called “write_dynamic_frame_from_jdbc_conf” 메서드를 사용합니다. AWS Glue 연결은 데이터 소스에 연결하는 방법에 대한 정보를 저장합니다. 이렇게 하면 해당 정보를 `connection_options`에 제공할 필요가 없습니다. 그러나 `dbtable`을(를) 제공하려면 여전히 `connection_options`을(를) 사용해야 합니다.
   > `write_dynamic_frame.from_catalog`는 데이터를 로드하는 데 일반적으로 사용되는 메서드가 아닙니다. 이 메서드는 기본 데이터 세트를 업데이트하지 않고 AWS Glue 데이터 카탈로그를 업데이트하며 기본 데이터 세트를 변경하는 다른 프로세스와 함께 사용됩니다. 자세한 내용은 the section called “스키마 업데이트 및 새 파티션 추가” 섹션을 참조하세요.

2. 설명서에서 the section called “write_dynamic_frame_from_options”를 검토합니다. 이 메서드에는 `frame`, `connection_type`, `format`, `connection_options` 및 `format_options`이(가) 필요합니다. `glueContext`에서 이 메서드를 호출합니다.

a. 필요한 파라미터를 식별하려면 `format_options`와 `format`에 대한 보충 문서를 참조하세요. 데이터 형식에 대한 설명을 보려면 the section called “데이터 포맷 옵션” 섹션을 참조하세요.

b. 필요한 파라미터를 식별하려면 `connection_type`와 `connection_options`에 대한 보충 문서를 참조하세요. 연결에 대한 설명을 보려면 the section called “연결 파라미터” 섹션을 참조하세요.

c. `write_dynamic_frame.from_options`에 필요한 파라미터를 제공합니다. 이 메서드는 `create_dynamic_frame.from_options`와(과) 유사한 구성을 가지고 있습니다.

3. 선택 사항 – 작업 책갈피를 지원하려면 `write_dynamic_frame.from_options`에 `transformation_ctx`을(를) 제공합니다. the section called “선택 사항 - 작업 북마크 활성화” 섹션에서 작업 북마크에 대해 알아볼 수 있습니다.

6단계. **Job** 객체 커밋

1단계에서 `Job` 객체를 초기화했습니다. 작업 북마크를 사용하는 경우와 같이 특정 선택적 기능이 제대로 작동하려면 스크립트가 끝날 때 수명 주기를 수동으로 종료해야 할 수 있습니다. 이 작업은 AWS Glue Studio에서 백그라운드로 수행됩니다.

이 단계에서는 `Job` 객체에 대한 `commit` 메서드를 호출합니다.

이 절차에서는 다음 코드를 작성합니다. 이 코드는 생성된 샘플 스크립트의 일부입니다.

```
job.commit()
```

Job 객체 커밋

1. 이 단계를 아직 수행하지 않은 경우 이전 섹션에 설명된 선택적 단계를 수행하여 `transformation_ctx`을(를) 포함합니다.
2. `commit`을 호출합니다.

선택 사항 - 작업 북마크 활성화

모든 이전 단계에서 `transformation_ctx` 파라미터를 설정하라는 지시를 받았습니다. 이는 작업 북마크라는 기능과 관련이 있습니다.

작업 북마크를 사용하면 이전 작업을 쉽게 추적할 수 있는 데이터 세트에 대해 반복적으로 실행되는 작업을 통해 시간과 비용을 절약할 수 있습니다. 작업 북마크는 이전 실행의 데이터 세트에서 AWS

Glue 변환의 진행 상황을 추적합니다. AWS Glue는 이전 실행이 끝난 위치를 추적하여 이전에 처리하지 않은 행으로 작업을 제한할 수 있습니다. 작업 북마크에 대한 자세한 내용을 알아보려면 the section called “처리된 데이터를 작업 북마크로 추적”을(를) 참조하세요.

작업 북마크를 활성화하려면 먼저 이전 예제에서 설명한 대로 제공된 함수에 `transformation_ctx` 문을 추가합니다. 작업 북마크 상태는 실행 시에도 유지됩니다. `transformation_ctx` 파라미터는 해당 상태에 액세스하는 데 사용되는 키입니다. 이러한 문은 그 자체로는 아무 것도 하지 않습니다. 작업에 대한 구성에서도 기능을 활성화해야 합니다.

이 절차에서는 AWS Management Console을(를) 사용하여 작업 북마크를 활성화합니다.

작업 북마크 활성화

1. 해당 작업의 Job details(작업 세부 정보) 섹션으로 이동합니다.
2. Job bookmark(작업 북마크)를 Enable(활성화)로 설정합니다.

7단계. 코드를 작업으로 실행

이 단계에서는 작업을 실행하여 이 자습서를 성공적으로 완료했는지 확인합니다. 이 작업은 AWS Glue Studio 시각적 편집기에서와 같이 버튼 클릭으로 수행됩니다.

코드를 작업으로 실행

1. 제목 표시줄에서 Untitled job(제목 없는 작업)을 선택하여 작업 이름을 편집하고 설정합니다.
2. Job details(작업 세부 정보) 탭으로 이동합니다. 작업에 IAM Role(IAM 역할)을 할당합니다. AWS Glue Studio 자습서의 사전 조건에서 AWS CloudFormation 템플릿으로 생성한 역할을 사용할 수 있습니다. 해당 자습서를 완료했다면 `AWS Glue StudioRole`을 사용할 수 있습니다.
3. Save(저장)를 선택하여 스크립트를 저장합니다.
4. Run(실행)을 선택하여 작업을 실행합니다.
5. Runs(실행) 탭으로 이동하여 작업이 완료되었는지 확인합니다.
6. `write_dynamic_frame.from_options`의 대상인 *`amzn-s3-demo-bucket`*으로 이동합니다. 출력이 예상과 일치하는지 확인합니다.

작업 구성 및 관리에 대한 자세한 내용을 알아보려면 the section called “사용자 지정 스크립트 제공”을(를) 참조하세요.

추가 정보

Apache Spark 라이브러리 및 메서드는 AWS Glue 스크립트에서 사용할 수 있습니다. 포함된 라이브러리로 수행할 수 있는 작업을 알아보려면 Spark 설명서를 참조하세요. 자세한 내용을 알아보려면 Spark 소스 리포지토리의 예제 섹션을 참조하세요.

AWS Glue 2.0+에는 기본적으로 몇 가지 일반적인 Python 라이브러리가 포함되어 있습니다. Scala 또는 Python 환경의 AWS Glue 작업에 자체 종속성을 로드하는 메커니즘도 있습니다. Python 종속성에 대한 내용을 알아보려면 the section called “Python 라이브러리”을(를) 참조하세요.

Python에서 AWS Glue 기능을 사용하는 방법에 대한 더 많은 예제를 알아보려면 the section called “Python 샘플”을(를) 참조하세요. Scala 및 Python 작업은 기능 패리티가 있으므로 Python 예제를 통해 Scala에서 비슷한 작업을 수행하는 방법을 알아볼 수 있습니다.

PySpark의 AWS Glue ETL 스크립트 프로그래밍

GitHub 웹 사이트의 AWS Glue 샘플 리포지토리에서 AWS Glue용 Python 코드 예제 및 유틸리티를 찾을 수 있습니다.

AWS Glue와 함께 Python 사용

AWS Glue는 추출, 변환, 로드 작업 스크립트에 대해 PySpark Python의 확장을 지원합니다. 이 섹션에서는 AWS Glue API와 함께 Python의 ETL 스크립트를 사용하는 방법을 설명합니다.

- AWS Glue과 함께 Python을 사용하기 위한 설정
- Python에서 AWS Glue API 호출
- AWS Glue와 함께 Python 라이브러리 사용
- AWS Glue Python 코드 샘플

AWS Glue PySpark 확장

AWS Glue는 다음 확장 기능을 PySpark Python 언어에 생성합니다.

- `getResolvedOptions`를 사용한 파라미터 액세스
- PySpark 확장 유형
- DynamicFrame 클래스
- DynamicFrameCollection 클래스

- DynamicFrameWriter 클래스
- DynamicFrameReader 클래스
- GlueContext 클래스

AWS Glue PySpark 변환

AWS Glue는 PySpark ETL 작업에 다음 변환 클래스를 사용하기 위해 생성됩니다.

- GlueTransform 베이스 클래스
- ApplyMapping 클래스
- DropFields 클래스
- DropNullField 클래스
- ErrorsAsDynamicFrame 클래스
- FillMissingValues 클래스
- Filter 클래스
- FindIncrementalMatches 클래스
- FindMatches 클래스
- FlatMap 클래스
- Join 클래스
- Map 클래스
- MapToCollection 클래스
- mergeDynamicFrame
- Relationalize 클래스
- RenameField 클래스
- ResolveChoice 클래스
- SelectFields 클래스
- SelectFromCollection 클래스
- Spigot 클래스
- SplitFields 클래스
- SplitRows 클래스
- Unbox 클래스

- UnnestFrame 클래스

AWS Glue과 함께 Python을 사용하기 위한 설정

Python을 사용하여 Spark 작업에 대한 ETL 스크립트를 개발합니다. ETL 작업에 대해 지원되는 Python 버전은 작업의 AWS Glue 버전에 따라 다릅니다. AWS Glue 버전에 대한 자세한 내용은 Glue version job property 섹션을 참조하세요.

AWS Glue와 함께 Python을 사용하기 위해 시스템을 설정합니다.

다음 관계에 따라 Python을 설치하고 AWS Glue API를 호출할 수 있도록 합니다.

1. Python을 아직 설치하지 않았다면 Python.org 다운로드 페이지에서 다운로드하여 설치합니다.
2. AWS Command Line Interface(AWS CLI)를 AWS CLI 설명서에 기록하여 설치합니다.

   AWS CLI는 Python을 사용하는 데 직접적으로 필요하지 않습니다. 하지만 Python을 설치하고 구성하는 것은 계정 자격으로 AWS를 설치하고 계정 자격이 효용성이 있는지 확인하는 편리한 방법입니다.
3. AWS SDK for Python(Boto 3)이 Boto3 빠른 시작에 기록되어 있어 설치할 수 있습니다.

   AWS Glue에서는 Boto 3 리소스 API를 사용할 수 없습니다. 현재, Boto 3 사용자 API만 사용할 수 있습니다.

   Boto 3에 대한 자세한 내용은 AWS SDK for Python(Boto3) 시작하기를 참조하세요.

AWS Glue용 Python 코드 예제와 유틸리티는 GitHub 웹 사이트의 AWS Glue 샘플 리포지토리에서 찾을 수 있습니다.

Python에서 AWS Glue API 호출

AWS Glue에서는 Boto 3 리소스 API를 사용할 수 없습니다. 현재, Boto 3 사용자 API만 사용할 수 있습니다.

Python에 있는 AWS Glue API 이름

Java 언어로 된 AWS Glue API 이름과 기타 프로그래밍 언어는 보통 카멜케이스(CamelCased)된 단어(단어들 사이의 간격이 없는 단어)입니다. 하지만 Python이 호출하면 일반 이름이 Python식으로 단어들이 밑줄 문자로 분리된 소문자로 바뀝니다. AWS Glue API 참조 문서에서는 이 Python식 이름이 일반 카멜케이스(CamelCased)된 이름 다음에 괄호 안에 기록됩니다.

그러나 비록 AWS Glue API 이름 자체가 소문자로 변환되었지만 이 파라미터 이름은 대문자로 남겨집니다. 이런 점을 기억해야 하는 이유는 파마리터가 다음 설명처럼 AWS Glue API를 호출할 때 이름에 따라 통과하기 때문입니다.

AWS Glue에서 Python 파라미터 전달 및 액세스

Python은 AWS Glue API를 호출할 경우, 파라미터를 이름에 따라 정확하게 통과하는 것이 최선의 방법입니다. 예:

```
job = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                      Command={'Name': 'glueetl',
                               'ScriptLocation': 's3://my_script_bucket/scripts/
my_etl_script.py'})
```

Python이 작업 구조 또는 JobRun 구조의 ETL 스크립트 논제로 지정한 이름/값의 딕셔너리를 생성한다는 점을 이해하면 도움이 됩니다. Boto 3은 JSON 형식으로 REST API 호출을 거쳐 AWS Glue를 통과합니다. 이런 과정은 스크립트에서 논제로 액세스할 경우 논제의 순서에 의존하지 않습니다.

예를 들어, Python Lambda 관리 함수에서 JobRun을 시작하고자 하고 몇 가지 파라미터를 지정하고자 한다고 가정합니다. 코드는 다음과 같을 것입니다.

```
from datetime import datetime, timedelta

client = boto3.client('glue')

def lambda_handler(event, context):
  last_hour_date_time = datetime.now() - timedelta(hours = 1)
  day_partition_value = last_hour_date_time.strftime("%Y-%m-%d")
  hour_partition_value = last_hour_date_time.strftime("%-H")

  response = client.start_job_run(
               JobName = 'my_test_Job',
               Arguments = {
                 '--day_partition_key':   'partition_0',
                 '--hour_partition_key':  'partition_1',
                 '--day_partition_value':  day_partition_value,
                 '--hour_partition_value': hour_partition_value } )
```

안전하게 ETL에서 이 파라미터로 액세스하기 위해서는 AWS Glue의 getResolvedOptions 함수를 사용하여 이름에 따라 지정하고 딕셔너리 결과로부터 파라미터로 액세스합니다.

```
import sys
```

```
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv,
                          ['JOB_NAME',
                           'day_partition_key',
                           'hour_partition_key',
                           'day_partition_value',
                           'hour_partition_value'])
print "The day partition key is: ", args['day_partition_key']
print "and the day partition value is: ", args['day_partition_value']
```

중첩된 JSON 문자열인 인수를 전달하려는 경우 파라미터 값이 AWS Glue ETL 작업에 전달될 때 해당 값을 유지하려면 작업 실행을 시작하기 전에 파라미터 문자열을 인코딩한 다음, 작업 스크립트를 참조하기 전에 파라미터 문자열을 디코딩해야 합니다. 예를 들어 다음 인수 문자열을 고려하세요.

```
glue_client.start_job_run(JobName = "gluejobname", Arguments={
"--my_curly_braces_string": '{"a": {"b": {"c": [{"d": {"e": 42}}]}}}'
})
```

이 파라미터를 올바르게 전달하려면 인수를 Base64로 인코딩한 문자열로 인코딩해야 합니다.

```
import base64
...
sample_string='{"a": {"b": {"c": [{"d": {"e": 42}}]}}}'
sample_string_bytes = sample_string.encode("ascii")

base64_bytes = base64.b64encode(sample_string_bytes)
base64_string = base64_bytes.decode("ascii")
...
glue_client.start_job_run(JobName = "gluejobname", Arguments={
"--my_curly_braces_string": base64_bytes})
...
sample_string_bytes = base64.b64decode(base64_bytes)
sample_string = sample_string_bytes.decode("ascii")
print(f"Decoded string: {sample_string}")
...
```

예: 작업 생성 및 실행

다음 예제는 Python을 사용하여 AWS Glue API를 호출하여 ETL 작업을 실행할 수 있는 방법을 보여줍니다.

예: 작업을 생성하고 실행하기 위해서

1. AWS Glue 클라이언트의 인스턴스를 만듭니다.

```
import boto3
glue = boto3.client(service_name='glue', region_name='us-east-1',
              endpoint_url='https://glue.us-east-1.amazonaws.com')
```

2. 작업 만들기 다음 코드에서 보여지듯이 `glueetl`를 ETL 명령어로 사용해야 합니다.

```
myJob = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                          Command={'Name': 'glueetl',
                                   'ScriptLocation': 's3://my_script_bucket/
scripts/my_etl_script.py'})
```

3. 이전 단계에서 만든 작업을 새롭게 시작합니다.

```
myNewJobRun = glue.start_job_run(JobName=myJob['Name'])
```

4. 작업 상태 얻기.

```
status = glue.get_job_run(JobName=myJob['Name'], RunId=myNewJobRun['JobRunId'])
```

5. 작업 실행의 현재 상태를 프린트합니다.

```
print(status['JobRun']['JobRunState'])
```

AWS Glue와 함께 Python 라이브러리 사용

AWS Glue를 사용하면 AWS Glue ETL과 함께 사용할 추가 Python 모듈 및 라이브러리를 설치할 수 있습니다.

주제

- requirements.txt를 사용하여 AWS Glue 5.0에 추가 Python 라이브러리 설치
- pip를 사용하여 AWS Glue 2.0 이상에 추가 Python 모듈 설치
- PySpark 네이티브 기능으로 Python 파일 포함
- 시각적 변환을 사용하는 프로그래밍 스크립트
- AWS Glue에서 이미 제공되는 Python 모듈

- 라이브러리 압축하여 포함
- AWS Glue Studio 노트북에서 Python 라이브러리 로드
- 개발 엔드포인트에서 Python 라이브러리 로딩
- Job 혹은 JobRun에서 Python 라이브러리 사용

requirements.txt를 사용하여 AWS Glue 5.0에 추가 Python 라이브러리 설치

AWS Glue 5.0에서는 Python 라이브러리 종속성을 관리하기 위한 defacto-standard `requirements.txt`를 제공할 수 있습니다. 이렇게 하려면 다음 두 가지 작업 파라미터를 제공합니다.

- 키: `--python-modules-installer-option`

  값: `-r`
- 키: `--additional-python-modules`

  값: `s3://path_to_requirements.txt`

AWS Glue 5.0 노드는 처음에 `requirements.txt`에 지정된 python 라이브러리를 로드합니다. 다음은 샘플 requirements.txt입니다.

```
awswrangler==3.9.1
elasticsearch==8.15.1
PyAthena==3.9.0
PyMySQL==1.1.1
PyYAML==6.0.2
pyodbc==5.2.0
pyorc==0.9.0
redshift-connector==2.1.3
scipy==1.14.1
scikit-learn==1.5.2
SQLAlchemy==2.0.36
```

pip를 사용하여 AWS Glue 2.0 이상에 추가 Python 모듈 설치

AWS Glue는 Python 패키지 설치 프로그램(pip3)을 사용하여 AWS Glue ETL에서 사용할 추가 모듈을 설치합니다. `--additional-python-modules` 파라미터를 쉼표로 구분된 Python 모듈 목록과 함께 사용하여 새 모듈을 추가하거나 기존 모듈의 버전을 변경할 수 있습니다. Amazon S3에 배포를 업로드

하여 라이브러리의 사용자 지정 배포를 설치한 다음, 모듈 목록에 Amazon S3 객체의 경로를 포함시킬 수 있습니다.

`--python-modules-installer-option` 파라미터를 사용하여 pip3에 추가 옵션을 전달할 수 있습니다. 예를 들어, `"--upgrade"`를 전달하여 `"--additional-python-modules"`로 지정된 패키지를 업그레이드할 수 있습니다. 더 많은 예제를 보려면 Building Python modules from a wheel for Spark ETL workloads using AWS Glue 2.0을 참조하세요.

Python 종속성이 컴파일된 네이티브 코드에 전이적으로 종속되는 경우 AWS Glue는 작업 환경에서 네이티브 코드 컴파일을 지원하지 않는다는 제한과 충돌할 수 있습니다. 그러나 AWS Glue 작업은 Amazon Linux 2 환경 내에서 실행됩니다. Wheel 배포판을 통해 컴파일된 형식으로 네이티브 종속성을 제공할 수 있습니다.

예를 들어 새 `scikit-learn` 모듈을 업데이트하거나 추가하려면 키/값 `"--additional-python-modules", "scikit-learn==0.21.3"`을 사용합니다.

또한 `--additional-python-modules` 옵션 내에서 Python 휠 모듈에 대한 Amazon S3 경로를 지정할 수 있습니다. 예시:

```
--additional-python-modules s3://aws-glue-native-spark/tests/j4.2/ephem-3.7.7.1-cp37-cp37m-linux_x86_64.whl,s3://aws-glue-native-spark/tests/j4.2/fbprophet-0.6-py3-none-any.whl,scikit-learn==0.21.3
```

AWS Glue 콘솔의 Job parameters(작업 파라미터) 필드를 사용하거나 AWS SDK에서 작업 인수를 변경하여 `--additional-python-modules`를 지정합니다. 작업 파라미터 설정에 대한 자세한 내용을 알아보려면 the section called “작업 파라미터” 섹션을 참조하세요.

PySpark 네이티브 기능으로 Python 파일 포함

AWS Glue는 PySpark를 사용하여 AWS Glue ETL 작업에 Python 파일을 포함합니다. 사용 가능한 경우 종속성을 관리하기 위해 `--additional-python-modules`를 사용하고 싶을 것입니다. `--extra-py-files` 작업 파라미터를 사용하여 Python 파일을 포함할 수 있습니다. 종속성은 Amazon S3에서 호스팅되어야 하며, 인수 값은 공백 없이 쉼표로 구분된 Amazon S3 경로 목록이어야 합니다. 이 기능은 Spark에서 사용하는 Python 종속성 관리처럼 작동합니다. Spark의 Python 종속성 관리에 대한 자세한 내용을 알아보려면 Apache Spark 설명서의Using PySpark Native Features(PySpark 기본 기능 사용) 페이지를 참조하세요. `--extra-py-files`는 추가 코드가 패키징되지 않은 경우 또는 종속성 관리를 위해 기존 도구 체인으로 Spark 프로그램을 마이그레이션하는 경우에 유용합니다. 종속성 도구를 유지 관리하려면 제출하기 전에 종속성을 번들로 묶어야 합니다.

시각적 변환을 사용하는 프로그래밍 스크립트

AWS Glue Studio 시각적 인터페이스를 사용하여 AWS Glue 작업을 생성하면 관리형 데이터 변환 노드와 사용자 지정 시각적 변환을 사용하여 데이터를 변환할 수 있습니다. 관리형 데이터 변환 노드에 대한 자세한 내용은 the section called “AWS Glue 관리형 변환으로 데이터 변환” 섹션을 참조하세요. 사용자 지정 시각적 변환에 대한 자세한 내용은 the section called “ 사용자 지정 시각적 변환으로 데이터 변환 ” 섹션을 참조하세요. 시각적 변환을 사용하는 스크립트는 작업 언어가 Python 사용으로 설정된 경우에만 생성할 수 있습니다.

시각적 변환을 사용하여 AWS Glue 작업을 생성할 때 AWS Glue Studio는 작업 구성의 `--extra-py-files` 파라미터를 사용하여 런타임 환경에 이러한 변환을 포함합니다. 작업 파라미터에 대한 자세한 내용을 알아보려면 the section called “작업 파라미터” 섹션을 참조하세요. 생성된 스크립트 또는 런타임 환경을 변경할 때 스크립트가 성공적으로 실행되도록 하려면 이 작업 구성을 유지해야 합니다.

AWS Glue에서 이미 제공되는 Python 모듈

이러한 제공된 모듈의 버전을 변경하려면 새 버전에 `--additional-python-modules` 작업 파라미터를 제공합니다.

AWS Glue version 5.0

AWS Glue 버전 5.0은 기본적으로 다음과 같은 Python 모듈을 포함합니다.

- aiobotocore==2.13.1
- aiohappyeyeballs==2.3.5
- aiohttp==3.10.1
- aioitertools==0.11.0
- aiosignal==1.3.1
- appdirs==1.4.4
- attrs==24.2.0
- boto3==1.34.131
- botocore==1.34.131
- certifi==2024.7.4
- charset-normalizer==3.3.2
- contourpy==1.2.1

- cycler==0.12.1
- fonttools==4.53.1
- frozenlist==1.4.1
- fsspec==2024.6.1
- idna==2.10
- jmespath==0.10.0
- kaleido==0.2.1
- kiwisolver==1.4.5
- matplotlib==3.9.0
- multidict==6.0.5
- numpy==1.26.4
- packaging==24.1
- pandas==2.2.2
- pillow==10.4.0
- pip==22.3.1
- plotly==5.23.0
- pyarrow==17.0.0
- pyparsing==3.1.2
- python-dateutil==2.9.0.post0
- pytz==2024.1
- requests==2.32.2
- s3fs==2024.6.1
- s3transfer==0.10.2
- seaborn==0.13.2
- setuptools==59.6.0
- six==1.16.0
- tenacity==9.0.0
- tzdata==2024.1

- urllib3==1.25.10
- virtualenv==20.4.0
- wrapt==1.16.0
- yarl==1.9.4

AWS Glue version 4.0

AWS Glue 버전 4.0은 기본적으로 다음과 같은 Python 모듈을 포함합니다.

- aiobotocore==2.4.1
- aiohttp==3.8.3
- aioitertools==0.11.0
- aiosignal==1.3.1
- async-timeout==4.0.2
- asynctest==0.13.0
- attrs==22.2.0
- avro-python3==1.10.2
- boto3==1.24.70
- botocore==1.27.59
- certifi==2021.5.30
- chardet==3.0.4
- charset-normalizer==2.1.1
- click==8.1.3
- cycler==0.10.0
- Cython==0.29.32
- docutils==0.17.1
- enum34==1.1.10
- frozenlist==1.3.3
- fsspec==2021.8.1
- idna==2.10

- importlib-metadata==5.0.0
- jmespath==0.10.0
- joblib==1.0.1
- kaleido==0.2.1
- kiwisolver==1.4.4
- matplotlib==3.4.3
- mpmath==1.2.1
- multidict==6.0.4
- nltk==3.7
- numpy==1.23.5
- packaging==23.0
- pandas==1.5.1
- patsy==0.5.1
- Pillow==9.4.0
- pip==23.0.1
- plotly==5.16.0
- pmdarima==2.0.1
- ptvsd==4.3.2
- pyarrow==10.0.0
- pydevd==2.5.0
- pyhocon==0.3.58
- PyMySQL==1.0.2
- pyparsing==2.4.7
- python-dateutil==2.8.2
- pytz==2021.1
- PyYAML==6.0.1
- regex==2022.10.31
- requests==2.23.0

- s3fs==2022.11.0
- s3transfer==0.6.0
- scikit-learn==1.1.3
- scipy==1.9.3
- setuptools==49.1.3
- six==1.16.0
- statsmodels==0.13.5
- subprocess32==3.5.4
- sympy==1.8
- tbats==1.1.0
- threadpoolctl==3.1.0
- tqdm==4.64.1
- typing_extensions==4.4.0
- urllib3==1.25.11
- wheel==0.37.0
- wrapt==1.14.1
- yarl==1.8.2
- zipp==3.10.0

AWS Glue version 3.0

AWS Glue 버전 3.0은 기본적으로 다음과 같은 Python 모듈을 포함합니다.

- aiobotocore==1.4.2
- aiohttp==3.8.3
- aioitertools==0.11.0
- aiosignal==1.3.1
- async-timeout==4.0.2
- asynctest==0.13.0
- attrs==22.2.0

- avro-python3==1.10.2
- boto3==1.18.50
- botocore==1.21.50
- certifi==2021.5.30
- chardet==3.0.4
- charset-normalizer==2.1.1
- click==8.1.3
- cycler==0.10.0
- Cython==0.29.4
- docutils==0.17.1
- enum34==1.1.10
- frozenlist==1.3.3
- fsspec==2021.8.1
- idna==2.10
- importlib-metadata==6.0.0
- jmespath==0.10.0
- joblib==1.0.1
- kiwisolver==1.3.2
- matplotlib==3.4.3
- mpmath==1.2.1
- multidict==6.0.4
- nltk==3.6.3
- numpy==1.19.5
- packaging==23.0
- pandas==1.3.2
- patsy==0.5.1
- Pillow==9.4.0
- pip==23.0
- pmdarima==1.8.2

- ptvsd==4.3.2
- pyarrow==5.0.0
- pydevd==2.5.0
- pyhocon==0.3.58
- PyMySQL==1.0.2
- pyparsing==2.4.7
- python-dateutil==2.8.2
- pytz==2021.1
- PyYAML==5.4.1
- regex==2022.10.31
- requests==2.23.0
- s3fs==2021.8.1
- s3transfer==0.5.0
- scikit-learn==0.24.2
- scipy==1.7.1
- six==1.16.0
- Spark==1.0
- statsmodels==0.12.2
- subprocess32==3.5.4
- sympy==1.8
- tbats==1.1.0
- threadpoolctl==3.1.0
- tqdm==4.64.1
- typing_extensions==4.4.0
- urllib3==1.25.11
- wheel==0.37.0
- wrapt==1.14.1
- yarl==1.8.2

- zipp==3.12.0

AWS Glue version 2.0

AWS Glue 버전 2.0은 기본적으로 다음과 같은 Python 모듈을 포함합니다.

- avro-python3==1.10.0
- awscli==1.27.60
- boto3==1.12.4
- botocore==1.15.4
- certifi==2019.11.28
- chardet==3.0.4
- click==8.1.3
- colorama==0.4.4
- cycler==0.10.0
- Cython==0.29.15
- docutils==0.15.2
- enum34==1.1.9
- fsspec==0.6.2
- idna==2.9
- importlib-metadata==6.0.0
- jmespath==0.9.4
- joblib==0.14.1
- kiwisolver==1.1.0
- matplotlib==3.1.3
- mpmath==1.1.0
- nltk==3.5
- numpy==1.18.1
- pandas==1.0.1
- patsy==0.5.1

- pmdarima==1.5.3
- ptvsd==4.3.2
- pyarrow==0.16.0
- pyasn1==0.4.8
- pydevd==1.9.0
- pyhocon==0.3.54
- PyMySQL==0.9.3
- pyparsing==2.4.6
- python-dateutil==2.8.1
- pytz==2019.3
- PyYAML==5.3.1
- regex==2022.10.31
- requests==2.23.0
- rsa==4.7.2
- s3fs==0.4.0
- s3transfer==0.3.3
- scikit-learn==0.22.1
- scipy==1.4.1
- setuptools==45.2.0
- six==1.14.0
- Spark==1.0
- statsmodels==0.11.1
- subprocess32==3.5.4
- sympy==1.5.1
- tbats==1.0.9
- tqdm==4.64.1
- typing-extensions==4.4.0
- urllib3==1.25.8
- wheel==0.35.1

- zipp==3.12.0

라이브러리 압축하여 포함

라이브러리가 하나의 `.py` 파일로 포함되지 않는 한 `.zip` 아카이브로 패킹되어야 합니다. 패키지 디렉터리는 아가이브의 루트에 있어야 하고 패키지용 `__init__.py` 파일을 포함해야 합니다. 그러면 Python은 정상적인 방법으로 패키지를 가져올 수 있습니다.

라이브러리가 하나의 `.py` 파일에 하나의 Python 모듈만 구성할 수 있으면 모듈을 `.zip`에 넣을 필요가 없습니다.

AWS Glue Studio 노트북에서 Python 라이브러리 로드

AWS Glue Studio 노트북에서 Python 라이브러리를 지정하려면 추가 Python 모듈 설치를 참조하세요.

개발 엔드포인트에서 Python 라이브러리 로딩

다른 ETL 스크립트를 설치하기 위해 다른 라이브러리를 사용할 경우, 각 세트에 따라 개별 개발 엔드포인트를 설치하거나 개발 엔드포인트가 스크립트를 스위치할 때마다 로딩되는 라이브러리 `.zip` 파일(들)을 덮어쓰기할 수 있습니다.

콘솔을 사용하여 개발 엔드포인트를 생성할 때 개발 엔드포인트를 위한 하나 이상의 .zip 파일을 지정할 수 있습니다. 이름과 IAM 역할을 할당한 후 [스크립트 라이브러리 및 작업 파라미터(선택 사항)(Script Libraries and job parameters (optional))]를 선택하고 [Python 라이브러리 경로(Python library path)] 상자에 라이브러리 `.zip` 파일에 대한 전체 Amazon S3 경로를 입력합니다. 예:

```
s3://bucket/prefix/site-packages.zip
```

원한다면, 다음과 같이 빈 칸없이 쉼표로만 나뉜 파일까지의 전체 경로를 지정할 수 있습니다.

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

이런 `.zip` 파일을 나중에 업데이트하면 콘솔을 사용하여 개발 엔드포인트로 이러 파일을 다시 가져올 수 있습니다. 해당 개발자 엔드포인트로 이동하여 옆에 있는 확인란을 선택한 다음 [작업(Action)] 메뉴에서 [ETL 라이브러리 업데이트(Update ETL libraries)]를 선택합니다.

비슷한 방식으로 AWS Glue API를 사용하여 라이브러리 파일을 지정할 수 있습니다. CreateDevEndpoint 작업(Python: create_dev_endpoint)를 호출하여 개발 엔드포인트를 생성하고자 할 때 다음과 같은 호출에 따라 `ExtraPythonLibsS3Path` 파라미터의 라이브러리까지 하나 이상의 전체 경로를 지정할 수 있습니다.

```
dep = glue.create_dev_endpoint(
             EndpointName="testDevEndpoint",
             RoleArn="arn:aws:iam::123456789012",
             SecurityGroupIds="sg-7f5ad1ff",
             SubnetId="subnet-c12fdba4",
             PublicKey="ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCtp04H/y...",
             NumberOfNodes=3,
             ExtraPythonLibsS3Path="s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/
lib_X.zip")
```

개발 엔드포인트를 업데이트할 때 DevEndpointCustomLibraries 객체를 사용하고 UpdateDevEndpoint(update_dev_endpoint)를 호출할 때 `UpdateEtlLibraries` 파라미터를 `True`로 설정하여 로드하는 라이브러리를 업데이트할 수도 있습니다.

Job 혹은 JobRun에서 Python 라이브러리 사용

콘솔에서 새로운 작업을 생성하고자 할 경우, [스크립트 라이브러리 및 작업 파라미터(선택 사항)(Script Libraries and job parameters (optional))]를 선택하고 개발 엔드포인트를 생성할 때와 동일한 방식으로 전체 Amazon S3 라이브러리 경로를 입력하여 라이브러리 .zip 파일을 하나 이상 지정할 수 있습니다.

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

CreateJob(create_job)를 호출한다면 다음과 같은 `--extra-py-files` 기본 파라미터를 사용하여 기본 라이브러리까지 하나 이상의 전체 경로를 지정할 수 있습니다.

```
job = glue.create_job(Name='sampleJob',
                      Role='Glue_DefaultRole',
                      Command={'Name': 'glueetl',
                               'ScriptLocation': 's3://my_script_bucket/scripts/
my_etl_script.py'},
                      DefaultArguments={'--extra-py-files': 's3://bucket/prefix/
lib_A.zip,s3://bucket_B/prefix/lib_X.zip'})
```

그런 다음 JobRun을 시작할 경우, 다른 것으로 기본 라이브러리 설정을 재정의할 수 있습니다.

```
runId = glue.start_job_run(JobName='sampleJob',
                           Arguments={'--extra-py-files': 's3://bucket/prefix/
lib_B.zip'})
```

AWS Glue Python 코드 샘플

- 코드 예: 데이터 조인 및 관계화
- 코드 예: ResolveChoice, Lambda, 및 ApplyMapping을 사용한 데이터 준비

코드 예: 데이터 조인 및 관계화

이 예제에서는 http://everypolitician.org/에서 Amazon Simple Storage Service(Amazon S3)(`s3://awsglue-datasets/examples/us-legislators/all`)의 `sample-dataset` 버킷으로 다운로드한 데이터 집합을 사용합니다. 데이터 세트에는 미국 입법 기관과 미 상원 및 하원에서 확보한 의석에 대한 JSON 형식의 데이터가 포함되어 있으며, 이 자습서의 용도에 맞게 퍼블릭 Amazon S3 버킷에서 사용할 수 있도록 데이터 세트가 약간 수정되었습니다.

이 예제의 소스 코드는 GitHub 웹 사이트의 AWS Glue 샘플 리포지토리에 있는 `join_and_relationalize.py` 파일에서 찾을 수 있습니다.

이 데이터를 사용하여 이 자습서에서는 다음을 수행하는 방법을 보여줍니다.

- AWS Glue 크롤러를 사용하여 퍼블릭 Amazon S3 버킷에 저장된 객체를 분류하고 AWS Glue Data Catalog에 객체의 스키마를 저장합니다.
- 크롤의 결과인 테이블 메타데이터 및 스키마를 검토합니다.
- Data Catalog의 메타데이터를 사용하여 Python 추출, 변환, 로드(ETL) 스크립트를 작성하고 다음을 실행합니다.
  - 하나의 데이터 테이블로 다른 원본 파일의 데이터를 모읍니다 (즉, 데이터를 비정규화합니다).
  - 제정자 유형에 따라 모아진 테이블을 개별 테이블로 필터링합니다.
  - 다음 분석을 위해 결과 데이터를 Apache Parquet 파일을 작성합니다.

AWS에서 실행하는 중에 Python 또는 PySpark 스크립트를 디버깅하는 기본적인 방법은 AWS Glue Studio에서 노트북을 사용하는 것입니다.

1단계: Amazon S3 버킷에서 데이터 크롤

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 크롤러 구성의 단계에 따라 AWS Glue Data Catalog에서 `legislators`라는 데이터베이스로 `s3://awsglue-datasets/examples/us-legislators/all` 데이터 집합을 크롤링할 수 있는 새 크롤러를 만듭니다. 이 예제 데이터는 이 퍼블릭 Amazon S3 버킷에 이미 존재합니다.

3. 새로운 크롤러를 실행한 다음 `legislators` 데이터베이스를 확인합니다.

   크롤러는 다음 메타데이터 테이블을 생성합니다.

   - `persons_json`
   - `memberships_json`
   - `organizations_json`
   - `events_json`
   - `areas_json`
   - `countries_r_json`

   제정자와 제정자 기록을 포함한 테이블 반정규화된 컬렉션입니다.

2단계: 개발 엔드포인트 노트북에 표준 문안 스크립트 추가

다음 표준 문안 스크립트를 개발 엔드포인트 노트북에 복사하고 필요한 AWS Glue 라이브러리로 들여와 단일 `GlueContext`를 설정합니다.

```
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

glueContext = GlueContext(SparkContext.getOrCreate())
```

3단계: Data Catalog의 데이터에서 스키마 검토

다음으로 AWS Glue Data Catalog에서 검사 DynamicFrame을 쉽게 생성하고 데이터의 스키마를 검사할 수 있습니다. 예를 들어, `persons_json` 테이블의 스키마를 보고 노트북에 다음을 추가합니다.

```
persons = glueContext.create_dynamic_frame.from_catalog(
             database="legislators",
             table_name="persons_json")
print "Count: ", persons.count()
persons.printSchema()
```

프린트 호출에 따른 출력값입니다.

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

테이블에 있는 각자는 미국 의회 기관의 멤버입니다.

memberships_json 테이블의 스키마를 보고 다음을 입력합니다.

```
memberships = glueContext.create_dynamic_frame.from_catalog(
                database="legislators",
                table_name="memberships_json")
```

```
print "Count: ", memberships.count()
memberships.printSchema()
```

출력값은 다음과 같습니다.

```
Count:  10439
root
|-- area_id: string
|-- on_behalf_of_id: string
|-- organization_id: string
|-- role: string
|-- person_id: string
|-- legislative_period_id: string
|-- start_date: string
|-- end_date: string
```

organizations는 정당이고 미국의 상원과 하원인 의회 양원입니다. organizations_json 테이블의 스키마를 보고 다음을 입력합니다.

```
orgs = glueContext.create_dynamic_frame.from_catalog(
           database="legislators",
           table_name="organizations_json")
print "Count: ", orgs.count()
orgs.printSchema()
```

출력값은 다음과 같습니다.

```
Count:  13
root
|-- classification: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
```

```
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- name: string
|-- seats: int
|-- type: string
```

4단계: 데이터 필터링

원하는 필드만 남기고 id를 org_id로 이름을 바꿉니다. 데이터셋은 전체를 볼 수 있을만큼 작습니다.

toDF()는 DynamicFrame를 Apache Spark DataFrame로 변환하기 때문에 Apache Spark SQL에 이미 존재한 변환을 적용할 수 있습니다.

```
orgs = orgs.drop_fields(['other_names',
                         'identifiers']).rename_field(
                             'id', 'org_id').rename_field(
                               'name', 'org_name')
orgs.toDF().show()
```

다음은 출력값을 보여줍니다.

```
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
|classification|              org_id|            org_name|               links|seats|
     type|               image|
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
|         party|            party/al|                  AL|                null| null|
     null|                null|
|         party|      party/democrat|            Democrat|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/democrat-li...|    Democrat-Liberal|[[website,http://...| null|
     null|                null|
|   legislature|d56acebe-8fdc-47b...|House of Represen...|                null|  435|
lower house|                null|
|         party|   party/independent|         Independent|                null| null|
     null|                null|
```

```
|         party|party/new_progres...|     New Progressive|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/popular_dem...|    Popular Democrat|[[website,http://...| null|
     null|                null|
|         party|    party/republican|          Republican|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/republican-...|Republican-Conser...|[[website,http://...| null|
     null|                null|
|         party|      party/democrat|            Democrat|[[website,http://...| null|
     null|https://upload.wi...|
|         party|   party/independent|         Independent|                null| null|
     null|                null|
|         party|    party/republican|          Republican|[[website,http://...| null|
     null|https://upload.wi...|
|   legislature|8fa6c3d2-71dc-478...|              Senate|                null|  100|
upper house|                null|
+--------------+--------------------+--------------------+--------------------+-----
+----------+--------------------+
```

memberships에 나타나는 organizations를 보려면 다음을 입력합니다.

```
memberships.select_fields(['organization_id']).toDF().distinct().show()
```

다음은 출력값을 보여줍니다.

```
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

5단계: 데이터 조인

이제 AWS Glue를 사용하여 이런 관계형 테이블을 조인하고 제정자 memberships의 전체 기록 테이블과 제정자에 대응하는 organizations을 생성합니다.

1. 우선, persons와 memberships를 id와 person_id로 조인합니다.
2. 다음으로 org_id 및 organization_id에서 orgs와 결과를 조인합니다.

3. 그 후, 중복 필드, person_id 및 org_id을 드롭합니다.

이 모든 작업을 하나 줄의 (확장된) 코드로 실행할 수 있습니다.

```
l_history = Join.apply(orgs,
                       Join.apply(persons, memberships, 'id', 'person_id'),
                       'org_id', 'organization_id').drop_fields(['person_id',
 'org_id'])
print "Count: ", l_history.count()
l_history.printSchema()
```

출력값은 다음과 같습니다.

```
Count:  10439
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
|-- birth_date: string
```

```
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- death_date: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- family_name: string
|-- id: string
|-- start_date: string
|-- end_date: string
```

이제, 분석을 위한 최종 테이블이 있습니다. AWS Glue, Amazon Athena 또는 Amazon Redshift Spectrum에서 SQL을 실행할 수 있는 압축되고 효율적인 분석 형식(즉, Parquet)으로 작성할 수 있습니다.

다음 호출은 여러 파일의 테이블을 작성하여 나중에 분석을 할 때 빠른 병렬 판독을 지원합니다.

```
glueContext.write_dynamic_frame.from_options(frame = l_history,
          connection_type = "s3",
          connection_options = {"path": "s3://glue-sample-target/output-dir/
legislator_history"},
          format = "parquet")
```

하나의 파일에 기록 데이터 모두를 넣으면 데이터 프레임으로 변환하고 다시 분할하며 작성할 수 있습니다.

```
s_history = l_history.toDF().repartition(1)
s_history.write.parquet('s3://glue-sample-target/output-dir/legislator_single')
```

상원과 하원으로 나누고자 한다면

```
l_history.toDF().write.parquet('s3://glue-sample-target/output-dir/legislator_part',
                               partitionBy=['org_name'])
```

6단계: 관계형 데이터베이스를 위한 데이터 변환

AWS Glue를 사용하면 반정형 데이터가 포함된, Amazon Redshift와 같은 관계형 데이터베이스에도 데이터를 쉽게 쓸 수 있습니다. 프레임의 객체가 아무리 복잡하더라도 DynamicFrames를 평면화하는 변환 relationalize를 제공합니다.

이 예제에 l_history DynamicFrame를 사용하여 루트 테이블의 이름(hist_root)과 임시 작업 경로를 relationalize로 넘깁니다. 이렇게 하면 DynamicFrameCollection가 반환됩니다. 컬렉션의 DynamicFrames의 이름을 열거할 수 있습니다.

```
dfc = l_history.relationalize("hist_root", "s3://glue-sample-target/temp-dir/")
dfc.keys()
```

다음은 이 keys 호출의 출력입니다.

```
[u'hist_root', u'hist_root_contact_details', u'hist_root_links',
 u'hist_root_other_names', u'hist_root_images', u'hist_root_identifiers']
```

Relationalize는 DynamicFrame의 각 객체의 기록을 포함한 루트 테이블과 배열의 보조 테이블을 포함한 6 개의 새로운 테이블로 기록 테이블을 나눕니다. 관계형 데이터베이스에서 처리한 배열은 보통 차선책일 수 있습니다. 특히 배열이 커지면 그렇습니다. 배열을 다른 테이블로 나누면 쿼리 과정이 빨라집니다.

이제, contact_details를 보고 나누는 과정에 대해 알아봅니다.

```
l_history.select_fields('contact_details').printSchema()
dfc.select('hist_root_contact_details').toDF().where("id = 10 or id =
 75").orderBy(['id','index']).show()
```

다음은 이 show 호출의 출력입니다.

```
root
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
```

```
|    |    |-- value: string
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
| 10|    0|                    fax|                       |
| 10|    1|                       |           202-225-1314|
| 10|    2|                  phone|                       |
| 10|    3|                       |           202-225-3772|
| 10|    4|                twitter|                       |
| 10|    5|                       |        MikeRossUpdates|
| 75|    0|                    fax|                       |
| 75|    1|                       |           202-225-7856|
| 75|    2|                  phone|                       |
| 75|    3|                       |           202-225-2711|
| 75|    4|                twitter|                       |
| 75|    5|                       |              SenCapito|
+---+-----+-----------------------+-----------------------+
```

contact_details 필드는 기존 DynamicFrame의 구조 배열이었습니다. 이 배열에 있는 각 요소가 index에 따라 보조 테이블의 별도 행입니다. 여기서 id는 contact_details 키가 있는 hist_root 테이블의 외래 키입니다.

```
dfc.select('hist_root').toDF().where(
    "contact_details = 10 or contact_details = 75").select(
       ['id', 'given_name', 'family_name', 'contact_details']).show()
```

출력값은 다음과 같습니다.

```
+-------------------+----------+-----------+---------------+
|                 id|given_name|family_name|contact_details|
+-------------------+----------+-----------+---------------+
|f4fc30ee-7b42-432...|      Mike|       Ross|             10|
|e3c60f34-7d1b-4c0...|   Shelley|     Capito|             75|
+-------------------+----------+-----------+---------------+
```

이 명령어에는 toDF()와 where 순으로 선택하여 보고자 하는 열을 필터링합니다.

따라서 보조 테이블과 함께 hist_root을 조인하여 다음과 같은 작업을 수행할 수 있습니다.

- 배열 지원 없이 데이터베이스로 데이터를 로드합니다.

- SQL을 사용하여 배열에서 개별 항목을 쿼리합니다.

AWS Glue 연결을 사용하여 Amazon Redshift 자격 증명을 안전하게 저장하고 액세스합니다. 자체 연결을 만드는 방법에 대한 자세한 내용은 데이터에 연결을 참조하십시오.

이제 DynamicFrames를 차례로 순환하여 데이터를 연결에 쓸 준비가 되었습니다.

```
for df_name in dfc.keys():
  m_df = dfc.select(df_name)
  print "Writing to table: ", df_name
  glueContext.write_dynamic_frame.from_jdbc_conf(frame = m_df, connection settings
 here)
```

연결 설정은 관계형 데이터베이스 유형에 따라 달라집니다.

- Amazon Redshift에 쓰는 방법에 대한 지침은 the section called “Redshift 연결” 섹션을 참조하세요.
- 다른 데이터베이스의 경우 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.

결론

전체적으로 AWS Glue는 유연성이 뛰어납니다. 보통 며칠을 걸려 작성해야 가능한 이 작업을 단 몇 줄의 코드로 얻을 수 있습니다. GitHub의 AWS Glue 샘플에 있는 Python 파일 join_and_relationalize.py에서 전체 소스-대상 ETL 스크립트를 찾을 수 있습니다.

코드 예: ResolveChoice, Lambda, 및 ApplyMapping을 사용한 데이터 준비

이 예에서 사용된 데이터 집합은 두 Data.CMS.gov 데이터 집합("상위 100개 진단 관련 그룹에 대한 입원 환자 예상 지불 시스템 제공자 요약 - FY2011" 및 "FY2011 입원 환자 비용 데이터")에서 다운로드한 Medicare Provider 지불 데이터로 구성됩니다. 데이터 다운로드 후 데이터 집합을 수정하여 파일 끝에 몇 가지 잘못된 기록을 소개합니다. 이 수정된 파일은 s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv의 퍼블릭 Amazon S3 버킷에 있습니다.

이 예제에 대한 소스 코드는 AWS Glue 예제 GitHub 리포지토리의 data_cleaning_and_lambda.py 파일에서 찾을 수 있습니다.

AWS에서 실행하는 중에 Python 또는 PySpark 스크립트를 디버깅하는 기본적인 방법은 AWS Glue Studio에서 노트북을 사용하는 것입니다.

1단계: Amazon S3 버킷에서 데이터 크롤

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 크롤러 구성의 절차를 밟고 `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv` 파일을 크롤할 수 있는 새로운 크롤러를 생성하며 그 결과 생성된 메타데이터를 AWS Glue Data Catalog의 `payments`라는 데이터베이스에 둡니다.
3. 새로운 크롤러를 실행한 다음 `payments` 데이터베이스를 확인합니다. 파일의 시작 부분을 읽어 형식과 구분 기호를 확인한 후 크롤러가 데이터베이스에 `medicare`라는 이름의 메타데이터 테이블을 생성했을 것입니다.

   새로운 `medicare`의 스키마는 다음과 같습니다.

```
Column  name                                Data type
==================================================
drg definition                              string
provider id                                 bigint
provider name                               string
provider street address                     string
provider city                               string
provider state                              string
provider zip code                           bigint
hospital referral region description        string
total discharges                            bigint
average covered charges                     string
average total payments                      string
average medicare payments                   string
```

2단계: 개발 엔드포인트 노트북에 표준 문안 스크립트 추가

다음 표준 문안 스크립트를 개발 엔드포인트 노트북에 복사하고 필요한 AWS Glue 라이브러리로 들여와 단일 `GlueContext`를 설정합니다.

```
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
```

```
glueContext = GlueContext(SparkContext.getOrCreate())
```

3단계: 다른 스키마 파싱과 비교

그 다음, Apache Spark DataFrame이 인지한 스키마가 AWS Glue 크롤러가 기록한 것과 동일한지 알아봅니다. 이 코드를 실행합니다.

```
medicare = spark.read.format(
   "com.databricks.spark.csv").option(
   "header", "true").option(
   "inferSchema", "true").load(
   's3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv')
medicare.printSchema()
```

printSchema 호출에 따른 출력값입니다.

```
root
 |-- DRG Definition: string (nullable = true)
 |-- Provider Id: string (nullable = true)
 |-- Provider Name: string (nullable = true)
 |-- Provider Street Address: string (nullable = true)
 |-- Provider City: string (nullable = true)
 |-- Provider State: string (nullable = true)
 |-- Provider Zip Code: integer (nullable = true)
 |-- Hospital Referral Region Description: string (nullable = true)
 |--  Total Discharges : integer (nullable = true)
 |--  Average Covered Charges : string (nullable = true)
 |--  Average Total Payments : string (nullable = true)
 |-- Average Medicare Payments: string (nullable = true)
```

다음, AWS Glue DynamicFrame가 생성한 스키마를 알아봅니다.

```
medicare_dynamicframe = glueContext.create_dynamic_frame.from_catalog(
       database = "payments",
       table_name = "medicare")
medicare_dynamicframe.printSchema()
```

printSchema의 출력값은 다음과 같습니다.

```
root
 |-- drg definition: string
 |-- provider id: choice
 |    |-- long
 |    |-- string
 |-- provider name: string
 |-- provider street address: string
 |-- provider city: string
 |-- provider state: string
 |-- provider zip code: long
 |-- hospital referral region description: string
 |-- total discharges: long
 |-- average covered charges: string
 |-- average total payments: string
 |-- average medicare payments: string
```

DynamicFrame는 provider id가 long 혹은 string 유형일 수 있는 스키마를 생성합니다. DataFrame 스키마는 Provider Id를 string 유형으로 목록에 기록하고 Data Catalog는 provider id를 bigint 유형으로 목록에 기록합니다.

무엇이 정답입니까? 파일의 끝에는 이 열에 있는 string 값과 함께 (160,000 기록 중) 두 기록이 있습니다. 이 두 기록은 문제를 보여주기 위한 잘못된 기록입니다.

이런 문제를 설명하기 위해서 AWS Glue DynamicFrame는 선택 유형의 개념을 도입합니다. 이 경우, DynamicFrame는 이 열에 나타나는 long 및 string 모두를 보여줍니다. AWS Glue 크롤러는 string 값을 누락했는데 그 이유는 데이터 2MB 접두사만 고려했기 때문입니다. Apache Spark DataFrame는 전체 데이터 세트를 고려했지만 string이라는 열에 가장 일반적인 유형을 지정하도록 강요되었습니다. 사실, Spark는 익숙하지 않은 복잡한 유형이나 변수가 있으면 가장 일반적인 케이스를 적용합니다.

provider id 열을 쿼리하려면 먼저 선택 유형을 선택합니다. DynamicFrame의 resolveChoice 변환 방법을 사용하여 cast:long 옵션으로 string 값을 long 값으로 변환할 수 있습니다.

```
medicare_res = medicare_dynamicframe.resolveChoice(specs = [('provider
 id','cast:long')])
medicare_res.printSchema()
```

이제 printSchema 출력값은 다음과 같습니다.

```
root
 |-- drg definition: string
 |-- provider id: long
 |-- provider name: string
 |-- provider street address: string
 |-- provider city: string
 |-- provider state: string
 |-- provider zip code: long
 |-- hospital referral region description: string
 |-- total discharges: long
 |-- average covered charges: string
 |-- average total payments: string
 |-- average medicare payments: string
```

값이 보낼 수 없는 `string`이면 AWS Glue는 `null`을 삽입합니다.

다른 방법으로 선택 유형을 `struct`으로 변환하는 것인데 두 유형의 값은 유지됩니다.

다음, 이례적인 열을 알아봅니다.

```
medicare_res.toDF().where("'provider id' is NULL").show()
```

다음을 알아봅니다.

```
+--------------------+-----------+--------------+----------------------+-------------
+--------------+-----------------+-------------------------------------+---------------
+----------------------+----------------------+------------------------+
|      drg definition|provider id|  provider name|provider street address|provider
 city|provider state|provider zip code|hospital referral region description|total
 discharges|average covered charges|average total payments|average medicare payments|
+--------------------+-----------+--------------+----------------------+-------------
+--------------+-----------------+-------------------------------------+---------------
+----------------------+----------------------+------------------------+
|948 - SIGNS & SYM...|       null|            INC|       1050 DIVISION ST|
 MAUSTON|            WI|            53948|                        WI - Madison|
      12|              $11961.41|              $4619.00|                 $3775.33|
|948 - SIGNS & SYM...|       null| INC- ST JOSEPH|     5000 W CHAMBERS ST|
 MILWAUKEE|            WI|            53210|                        WI - Milwaukee|
        14|              $10514.28|              $5562.50|                 $4522.78|
```

```
+--------------------+-----------+--------------+----------------------+-------------
+--------------+-----------------+-------------------------------------+---------------
+----------------------+----------------------+------------------------+
```

이제, 오류 기록을 다음과 같이 제거합니다.

```
medicare_dataframe = medicare_res.toDF()
medicare_dataframe = medicare_dataframe.where("'provider id' is NOT NULL")
```

4단계: 데이터 매핑 및 Apache Spark Lambda 함수 사용

AWS Glue는 사용자 정의 함수인 Lambda 함수를 직접 지원하지 않습니다. 하지만 항상 DynamicFrame와 Apache Spark DataFrame 간에 변환하여 DynamicFrames의 특별한 기능 외에도 Spark 기능도 활용할 수 있습니다.

그런 다음 결제 정보를 숫자로 변환하여 Amazon Redshift 또는 Amazon Athena와 같은 분석 엔진이 빠르게 수 처리를 할 수 있게 만듭니다.

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

chop_f = udf(lambda x: x[1:], StringType())
medicare_dataframe = medicare_dataframe.withColumn(
        "ACC", chop_f(
            medicare_dataframe["average covered charges"])).withColumn(
                "ATP", chop_f(
                    medicare_dataframe["average total payments"])).withColumn(
                        "AMP", chop_f(
                            medicare_dataframe["average medicare payments"]))
medicare_dataframe.select(['ACC', 'ATP', 'AMP']).show()
```

show 호출의 출력값은 다음과 같습니다.

```
+--------+-------+-------+
|     ACC|    ATP|    AMP|
+--------+-------+-------+
|32963.07|5777.24|4763.73|
|15131.85|5787.57|4976.71|
|37560.37|5434.95|4453.79|
|13998.28|5417.56|4129.16|
|31633.27|5658.33|4851.44|
```

```
|16920.79|6653.80|5374.14|
|11977.13|5834.74|4761.41|
|35841.09|8031.12|5858.50|
|28523.39|6113.38|5228.40|
|75233.38|5541.05|4386.94|
|67327.92|5461.57|4493.57|
|39607.28|5356.28|4408.20|
|22862.23|5374.65|4186.02|
|31110.85|5366.23|4376.23|
|25411.33|5282.93|4383.73|
| 9234.51|5676.55|4509.11|
|15895.85|5930.11|3972.85|
|19721.16|6192.54|5179.38|
|10710.88|4968.00|3898.88|
|51343.75|5996.00|4962.45|
+--------+-------+-------+
only showing top 20 rows
```

이것은 아직도 모두 데이터에서 문자열입니다. 강력한 `apply_mapping` 변환 방법을 사용하여 데이터를 드롭, 이름 바꾸기, 중첩할 수 있어 다른 데이터 프로그램 언어 및 시스템이 쉽게 접근할 수 있도록 합니다.

```
from awsglue.dynamicframe import DynamicFrame
medicare_tmp_dyf = DynamicFrame.fromDF(medicare_dataframe, glueContext, "nested")
medicare_nest_dyf = medicare_tmp_dyf.apply_mapping([('drg definition', 'string', 'drg',
 'string'),
                ('provider id', 'long', 'provider.id', 'long'),
                ('provider name', 'string', 'provider.name', 'string'),
                ('provider city', 'string', 'provider.city', 'string'),
                ('provider state', 'string', 'provider.state', 'string'),
                ('provider zip code', 'long', 'provider.zip', 'long'),
                ('hospital referral region description', 'string','rr', 'string'),
                ('ACC', 'string', 'charges.covered', 'double'),
                ('ATP', 'string', 'charges.total_pay', 'double'),
                ('AMP', 'string', 'charges.medicare_pay', 'double')])
medicare_nest_dyf.printSchema()
```

`printSchema` 출력값은 다음과 같습니다.

```
root
 |-- drg: string
 |-- provider: struct
```

```
 |    |-- id: long
 |    |-- name: string
 |    |-- city: string
 |    |-- state: string
 |    |-- zip: long
 |-- rr: string
 |-- charges: struct
 |    |-- covered: double
 |    |-- total_pay: double
 |    |-- medicare_pay: double
```

데이터를 Spark DataFrame으로 되돌리면 현재는 어떻게 생겼는지 볼 수 있습니다.

```
medicare_nest_dyf.toDF().show()
```

출력값은 다음과 같습니다.

```
+--------------------+--------------------+---------------+--------------------+
|                 drg|            provider|             rr|             charges|
+--------------------+--------------------+---------------+--------------------+
|039 - EXTRACRANIA...|[10001,SOUTHEAST ...|    AL - Dothan|[32963.07,5777.24...|
|039 - EXTRACRANIA...|[10005,MARSHALL M...|AL - Birmingham|[15131.85,5787.57...|
|039 - EXTRACRANIA...|[10006,ELIZA COFF...|AL - Birmingham|[37560.37,5434.95...|
|039 - EXTRACRANIA...|[10011,ST VINCENT...|AL - Birmingham|[13998.28,5417.56...|
|039 - EXTRACRANIA...|[10016,SHELBY BAP...|AL - Birmingham|[31633.27,5658.33...|
|039 - EXTRACRANIA...|[10023,BAPTIST ME...|AL - Montgomery|[16920.79,6653.8,...|
|039 - EXTRACRANIA...|[10029,EAST ALABA...|AL - Birmingham|[11977.13,5834.74...|
|039 - EXTRACRANIA...|[10033,UNIVERSITY...|AL - Birmingham|[35841.09,8031.12...|
|039 - EXTRACRANIA...|[10039,HUNTSVILLE...|AL - Huntsville|[28523.39,6113.38...|
|039 - EXTRACRANIA...|[10040,GADSDEN RE...|AL - Birmingham|[75233.38,5541.05...|
|039 - EXTRACRANIA...|[10046,RIVERVIEW ...|AL - Birmingham|[67327.92,5461.57...|
|039 - EXTRACRANIA...|[10055,FLOWERS HO...|    AL - Dothan|[39607.28,5356.28...|
|039 - EXTRACRANIA...|[10056,ST VINCENT...|AL - Birmingham|[22862.23,5374.65...|
|039 - EXTRACRANIA...|[10078,NORTHEAST ...|AL - Birmingham|[31110.85,5366.23...|
|039 - EXTRACRANIA...|[10083,SOUTH BALD...|    AL - Mobile|[25411.33,5282.93...|
|039 - EXTRACRANIA...|[10085,DECATUR GE...|AL - Huntsville|[9234.51,5676.55,...|
|039 - EXTRACRANIA...|[10090,PROVIDENCE...|    AL - Mobile|[15895.85,5930.11...|
|039 - EXTRACRANIA...|[10092,D C H REGI...|AL - Tuscaloosa|[19721.16,6192.54...|
|039 - EXTRACRANIA...|[10100,THOMAS HOS...|    AL - Mobile|[10710.88,4968.0,...|
|039 - EXTRACRANIA...|[10103,BAPTIST ME...|AL - Birmingham|[51343.75,5996.0,...|
+--------------------+--------------------+---------------+--------------------+
only showing top 20 rows
```

5단계: 데이터를 Apache Parquet에 쓰기

AWS Glue는 Apache Parquet과 같은 형식으로 데이터를 쉽게 작성할 수 있어 관계형 데이터베이스가 효과적으로 소비될 수 있습니다.

```
glueContext.write_dynamic_frame.from_options(
       frame = medicare_nest_dyf,
       connection_type = "s3",
       connection_options = {"path": "s3://glue-sample-target/output-dir/
medicare_parquet"},
       format = "parquet")
```

AWS Glue PySpark 확장 참조

AWS Glue는 다음 확장 기능을 PySpark Python 언어에 생성합니다.

- getResolvedOptions를 사용한 파라미터 액세스
- PySpark 확장 유형
- DynamicFrame 클래스
- DynamicFrameCollection 클래스
- DynamicFrameWriter 클래스
- DynamicFrameReader 클래스
- GlueContext 클래스

getResolvedOptions를 사용한 파라미터 액세스

AWS Glue `getResolvedOptions(args, options)` 유틸리티 함수는 작업이 실행되면 스크립트로 통과된 인수로 액세스할 수 있도록 도와줍니다. 이 함수를 사용하려면 먼저 `sys` 모듈과 함께 AWS Glue `utils` 모듈에서 가져옵니다.

```
import sys
from awsglue.utils import getResolvedOptions
```

getResolvedOptions(args, options)

- `args` - `sys.argv`에 포함된 인수 목록입니다.
- `options` - 가져오고자 하는 인수 이름의 Python 배열입니다.

Example JobRun으로 통과된 인수 가져오기

JobRun을 Lambda 함수 내 스크립트에 생성했다고 가정합니다.

```
response = client.start_job_run(
             JobName = 'my_test_Job',
             Arguments = {
               '--day_partition_key':   'partition_0',
               '--hour_partition_key':  'partition_1',
               '--day_partition_value':  day_partition_value,
               '--hour_partition_value': hour_partition_value } )
```

통과된 인수를 가져오기 위해서는 다음과 같이 `getResolvedOptions` 함수를 사용합니다.

```
import sys
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv,
                          ['JOB_NAME',
                           'day_partition_key',
                           'hour_partition_key',
                           'day_partition_value',
                           'hour_partition_value'])
print "The day-partition key is: ", args['day_partition_key']
print "and the day-partition value is: ", args['day_partition_value']
```

각 인수는 두 개의 하이픈으로 시작된 다음 하이픈 없이 스크립트에서 참조되는 것으로 정의됩니다. 인수는 하이픈이 아닌 밑줄만 사용합니다. 인수가 해석되려면 이 규칙을 따라야 합니다.

PySpark 확장 유형

AWS Glue PySpark 확장에 사용된 유형.

데이터 유형

다른 AWS Glue 유형의 베이스 클래스.

`__init__(properties={})`

- `properties` - 데이터 유형의 속성(선택 사항)입니다.

typeName(cls)

AWS Glue 유형 클래스(즉, 끝에서 "Type"이 제거된 클래스)의 유형을 반환합니다.

- cls – DataType에서 파생된 AWS Glue 클래스 인스턴스입니다.

jsonValue( )

클래스 속성의 데이터 유형을 포함하는 JSON 객체를 반환합니다.

```
{
  "dataType": typeName,
  "properties": properties
}
```

AtomicType 및 단순 파생물

데이터 유형 클래스에서 상속되고 확장되며 모든 AWS Glue 원자 데이터 유형의 베이스 클래스로서 역할을 합니다.

fromJsonValue(cls, json_value)

JSON 객체의 값으로 클래스 인스턴스를 초기화합니다.

- cls – 초기화할 AWS Glue 유형 클래스 인스턴스입니다.
- json_value – JSON 객체에서 키-값 페어를 로드합니다.

다음 유형은 AtomicType 클래스의 단순 파생물입니다.

- BinaryType – 이진 데이터입니다.
- BooleanType – 부울 값입니다.
- ByteType - 바이트 값입니다.
- DateType - 날짜/시간 값입니다.
- DoubleType – 부동 소수점 값입니다.
- IntegerType – 정수 값입니다.

- `LongType` - 정수(long) 값입니다.
- `NullType` – null 값입니다.
- `ShortType` - 정수(short) 값입니다.
- `StringType` - 텍스트 문자열입니다.
- `TimestampType` – 타임스탬프 값(일반적으로 1/1/1970에서 초 단위)입니다.
- `UnknownType` - 식별되지 않은 유형의 값입니다.

DecimalType(AtomicType)

AtomicType 클래스에서 상속받고 확장하여 십진수(이진수 기반 2 숫자와 반대 개념인 십진수로 표현된 수)를 나타냅니다.

`__init__(precision=10, scale=2, properties={})`

- `precision` – 십진수의 단위 수입니다(선택; 기본값이 10).
- `scale` – 십진수 오른쪽의 단위 수입니다(선택; 기본값이 2).
- `properties` – 십진수 유형의 속성(선택 사항).

EnumType(AtomicType)

AtomicType 클래스에서 상속받고 확장되어 유효한 옵션을 열거합니다.

`__init__(options)`

- `options` – 열거된 옵션 목록입니다.

컬렉션 형식

- ArrayType(DataType)
- ChoiceType(DataType)
- MapType(DataType)
- 필드(객체)
- StructType(DataType)
- EntityType(DataType)

ArrayType(DataType)

`__init__(elementType=UnknownType(), properties={})`

- `elementType` – 배열의 요소 유형(선택; 기본값은 UnknownType입니다).
- `properties` - 배열의 속성입니다(선택 사항).

ChoiceType(DataType)

`__init__(choices=[], properties={})`

- `choices` - 가능한 선택 항목의 목록입니다(선택 사항).
- `properties` - 이러한 선택 항목의 속성입니다(선택 사항).

`add(new_choice)`

가능한 선택 목록에 새로운 선택을 추가합니다.

- `new_choice` – 가능한 선택 항목 목록에 추가할 선택 항목입니다.

`merge(new_choices)`

기존 선택 목록과 새로운 선택 목록을 병합합니다.

- `new_choices` – 기존 선택 항목과 병합할 새 선택 항목 목록입니다.

MapType(DataType)

`__init__(valueType=UnknownType, properties={})`

- `valueType` – 맵 값의 유형(선택; 기본값은 UnknownType입니다).
- `properties` - 맵의 속성입니다(선택 사항).

필드(객체)

데이터 유형에서 상속받은 객체에서 필드 객체를 생성합니다.

__init__(name, dataType, properties={})

- name – 필드에 지정된 이름입니다.
- dataType - 필드를 생성할 객체입니다.
- properties - 필드의 속성입니다(선택 사항).

StructType(DataType)

데이터 구조를 정의합니다 (struct).

__init__(fields=[], properties={})

- fields – 구조에 포함할 Field 유형의 필드는 목록입니다(선택 사항).
- properties - 구조의 속성입니다(선택 사항).

add(field)

- field – 구조에 추가할 Field 유형의 객체입니다.

hasField(field)

이 구조가 동일 이름의 필드를 가지고 있다면 True를 반환하고 그렇지 않으면 False를 반환합니다.

- field - 필드 이름 혹은 이름이 사용된 Field 유형의 객체입니다.

getField(field)

- field - 필드 이름 혹은 이름이 사용된 Field 유형의 객체입니다. 구조가 동일 이름의 필드를 가지고 있다면 이것은 반환됩니다.

EntityType(DataType)

__init__(entity, base_type, properties)

이 클래스는 아직 구현되지 않았습니다.

기타 유형

- DataSource(객체)
- DataSink(객체)

DataSource(객체)

__init__(j_source, sql_ctx, name)

- j_source - 데이터 원본입니다.
- sql_ctx - SQL 컨텍스트입니다.
- name – 데이터 원본 이름입니다.

setFormat(format, **options)

- format – 데이터 원본을 설정하기 위한 포맷입니다.
- options – 데이터 원본을 설정하기 위한 옵션 모음입니다. 형식 옵션에 대한 자세한 내용은 the section called “데이터 포맷 옵션” 섹션을 참조하세요.

getFrame()

데이터 원본을 위한 DynamicFrame을 반환합니다.

DataSink(객체)

__init__(j_sink, sql_ctx)

- j_sink - 생성할 싱크입니다.
- sql_ctx – 데이터 싱크를 위한 SQL 컨텍스트입니다.

setFormat(format, **options)

- format - 데이터 싱크에 대해 설정할 포맷입니다.
- options – 데이터 싱크를 설정하기 위한 옵션 모음입니다. 형식 옵션에 대한 자세한 내용은 the section called “데이터 포맷 옵션” 섹션을 참조하세요.

setAccumulableSize(size)

- size – 바이트 단위로 설정할 늘어날 수 있는 사이즈입니다.

writeFrame(dynamic_frame, info="")

- dynamic_frame - 작성할 DynamicFrame입니다.
- info - DynamicFrame에 대한 정보입니다(선택 사항).

write(dynamic_frame_or_dfc, info="")

DynamicFrame 혹은 DynamicFrameCollection 작성.

- dynamic_frame_or_dfc – 작성할 DynamicFrame 객체 또는 DynamicFrameCollection 객체입니다.
- info- 작성할 DynamicFrame 또는 DynamicFrames에 대한 정보입니다(선택 사항).

DynamicFrame 클래스

Apache Spark의 주요 추상화 중 하나는 SparkSQL DataFrame이며, R 및 Pandas에서 찾아볼 수 있는 DataFrame 구문과 유사합니다. DataFrame은 테이블과 비슷한 기능적 스타일(맵/줄임/필터 등) 작업과 SQL 작업(선택, 계획, 집계)을 지원합니다.

DataFrames 는 광범위하게 사용되는 유용한 것이지만 추출, 변환, 로드(ETL) 작업 시 제한이 있습니다. 무엇보다도 데이터가 로딩되기 전에 스키마를 명시해야 합니다. 첫 번째는 스키마를 추론하고 두 번째는 데이터를 로드하도록 두 개를 데이터에 통과시켜 SparkSQL이 이를 해결합니다. 하지만 이러한 추론은 제한적이며 복잡한 데이터의 현실을 해결해주지 않습니다. 예를 들어, 동일한 필드는 다른 기록에서 다른 유형을 가져야 합니다. 아파치 스파크는 간혹 실행을 포기하고 기존 필드 텍스트를 사용하여 string으로써 유형을 보고합니다. 이는 올바르지 않을 수 있고 스키마 차이를 해결할 수 있는 좀 더 확실한 관리법을 알고 싶을 겁니다. 라지 데이터세트의 경우, 추가로 소스 데이터를 통과하는 것이 매우 비쌀 수 있습니다.

이러한 상황을 해결하려면 AWS Glue는 DynamicFrame을 도입합니다. DynamicFrame는 애초에 스키마가 필요없이 각 기록이 자기 설명적인 것을 제외하고는 DataFrame와 비슷합니다. 대신에 AWS Glue는 필요하면 스키마 온더플라이를 계산하고 선택 (혹은 재결합) 유형을 사용하여 스키마 불일치를

확실하게 암호화합니다. 이런 불일치를 해결하여 데이터세트와 고정 스키마가 필요한 데이터 스토어를 호환하게 만들 수 있습니다.

비슷하게, DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 스파크 DataFrame의 열과 같습니다. PySpark에서 AWS Glue를 사용하는 경우 일반적으로 독립된 DynamicRecords를 조작하지 않습니다. 대신, 해당 DynamicFrame을 통해 데이터 세트를 함께 변환합니다.

모든 스키마 불일치를 해결한 후 DynamicFrames로 그리고 DataFrames에서 변환할 수 있습니다.

- 생성 -

- __init__
- fromDF
- toDF

__init__

__init__(jdf, glue_ctx, name)

- jdf - Java Virtual Machine(JVM)의 데이터 프레임 참조입니다.
- glue_ctx – GlueContext 클래스 객체입니다.
- name - 기본값이 빈 선택적 이름 문자열입니다.

fromDF

fromDF(dataframe, glue_ctx, name)

DataFrame 필드를 DynamicRecord 필드로 변환하여 DataFrame를 DynamicFrame로 변환합니다. 새로운 DynamicFrame을 반환합니다.

DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 스파크 DataFrame의 열과 비슷합니다.

이 함수는 DataFrame에서 이름이 중복된 열이 이미 해결된 것으로 예상합니다.

- dataframe - Apache Spark SQL DataFrame으로 변환합니다(필수).
- glue_ctx - 이 변환의 맥락을 명시하는 GlueContext 클래스 객체입니다(필수).
- name – 결과 DynamicFrame의 이름(AWS Glue 3.0부터 선택 사항).

toDF

toDF(options)

DynamicRecords를 DataFrame 필드로 변환하여 DynamicFrame을 Apache Spark DataFrame으로 변환합니다. 새로운 DataFrame을 반환합니다.

DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 스파크 DataFrame의 열과 비슷합니다.

- options - 옵션의 목록입니다. 변환 프로세스에 대한 추가 옵션을 지정할 수 있습니다. 'options' 파라미터와 함께 사용할 수 있는 몇 가지 유효한 옵션은 다음과 같습니다.
  - format - 데이터 형식을 지정합니다(예: json, csv, parquet).
  - separater or sep - CSV 파일의 경우 구분 기호를 지정합니다.
  - header - CSV 파일의 경우 첫 번째 행이 헤더인지 여부를 나타냅니다(True/False).
  - inferSchema - Spark가 스키마를 자동으로 유추하도록 지시합니다(True/False).

  다음은 'options' 파라미터를 'toDF' 메서드와 함께 사용하는 예제입니다.

```
from awsglue.context import GlueContext
from awsglue.dynamicframe import DynamicFrame
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

csv_dyf = glueContext.create_dynamic_frame.from_options(
  connection_type="s3",
  connection_options={"paths": ["s3://my-bucket/path/to/csv/"]},
  format="csv"
)
csv_cf = csv_dyf.toDF(options={
  "separator": ",",
  "header": "true",
  "inferSchema": "true"
})
```

  Project와 Cast 작업 유형을 선택한 경우 대상 유형을 지정합니다. 예는 다음과 같습니다.

```
>>>toDF([ResolveOption("a.b.c", "KeepAsStruct")])
```

```
>>>toDF([ResolveOption("a.b.c", "Project", DoubleType())])
```

- 정보 -

- count
- schema
- printSchema
- show
- repartition
- coalesce

count

`count( )` - 기본 `DataFrame`의 행 수를 반환합니다.

schema

`schema( )` – 이 `DynamicFrame`의 스키마를 반환하거나, 가능하지 않는 경우 기본 `DataFrame`의 스키마를 반환합니다.

이 스키마를 구성하는 `DynamicFrame` 유형에 대한 자세한 내용은 the section called “타입” 섹션을 참조하세요.

printSchema

`printSchema( )` - 기본 `DataFrame`의 스키마를 인쇄합니다.

show

`show(num_rows)` - 기본 `DataFrame`으로부터 지정된 수의 행을 인쇄합니다.

repartition

`repartition(numPartitions)` – `numPartitions` 파티션이 있는 새 `DynamicFrame`을 반환합니다.

coalesce

`coalesce(numPartitions)` – `numPartitions` 파티션이 있는 새 `DynamicFrame`을 반환합니다.

- 변형 -

- apply_mapping
- drop_fields
- 필터
- join
- map
- mergeDynamicFrame
- relationalize
- rename_field
- resolveChoice
- select_fields
- spigot
- split_fields
- split_rows
- unbox
- the section called “결합”
- unnest
- unnest_ddb_json
- write

apply_mapping

`apply_mapping(mappings, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

`DynamicFrame`으로 매핑을 선언하고 지정한 필드에 해당 매핑이 적용된 새로운 `DynamicFrame`을 반환합니다. 지정되지 않은 필드는 새 `DynamicFrame` 필드에서 생략됩니다.

- `mappings` - 매핑 튜플 목록(필수). 각각 (소스 열, 소스 유형, 대상 열, 대상 유형)으로 구성된 매핑 튜플 목록입니다.

  소스 열 이름에 점(`'.'`)이 있는 경우 백틱(`` ``)으로 묶어야 합니다. 예를 들어 `this.old.name`(문자열)을 `thisNewName`에 매핑하려면 다음 튜플을 사용합니다.

```
("`this.old.name`", "string", "thisNewName", "string")
```

- `transformation_ctx` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- `info` - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- `stageThreshold` - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- `totalThreshold` - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예: apply_maping을 사용하여 필드 이름을 바꾸고 필드 유형을 변경합니다.

다음 코드 예제에서는 `apply_mapping` 메서드를 사용하여 선택한 필드의 이름을 바꾸고 필드 유형을 변경하는 방법을 보여 줍니다.

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: 데이터 조인 및 관계화 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

```
# Example: Use apply_mapping to reshape source data into
# the desired column names and types as a new DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Select and rename fields, change field type
print("Schema for the persons_mapped DynamicFrame, created with apply_mapping:")
```

```
persons_mapped = persons.apply_mapping(
    [
        ("family_name", "String", "last_name", "String"),
        ("name", "String", "first_name", "String"),
        ("birth_date", "String", "date_of_birth", "Date"),
    ]
)
persons_mapped.printSchema()
```

출력

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

```
Schema for the persons_mapped DynamicFrame, created with apply_mapping:
root
|-- last_name: string
|-- first_name: string
|-- date_of_birth: date
```

drop_fields

drop_fields(paths, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

FlatMap 클래스 변환을 불러들여 DynamicFrame에서 필드를 제거합니다. 특정 필드가 드롭된 새로운 DynamicFrame을 반환합니다.

- paths – 문자열 목록입니다. 각각에는 드롭할 필드 노드에 대한 전체 경로가 포함되어 있습니다. 점 표기를 사용하여 중첩된 필드를 지정할 수 있습니다. 예를 들어 first 필드가 트리에서 name 필드의 자식 필드인 경우 해당 경로에 "name.first"를 지정합니다.

  필드 노드 이름에 . 리터럴이 있는 경우 이름을 백틱(`)으로 묶어야 합니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예: drop_fields를 사용하여 **DynamicFrame**에서 필드를 제거합니다.

이 코드 예제에서는 drop_fields 메서드를 사용하여 DynamicFrame에서 선택한 최상위 및 중첩 필드를 제거합니다.

예제 데이터 세트

이 예제에서는 코드의 EXAMPLE-FRIENDS-DATA 테이블로 표시되는 다음 데이터 세트를 사용합니다.

```
{"name": "Sally", "age": 23, "location": {"state": "WY", "county": "Fremont"},
 "friends": []}
```

```
{"name": "Varun", "age": 34, "location": {"state": "NE", "county": "Douglas"},
 "friends": [{"name": "Arjun", "age": 3}]}
{"name": "George", "age": 52, "location": {"state": "NY"}, "friends": [{"name":
 "Fred"}, {"name": "Amy", "age": 15}]}
{"name": "Haruki", "age": 21, "location": {"state": "AK", "county": "Denali"}}
{"name": "Sheila", "age": 63, "friends": [{"name": "Nancy", "age": 22}]}
```

예제 코드

```
# Example: Use drop_fields to remove top-level and nested fields from a DynamicFrame.
# Replace MY-EXAMPLE-DATABASE with your Glue Data Catalog database name.
# Replace EXAMPLE-FRIENDS-DATA with your table name.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame from Glue Data Catalog
glue_source_database = "MY-EXAMPLE-DATABASE"
glue_source_table = "EXAMPLE-FRIENDS-DATA"

friends = glueContext.create_dynamic_frame.from_catalog(
    database=glue_source_database, table_name=glue_source_table
)
print("Schema for friends DynamicFrame before calling drop_fields:")
friends.printSchema()

# Remove location.county, remove friends.age, remove age
friends = friends.drop_fields(paths=["age", "location.county", "friends.age"])
print("Schema for friends DynamicFrame after removing age, county, and friend age:")
friends.printSchema()
```

출력

```
Schema for friends DynamicFrame before calling drop_fields:
root
|-- name: string
|-- age: int
|-- location: struct
|    |-- state: string
```

```
|    |-- county: string
|-- friends: array
|    |-- element: struct
|    |    |-- name: string
|    |    |-- age: int

Schema for friends DynamicFrame after removing age, county, and friend age:
root
|-- name: string
|-- location: struct
|    |-- state: string
|-- friends: array
|    |-- element: struct
|    |    |-- name: string
```

필터

filter(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

지정된 술어 함수 f를 만족하는 DynamicFrame이 입력된 DynamicRecords에 모두 포함된 새로운 DynamicFrame을 반환합니다.

- f - DynamicFrame에 적용하는 조건자 함수입니다. 함수는 DynamicRecord를 논리로 받아들이며, DynamicRecord가 필터 요구 사항과 맞으면 True을 반환하고 아니면 False를 반환합니다(필수).

  DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 스파크 DataFrame의 열과 비슷합니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예: 필터를 사용하여 필터링된 필드 선택 가져오기

이 예에서는 filter 메서드를 사용하여 다른 사람의 DynamicFrame 필드에 대한 필터링된 선택을 포함하는 새 DynamicFrame 항목을 만듭니다.

map 메서드와 같이 filter는 함수를 인수로 취하여 원본 DynamicFrame의 각 레코드에 적용됩니다. 이 함수는 레코드를 입력으로 받아 부울 값을 반환합니다. 반환 값이 true이면 레코드가 결과 DynamicFrame에 포함됩니다. 거짓이면 레코드가 제외됩니다.

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: ResolveChoice, Lambda, 및 ApplyMapping을 사용한 데이터 준비 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

```
# Example: Use filter to create a new DynamicFrame
# with a filtered selection of records

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create DynamicFrame from Glue Data Catalog
medicare = glueContext.create_dynamic_frame.from_options(
    "s3",
    {
        "paths": [
            "s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv"
        ]
    },
    "csv",
    {"withHeader": True},
)

# Create filtered DynamicFrame with custom lambda
# to filter records by Provider State and Provider City
sac_or_mon = medicare.filter(
    f=lambda x: x["Provider State"] in ["CA", "AL"]
    and x["Provider City"] in ["SACRAMENTO", "MONTGOMERY"]
)

# Compare record counts
print("Unfiltered record count: ", medicare.count())
```

```
print("Filtered record count:  ", sac_or_mon.count())
```

출력

```
Unfiltered record count:  163065
Filtered record count:   564
```

join

join(paths1, paths2, frame2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

다른 DynamicFrame로 균등 연결을 실행하고 결과인 DynamicFrame을 반환합니다.

- paths1 - 조인할 이 프레임의 키 목록입니다.
- paths2 - 조인할 다른 프레임의 키 목록입니다.
- frame2 - 조인할 다른 DynamicFrame입니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예: 조인을 사용하여 **DynamicFrames** 결합

이 예제에서는 join 메서드를 사용하여 3개의 DynamicFrames에서 조인을 수행합니다. AWS Glue는 사용자가 제공한 필드 키를 기반으로 조인을 수행합니다. 결과 DynamicFrame에는 지정된 키가 일치하는 두 원본 프레임의 행이 포함됩니다.

join 변환은 모든 필드를 그대로 유지한다는 점에 유의하세요. 즉, 일치하도록 지정하는 필드가 중복되고 동일한 키가 포함된 경우에도 결과 DynamicFrame에 나타납니다. 이 예에서는 drop_fields를 사용하여 조인 후 이러한 중복 키를 제거합니다.

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: 데이터 조인 및 관계화 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

```
# Example: Use join to combine data from three DynamicFrames

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load DynamicFrames from Glue Data Catalog
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
memberships = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="memberships_json"
)
orgs = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="organizations_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()
print("Schema for the memberships DynamicFrame:")
memberships.printSchema()
print("Schema for the orgs DynamicFrame:")
orgs.printSchema()

# Join persons and memberships by ID
persons_memberships = persons.join(
    paths1=["id"], paths2=["person_id"], frame2=memberships
)

# Rename and drop fields from orgs
# to prevent field name collisions with persons_memberships
orgs = (
    orgs.drop_fields(["other_names", "identifiers"])
    .rename_field("id", "org_id")
```

```
    .rename_field("name", "org_name")
)

# Create final join of all three DynamicFrames
legislators_combined = orgs.join(
    paths1=["org_id"], paths2=["organization_id"], frame2=persons_memberships
).drop_fields(["person_id", "org_id"])

# Inspect the schema for the joined data
print("Schema for the new legislators_combined DynamicFrame:")
legislators_combined.printSchema()
```

출력

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
```

```
|    |    |-- value: string
|-- death_date: string

Schema for the memberships DynamicFrame:
root
|-- area_id: string
|-- on_behalf_of_id: string
|-- organization_id: string
|-- role: string
|-- person_id: string
|-- legislative_period_id: string
|-- start_date: string
|-- end_date: string

Schema for the orgs DynamicFrame:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- classification: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string

Schema for the new legislators_combined DynamicFrame:
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
```

```
|    |    |-- url: string
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- start_date: string
|-- family_name: string
|-- id: string
|-- death_date: string
|-- end_date: string
```

map

map(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

특정 매핑 함수를 기존 DynamicFrame의 모든 기록에 적용한 결과인 새로운 DynamicFrame을 반환합니다.

- f - DynamicFrame의 모든 레코드에 적용하는 매핑 함수입니다. 함수는 DynamicRecord를 인수로 받아들이고 새로운 DynamicRecord를 반환합니다(필수).

  DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 Apache Spark DataFrame의 열과 비슷합니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

예: map을 사용하여 **DynamicFrame**의 모든 레코드에 함수를 적용합니다.

이 예제에서는 map 메서드를 사용하여 DynamicFrame의 모든 레코드에 기능을 적용하는 방법을 보여 줍니다. 특히 이 예에서는 여러 주소 필드를 단일 struct 유형으로 병합하기 위해 각 레코드에 MergeAddress라는 함수를 적용합니다.

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: ResolveChoice, Lambda, 및 ApplyMapping을 사용한 데이터 준비 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

```
# Example: Use map to combine fields in all records
# of a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
medicare = glueContext.create_dynamic_frame.from_options(
        "s3",
```

```
        {"paths": ["s3://awsglue-datasets/examples/medicare/
Medicare_Hospital_Provider.csv"]},
        "csv",
        {"withHeader": True})
print("Schema for medicare DynamicFrame:")
medicare.printSchema()

# Define a function to supply to the map transform
# that merges address fields into a single field
def MergeAddress(rec):
  rec["Address"] = {}
  rec["Address"]["Street"] = rec["Provider Street Address"]
  rec["Address"]["City"] = rec["Provider City"]
  rec["Address"]["State"] = rec["Provider State"]
  rec["Address"]["Zip.Code"] = rec["Provider Zip Code"]
  rec["Address"]["Array"] = [rec["Provider Street Address"], rec["Provider City"],
 rec["Provider State"], rec["Provider Zip Code"]]
  del rec["Provider Street Address"]
  del rec["Provider City"]
  del rec["Provider State"]
  del rec["Provider Zip Code"]
  return rec


# Use map to apply MergeAddress to every record
mapped_medicare = medicare.map(f = MergeAddress)
print("Schema for mapped_medicare DynamicFrame:")
mapped_medicare.printSchema()
```

출력

```
Schema for medicare DynamicFrame:
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

Schema for mapped_medicare DynamicFrame:
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

mergeDynamicFrame

mergeDynamicFrame(stage_dynamic_frame, primary_keys, transformation_ctx = "", options = {}, info = "", stageThreshold = 0, totalThreshold = 0)

레코드를 식별하기 위해, 지정된 기본 키를 기반으로 이 DynamicFrame을 스테이징 DynamicFrame과 병합합니다. 중복 레코드(기본 키가 동일한 레코드)는 중복 제거되지 않습니다. 스테이징 프레임에 일치하는 레코드가 없으면 모든 레코드(중복 레코드 포함)가 소스에서 유지됩니다. 스테이징 프레임에 일치하는 레코드가 있으면 스테이징 프레임의 레코드가 AWS Glue의 소스에 있는 레코드를 덮어씁니다.

- stage_dynamic_frame - 병합할 스테이징 DynamicFrame입니다.
- primary_keys - 소스 및 스테이징 동적 프레임의 레코드와 일치시킬 기본 키 필드 목록입니다.
- transformation_ctx - 현재 변환에 대한 메타데이터를 검색하는 데 사용되는 고유한 문자열입니다(선택 사항).
- options - 이 변환에 대한 추가 정보를 제공하는 JSON 이름-값 페어 문자열입니다. 이 인수는 현재 사용되지 않습니다.
- info - String입니다. 이 변환에 따른 오류와 관련된 문자열입니다.
- stageThreshold - Long입니다. 오류가 발생되는 지정된 변환의 오류 수입니다.

- totalThreshold - Long입니다. 오류가 발생되는 이 변환의 총 오류 수입니다.

이 메소드는 이 DynamicFrame을 스테이징 DynamicFrame과 병합하여 얻은 새 DynamicFrame을 반환합니다.

다음과 같은 경우 반환된 DynamicFrame에 레코드 A가 포함되어 있습니다.

- A가 소스 프레임과 스테이징 프레임 모두에 있는 경우에는 스테이징 프레임의 A가 반환됩니다.
- A가 소스 테이블에 있고 A.primaryKeys가 stagingDynamicFrame에 없는 경우에는 A는 스테이징 테이블에서 업데이트되지 않습니다.

소스 프레임과 스테이징 프레임이 동일한 스키마를 가질 필요는 없습니다.

예: MergeDynamicFrame을 사용하여 기본 키를 기반으로 두 개 **DynamicFrames**를 병합합니다

다음 코드 예제는 기본 키 id를 기반으로 mergeDynamicFrame 메서드를 사용하여 DynamicFrame를 “스테이징” DynamicFrame과 병합하는 방법을 보여줍니다.

예제 데이터 세트

이 예제에서는 split_rows_collection로 호출된 DynamicFrameCollection에서 두 개의 DynamicFrames을 사용합니다. 다음은 split_rows_collection에서의 키 목록입니다.

```
dict_keys(['high', 'low'])
```

예제 코드

```
# Example: Use mergeDynamicFrame to merge DynamicFrames
# based on a set of specified primary keys

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import SelectFromCollection

# Inspect the original DynamicFrames
frame_low = SelectFromCollection.apply(dfc=split_rows_collection, key="low")
print("Inspect the DynamicFrame that contains rows where ID < 10")
frame_low.toDF().show()

frame_high = SelectFromCollection.apply(dfc=split_rows_collection, key="high")
```

```
print("Inspect the DynamicFrame that contains rows where ID > 10")
frame_high.toDF().show()

# Merge the DynamicFrames based on the "id" primary key
merged_high_low = frame_high.mergeDynamicFrame(
    stage_dynamic_frame=frame_low, primary_keys=["id"]
)

# View the results where the ID is 1 or 20
print("Inspect the merged DynamicFrame that contains the combined rows")
merged_high_low.toDF().where("id = 1 or id= 20").orderBy("id").show()
```

출력

```
Inspect the DynamicFrame that contains rows where ID < 10
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                    fax|           202-225-3307|
|  1|    1|                  phone|           202-225-5731|
|  2|    0|                    fax|           202-225-3307|
|  2|    1|                  phone|           202-225-5731|
|  3|    0|                    fax|           202-225-3307|
|  3|    1|                  phone|           202-225-5731|
|  4|    0|                    fax|           202-225-3307|
|  4|    1|                  phone|           202-225-5731|
|  5|    0|                    fax|           202-225-3307|
|  5|    1|                  phone|           202-225-5731|
|  6|    0|                    fax|           202-225-3307|
|  6|    1|                  phone|           202-225-5731|
|  7|    0|                    fax|           202-225-3307|
|  7|    1|                  phone|           202-225-5731|
|  8|    0|                    fax|           202-225-3307|
|  8|    1|                  phone|           202-225-5731|
|  9|    0|                    fax|           202-225-3307|
|  9|    1|                  phone|           202-225-5731|
| 10|    0|                    fax|           202-225-6328|
| 10|    1|                  phone|           202-225-4576|
+---+-----+-----------------------+-----------------------+
only showing top 20 rows

Inspect the DynamicFrame that contains rows where ID > 10
+---+-----+-----------------------+-----------------------+
```

```
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+------------------------+------------------------+
| 11|    0|                     fax|              202-225-6328|
| 11|    1|                   phone|              202-225-4576|
| 11|    2|                 twitter|           RepTrentFranks|
| 12|    0|                     fax|              202-225-6328|
| 12|    1|                   phone|              202-225-4576|
| 12|    2|                 twitter|           RepTrentFranks|
| 13|    0|                     fax|              202-225-6328|
| 13|    1|                   phone|              202-225-4576|
| 13|    2|                 twitter|           RepTrentFranks|
| 14|    0|                     fax|              202-225-6328|
| 14|    1|                   phone|              202-225-4576|
| 14|    2|                 twitter|           RepTrentFranks|
| 15|    0|                     fax|              202-225-6328|
| 15|    1|                   phone|              202-225-4576|
| 15|    2|                 twitter|           RepTrentFranks|
| 16|    0|                     fax|              202-225-6328|
| 16|    1|                   phone|              202-225-4576|
| 16|    2|                 twitter|           RepTrentFranks|
| 17|    0|                     fax|              202-225-6328|
| 17|    1|                   phone|              202-225-4576|
+---+-----+------------------------+------------------------+
only showing top 20 rows

Inspect the merged DynamicFrame that contains the combined rows
+---+-----+------------------------+------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+------------------------+------------------------+
|  1|    0|                     fax|              202-225-3307|
|  1|    1|                   phone|              202-225-5731|
| 20|    0|                     fax|              202-225-5604|
| 20|    1|                   phone|              202-225-6536|
| 20|    2|                 twitter|                 USRepLong|
+---+-----+------------------------+------------------------+
```

relationalize

relationalize(root_table_name, staging_path, options, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

DynamicFrame을 관계형 데이터베이스에 맞는 형식으로 변환합니다. DynamicFrame을 관계형으로 만들면 DynamoDB와 같은 NoSQL 환경에서 MySQL과 같은 관계형 데이터베이스로 데이터를 이동하려는 경우에 특히 유용합니다.

변환은 중첩된 열의 중첩을 해제하고 배열 열을 회전하여 프레임 목록을 생성합니다. 회전 배열 열은 중첩되지 않는 상태에서 생성된 조인 키를 사용하여 루트 테이블에 연결될 수 있습니다.

- root_table_name - 루트 테이블의 스키마 이름입니다.
- staging_path - 메소드가 CSV 포맷으로 회전 테이블의 파티션을 저장하는 경로입니다(선택 사항). 회전 테이블은 이 경로부터 다시 읽습니다.
- options - 선택적 파라미터의 딕셔너리입니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예: 관계형 만들기를 사용하여 **DynamicFrame** 내 중첩된 스키마를 플랫하게 만들기

이 코드 예제는 relationalize 메서드를 사용하여 중첩된 스키마를 관계형 데이터베이스에 맞는 형식으로 평면화합니다.

예제 데이터 세트

이 예제에서는 다음 스키마와 함께 legislators_combined를 호출한 DynamicFrame을 사용합니다. legislators_combined은 links, images, 및 contact_details와 같은 여러 개의 중첩 필드를 가지고, relationalize 변환에 의해 평면화됩니다.

```
root
|-- role: string
|-- seats: int
|-- org_name: string
```

```
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- start_date: string
|-- family_name: string
|-- id: string
|-- death_date: string
|-- end_date: string
```

예제 코드

```
# Example: Use relationalize to flatten
# a nested schema into a format that fits
# into a relational database.
```

```
# Replace DOC-EXAMPLE-S3-BUCKET/tmpDir with your own location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Apply relationalize and inspect new tables
legislators_relationalized = legislators_combined.relationalize(
    "l_root", "s3://DOC-EXAMPLE-BUCKET/tmpDir"
)
legislators_relationalized.keys()

# Compare the schema of the contact_details
# nested field to the new relationalized table that
# represents it
legislators_combined.select_fields("contact_details").printSchema()
legislators_relationalized.select("l_root_contact_details").toDF().where(
    "id = 10 or id = 75"
).orderBy(["id", "index"]).show()
```

출력

다음 출력을 통해 contact_details을 호출한 중첩 필드의 스키마를 relationalize 변환이 만든 테이블과 비교할 수 있습니다. 테이블 레코드는 id라는 외래 키와 배열의 위치를 나타내는 index 열을 사용하여 기본 테이블로 다시 연결됩니다.

```
dict_keys(['l_root', 'l_root_images', 'l_root_links', 'l_root_other_names',
 'l_root_contact_details', 'l_root_identifiers'])

root
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
| 10|    0|                    fax|           202-225-4160|
| 10|    1|                  phone|           202-225-3436|
```

```
| 75|    0|                     fax|             202-225-6791|
| 75|    1|                   phone|             202-225-2861|
| 75|    2|                 twitter|               RepSamFarr|
+---+-----+------------------------+-------------------------+
```

rename_field

rename_field(oldName, newName, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

DynamicFrame에서 필드 이름을 바꾸고 바꾼 필드로 새로운 DynamicFrame을 반환합니다.

- oldName - 바꾸고자 하는 노드의 전체 경로입니다.

  기존 이름에 점이 있으면 그 주위로 억음 부호(`)를 하지 않는 한 RenameField는 실행되지 않습니다. 예를 들어 this.old.name를 thisNewName으로 바꾸려면 다음과 같이 rename_field를 불러올 수 있습니다.

  ```
  newDyF = oldDyF.rename_field("`this.old.name`", "thisNewName")
  ```

- newName - 전체 경로로써 새로운 이름입니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예: rename_field를 사용하여 **DynamicFrame**의 필드 이름을 변경합니다

이 코드 예제는 rename_field 메서드를 사용하여 DynamicFrame의 필드 이름을 변경합니다. 참고로 이 예제에서는 메서드 체인을 사용하여 여러 필드의 이름을 동시에 바꾸고 있습니다.

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: 데이터 조인 및 관계화 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

예제 코드

```
# Example: Use rename_field to rename fields
# in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Inspect the original orgs schema
orgs = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="organizations_json"
)
print("Original orgs schema: ")
orgs.printSchema()

# Rename fields and view the new schema
orgs = orgs.rename_field("id", "org_id").rename_field("name", "org_name")
print("New orgs schema with renamed fields: ")
orgs.printSchema()
```

출력

```
Original orgs schema:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- classification: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
```

```
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string

New orgs schema with renamed fields:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- classification: string
|-- org_id: string
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string
```

resolveChoice

resolveChoice(specs = None, choice = "" , database = None , table_name = None , transformation_ctx="", info="", stageThreshold=0, totalThreshold=0, catalog_id = None)

이 DynamicFrame로 선택 유형을 해결하고 새로운 DynamicFrame을 반환합니다.

- specs - 해결할 특정 모호성 목록이며, 각각은 (field_path, action)의 튜플 형식입니다.

  resolveChoice를 사용하는 방법은 두 가지입니다. 첫 번째는 specs 인수를 사용하여 특정 필드의 시퀀스와 이를 확인하는 방법을 지정하는 것입니다. resolveChoice의 다른 모드는 choice 인수를 사용하여 모든 ChoiceTypes에 대해 단 하나의 해결책을 지정합니다.

specs의 값은 (field_path, action) 페어로 구성된 튜플로 지정됩니다. field_path 가치는 특정 모호한 요소를 확인하고 action 가치는 관련 해결 방안을 제안합니다. 가능한 작업은 다음과 같습니다.

- cast:*type* - 모든 값을 지정된 유형으로 캐스트하는 시도를 합니다. 예를 들면 cast:int입니다.
- make_cols - 각 고유 유형을 *columnName_type*이라는 이름을 가진 열로 변환합니다. 데이터를 평면화하여 잠재적 모호성을 해결합니다. 예를 들어, columnA가 int 혹은 string이면 해결 방안은 DynamicFrame에서 columnA_int와 columnA_string으로 된 두 열을 생산하는 것입니다.
- make_struct – 데이터를 나타내도록 struct를 사용하여 잠재적 모호성을 해결합니다. 예를 들어, 열에서 데이터가 int 혹은 string이면 make_struct 작업은 결과적인 DynamicFrame의 구조 열을 생성합니다. 각 구조에는 int 및 string가 모두 포함됩니다.
- project:*type* - 모든 데이터를 가능한 데이터 유형 중 하나로 투영하여 잠재적인 모호성을 해결합니다. 예를 들어, 열에서 데이터가 int 혹은 string이면 project:string 작업을 사용하면 모든 int 가치가 문자열로 변환된 결과인 DynamicFrame의 열을 생성합니다.

만약 field_path가 배열을 확인하면, 빈 대괄호를 배열 이름 다음에 만들어 모호성을 피합니다. 예를 들어, 다음과 같은 구조화된 데이터와 작업한다고 가정합시다.

```
"myList": [
  { "price": 100.00 },
  { "price": "$100.00" }
]
```

field_path를 "myList[].price"로 설정하고, action을 "cast:double"로 설정하여 가격의 문자열 버전보다 숫자 버전을 선택할 수 있습니다.

> **Note**
>
> specs 및 choice 파라미터 중 하나만 사용할 수 있습니다. specs 파라미터가 None이 아니면 choice 파라미터는 빈 문자열이어야 합니다. 반대로 choice 파라미터가 빈 문자열이 아니면, specs 파라미터는 None이어야 합니다.

- choice- 모든 ChoiceTypes에 대해 단일 해상도를 지정합니다. 실행 시간 전에 ChoiceTypes의 완전한 목록을 모르는 경우에 사용할 수 있습니다. 이 인수는 specs에 대해 앞에서 나열한 작업 외에 다음 작업을 지원합니다.

  - `match_catalog` – 각 `ChoiceType`을 지정된 Data Catalog 테이블의 해당 유형에 캐스팅해봅니다.
- `database` - `match_catalog` 작업에 사용할 Data Catalog 데이터베이스입니다.
- `table_name` - `match_catalog` 작업에 사용할 Data Catalog 테이블입니다.
- `transformation_ctx` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- `info` - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- `stageThreshold` - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- `totalThreshold` - 오류가 발생되는 변환을 포함하여 최대 발생하는 오류 수입니다(선택). 기본 값은 0이고 이는 프로세스에서 오류가 발생되지 않음을 나타냅니다.
- `catalog_id` - 액세스 중인 Data Catalog의 카탈로그 ID(Data Catalog의 계정 ID)입니다. None(기본값)으로 설정하면 호출 계정의 카탈로그 ID를 사용합니다.

예: resolveChoice를 사용하여 여러 유형이 포함된 열을 처리합니다

이 코드 예제는 `resolveChoice` 메서드를 사용하여 여러 유형의 값이 포함된 `DynamicFrame` 열을 처리하는 방법을 지정합니다. 이 예제는 유형이 다른 열을 처리하는 두 가지 일반적인 방법을 보여줍니다.

- 열을 단일 데이터 유형으로 보내버립니다.
- 모든 유형을 별도의 열에 보관합니다.

예제 데이터 세트

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: ResolveChoice, Lambda, 및 ApplyMapping을 사용한 데이터 준비 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

이 예제에서는 다음 스키마와 함께 `medicare`를 호출한 `DynamicFrame`을 사용합니다.

```
root
|-- drg definition: string
|-- provider id: choice
```

```
|    |-- long
|    |-- string
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string
```

예제 코드

```
# Example: Use resolveChoice to handle
# a column that contains multiple types

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load the input data and inspect the "provider id" column
medicare = glueContext.create_dynamic_frame.from_catalog(
    database="payments", table_name="medicare_hospital_provider_csv"
)
print("Inspect the provider id column:")
medicare.toDF().select("provider id").show()

# Cast provider id to type long
medicare_resolved_long = medicare.resolveChoice(specs=[("provider id", "cast:long")])
print("Schema after casting provider id to type long:")
medicare_resolved_long.printSchema()
medicare_resolved_long.toDF().select("provider id").show()

# Create separate columns
# for each provider id type
medicare_resolved_cols = medicare.resolveChoice(choice="make_cols")
print("Schema after creating separate columns for each type:")
medicare_resolved_cols.printSchema()
```

```
medicare_resolved_cols.toDF().select("provider id_long", "provider id_string").show()
```

출력

```
Inspect the 'provider id' column:
+-----------+
|provider id|
+-----------+
|   [10001,]|
|   [10005,]|
|   [10006,]|
|   [10011,]|
|   [10016,]|
|   [10023,]|
|   [10029,]|
|   [10033,]|
|   [10039,]|
|   [10040,]|
|   [10046,]|
|   [10055,]|
|   [10056,]|
|   [10078,]|
|   [10083,]|
|   [10085,]|
|   [10090,]|
|   [10092,]|
|   [10100,]|
|   [10103,]|
+-----------+
only showing top 20 rows

Schema after casting 'provider id' to type long:
root
|-- drg definition: string
|-- provider id: long
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
```

```
|-- average total payments: string
|-- average medicare payments: string

+-----------+
|provider id|
+-----------+
|      10001|
|      10005|
|      10006|
|      10011|
|      10016|
|      10023|
|      10029|
|      10033|
|      10039|
|      10040|
|      10046|
|      10055|
|      10056|
|      10078|
|      10083|
|      10085|
|      10090|
|      10092|
|      10100|
|      10103|
+-----------+
only showing top 20 rows

Schema after creating separate columns for each type:
root
|-- drg definition: string
|-- provider id_string: string
|-- provider id_long: long
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string
```

```
+----------------+------------------+
|provider id_long|provider id_string|
+----------------+------------------+
|           10001|              null|
|           10005|              null|
|           10006|              null|
|           10011|              null|
|           10016|              null|
|           10023|              null|
|           10029|              null|
|           10033|              null|
|           10039|              null|
|           10040|              null|
|           10046|              null|
|           10055|              null|
|           10056|              null|
|           10078|              null|
|           10083|              null|
|           10085|              null|
|           10090|              null|
|           10092|              null|
|           10100|              null|
|           10103|              null|
+----------------+------------------+
only showing top 20 rows
```

select_fields

select_fields(paths, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

선택된 필드를 포함한 새로운 DynamicFrame을 반환합니다.

- paths – 문자열 목록입니다. 각 문자열은 선택하려는 최상위 노드에 대한 경로입니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

- `totalThreshold` - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예: select_fields를 사용하여 선택한 필드로 새 **DynamicFrame** 생성

다음 코드 예제에서는 `select_fields` 메서드를 사용하여 기존 DynamicFrame에서 선택한 필드 목록으로 새 DynamicFrame을 만드는 방법을 보여줍니다.

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: 데이터 조인 및 관계화 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

```
# Example: Use select_fields to select specific fields from a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Create a new DynamicFrame with chosen fields
names = persons.select_fields(paths=["family_name", "given_name"])
print("Schema for the names DynamicFrame, created with select_fields:")
names.printSchema()
names.toDF().show()
```

출력

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
```

```
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the names DynamicFrame:
root
|-- family_name: string
|-- given_name: string

+-----------+----------+
|family_name|given_name|
+-----------+----------+
|    Collins|   Michael|
|   Huizenga|      Bill|
|    Clawson|    Curtis|
|    Solomon|    Gerald|
|     Rigell|    Edward|
|      Crapo|   Michael|
|      Hutto|      Earl|
```

```
|      Ertel|     Allen|
|     Minish|    Joseph|
|    Andrews|    Robert|
|     Walden|      Greg|
|      Kazen|   Abraham|
|     Turner|   Michael|
|      Kolbe|     James|
|  Lowenthal|      Alan|
|    Capuano|   Michael|
|   Schrader|      Kurt|
|     Nadler|   Jerrold|
|     Graves|       Tom|
|   McMillan|      John|
+-----------+----------+
only showing top 20 rows
```

simplify_ddb_json

`simplify_ddb_json(): DynamicFrame`

DynamicFrame의 중첩된 열 중 특히 DynamoDB JSON 구조에 있는 열을 단순화하고 단순화된 새로운 DynamicFrame을 반환합니다. 목록 유형에 여러 유형 또는 맵 유형이 있는 경우 목록의 요소는 단순화되지 않습니다. 이 변환은 일반적인 unnest 변환과 다르게 작동하는 특정 유형의 변환이며 데이터가 이미 DynamoDB JSON 구조에 있어야 한다는 점에 유의하세요. 자세한 내용은 DynamoDB JSON을 참조하세요.

예를 들어, DynamoDB JSON 구조를 사용하여 내보내기를 읽는 스키마는 다음과 같을 수 있습니다.

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
```

```
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

`simplify_ddb_json()` 변환은 이 구조를 다음과 같이 변환합니다.

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

예제: simplify_ddb_json을 사용하여 DynamoDB JSON 단순화 호출

이 코드 예제에서는 `simplify_ddb_json` 메서드를 사용하여 AWS Glue DynamoDB 내보내기 커넥터를 사용하고, DynamoDB JSON 단순화를 호출하고, 파티션 수를 인쇄합니다.

예제 코드

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

sc = SparkContext()
glueContext = GlueContext(sc)
```

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type = "dynamodb",
    connection_options = {
        'dynamodb.export': 'ddb',
        'dynamodb.tableArn': '<table arn>',
        'dynamodb.s3.bucket': '<bucket name>',
        'dynamodb.s3.prefix': '<bucket prefix>',
        'dynamodb.s3.bucketOwner': '<account_id of bucket>'
    }
)
simplified = dynamicFrame.simplify_ddb_json()
print(simplified.getNumPartitions())
```

spigot

`spigot(path, options={})`

작업에 의해 수행된 변환을 확인하는 데 도움이 되도록 샘플 레코드를 지정된 대상에 기록합니다.

- `path` - 작성할 기본 대상 주소 경로입니다(필수).
- `options` - 옵션을 지정하는 키-값 페어입니다(선택 사항). `"topk"` 옵션은 첫 번째 k 기록이 작성되어야 한다는 것을 명시합니다. `"prob"` 옵션은 특정 레코드를 선택할 가능성(십진수)을 지정합니다. 작성할 레코드를 선택할 때 사용할 수 있습니다.
- `transformation_ctx` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).

예: spigot을 사용하여 **`DynamicFrame`**에서 Amazon S3에 샘플 필드를 작성합니다

이 코드 예제는 `select_fields` 변환을 적용한 후 `spigot` 메서드를 사용하여 Amazon S3 버킷에 샘플 레코드를 작성합니다.

예제 데이터 세트

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: 데이터 조인 및 관계화 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

이 예제에서는 다음 스키마와 함께 `persons`를 호출한 `DynamicFrame`을 사용합니다.

```
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

예제 코드

```
# Example: Use spigot to write sample records
# to a destination during a transformation
# from pyspark.context import SparkContext.
# Replace DOC-EXAMPLE-BUCKET with your own location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
```

```
glueContext = GlueContext(sc)

# Load table data into a DynamicFrame
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)

# Perform the select_fields on the DynamicFrame
persons = persons.select_fields(paths=["family_name", "given_name", "birth_date"])

# Use spigot to write a sample of the transformed data
# (the first 10 records)
spigot_output = persons.spigot(
    path="s3://DOC-EXAMPLE-BUCKET", options={"topk": 10}
)
# Example: Use spigot to write sample records
# to a destination during a transformation
# from pyspark.context import SparkContext.
# Replace DOC-EXAMPLE-BUCKET with your own location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load table data into a DynamicFrame
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)

# Perform the select_fields on the DynamicFrame
persons = persons.select_fields(paths=["family_name", "given_name", "birth_date"])

# Use spigot to write a sample of the transformed data
# (the first 10 records)
spigot_output = persons.spigot(
    path="s3://DOC-EXAMPLE-BUCKET", options={"topk": 10}
)
```

출력

다음은 spigot이 Amazon S3에 작성한 데이터 예제입니다. 예제 코드가 options={"topk": 10}를 지정하였으므로 샘플 데이터에는 처음 10개의 레코드가 포함됩니다.

```
{"family_name":"Collins","given_name":"Michael","birth_date":"1944-10-15"}
{"family_name":"Huizenga","given_name":"Bill","birth_date":"1969-01-31"}
{"family_name":"Clawson","given_name":"Curtis","birth_date":"1959-09-28"}
{"family_name":"Solomon","given_name":"Gerald","birth_date":"1930-08-14"}
{"family_name":"Rigell","given_name":"Edward","birth_date":"1960-05-28"}
{"family_name":"Crapo","given_name":"Michael","birth_date":"1951-05-20"}
{"family_name":"Hutto","given_name":"Earl","birth_date":"1926-05-12"}
{"family_name":"Ertel","given_name":"Allen","birth_date":"1937-11-07"}
{"family_name":"Minish","given_name":"Joseph","birth_date":"1916-09-01"}
{"family_name":"Andrews","given_name":"Robert","birth_date":"1957-08-04"}
```

split_fields

split_fields(paths, name1, name2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

두 개의 DynamicFrames이 포함된 새 DynamicFrameCollection을 반환합니다. 첫 번째 DynamicFrame은 분할된 모든 노드를 포함하고 두 번째는 남겨진 노드를 포함합니다.

- paths - 문자열 목록이며, 각 문자열 목록은 새로운 DynamicFrame으로 스플릿하려는 노드의 전체 경로입니다.
- name1 - 스플릿된 DynamicFrame의 이름 문자열입니다.
- name2 - 스플릿된 특정 노드 이후에 남겨진 DynamicFrame을 위한 이름 문자열입니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예제: split_fields를 사용하여 선택한 필드를 별도의 **DynamicFrame**로 분할합니다

이 코드 예제는 split_fields 메서드를 사용하여 지정된 필드 목록을 별도의 DynamicFrame으로 분할합니다.

예제 데이터 세트

이 예제에서는 legislators_relationalized라는 컬렉션의 l_root_contact_details를 호출한 DynamicFrame을 사용합니다.

l_root_contact_details에는 다음 스키마와 엔트리가 포함됩니다.

```
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                  phone|             202-225-5265|
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
...
```

예제 코드

```
# Example: Use split_fields to split selected
# fields into a separate DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load the input DynamicFrame and inspect its schema
frame_to_split = legislators_relationalized.select("l_root_contact_details")
```

```
print("Inspect the input DynamicFrame schema:")
frame_to_split.printSchema()

# Split id and index fields into a separate DynamicFrame
split_fields_collection = frame_to_split.split_fields(["id", "index"], "left", "right")

# Inspect the resulting DynamicFrames
print("Inspect the schemas of the DynamicFrames created with split_fields:")
split_fields_collection.select("left").printSchema()
split_fields_collection.select("right").printSchema()
```

출력

```
Inspect the input DynamicFrame's schema:
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

Inspect the schemas of the DynamicFrames created with split_fields:
root
|-- id: long
|-- index: int

root
|-- contact_details.val.type: string
|-- contact_details.val.value: string
```

split_rows

`split_rows(comparison_dict, name1, name2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

새로운 DynamicFrame으로 DynamicFrame의 하나 이상의 열을 스플릿합니다.

이 메서드는 2개의 DynamicFrames가 포함된 새 DynamicFrameCollection을 반환합니다. 첫 번째 DynamicFrame은 스플릿된 모든 행을 포함하고 두 번째는 남겨진 열을 포함합니다.

- `comparison_dict` - 열까지 경로의 키와 열 값이 비교된 값의 매핑 비교기를 위한 다른 딕셔너리인 값의 딕셔너리입니다. 예를 들어, `{"age": {">": 10, "<": 20}}`는 나이 열에서 가치가 10 초과 20 미만인 모든 열을 스플릿합니다.

- name1 - 스플릿된 DynamicFrame의 이름 문자열입니다.
- name2 - 스플릿된 특정 노드 이후에 남겨진 DynamicFrame을 위한 이름 문자열입니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예제: split_rows를 사용하여 **DynamicFrame** 행을 분할합니다

이 코드 예제는 split_rows 메서드를 사용하여 id 필드 값을 기준으로 DynamicFrame에서 행을 분할합니다.

예제 데이터 세트

이 예제에서는 legislators_relationalized라는 컬렉션에서 선택한 l_root_contact_details를 호출한 DynamicFrame을 사용합니다.

l_root_contact_details에는 다음 스키마와 엔트리가 포함됩니다.

```
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

+---+-----+------------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+------------------------+-------------------------+
|  1|    0|                   phone|              202-225-5265|
|  1|    1|                 twitter|               kathyhochul|
|  2|    0|                   phone|              202-225-3252|
|  2|    1|                 twitter|             repjackyrosen|
|  3|    0|                     fax|              202-225-1314|
|  3|    1|                   phone|              202-225-3772|
|  3|    2|                 twitter|           MikeRossUpdates|
|  4|    0|                     fax|              202-225-1314|
|  4|    1|                   phone|              202-225-3772|
|  4|    2|                 twitter|           MikeRossUpdates|
```

```
|  5|    0|                     fax|             202-225-1314|
|  5|    1|                   phone|             202-225-3772|
|  5|    2|                 twitter|          MikeRossUpdates|
|  6|    0|                     fax|             202-225-1314|
|  6|    1|                   phone|             202-225-3772|
|  6|    2|                 twitter|          MikeRossUpdates|
|  7|    0|                     fax|             202-225-1314|
|  7|    1|                   phone|             202-225-3772|
|  7|    2|                 twitter|          MikeRossUpdates|
|  8|    0|                     fax|             202-225-1314|
+---+-----+----------------------+-------------------------+
```

예제 코드

```
# Example: Use split_rows to split up
# rows in a DynamicFrame based on value

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Retrieve the DynamicFrame to split
frame_to_split = legislators_relationalized.select("l_root_contact_details")

# Split up rows by ID
split_rows_collection = frame_to_split.split_rows({"id": {">": 10}}, "high", "low")

# Inspect the resulting DynamicFrames
print("Inspect the DynamicFrame that contains IDs < 10")
split_rows_collection.select("low").toDF().show()
print("Inspect the DynamicFrame that contains IDs > 10")
split_rows_collection.select("high").toDF().show()
```

출력

```
Inspect the DynamicFrame that contains IDs < 10
+---+-----+----------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-------------------------+
|  1|    0|                   phone|             202-225-5265|
```

```
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
|  3|    2|                twitter|          MikeRossUpdates|
|  4|    0|                    fax|             202-225-1314|
|  4|    1|                  phone|             202-225-3772|
|  4|    2|                twitter|          MikeRossUpdates|
|  5|    0|                    fax|             202-225-1314|
|  5|    1|                  phone|             202-225-3772|
|  5|    2|                twitter|          MikeRossUpdates|
|  6|    0|                    fax|             202-225-1314|
|  6|    1|                  phone|             202-225-3772|
|  6|    2|                twitter|          MikeRossUpdates|
|  7|    0|                    fax|             202-225-1314|
|  7|    1|                  phone|             202-225-3772|
|  7|    2|                twitter|          MikeRossUpdates|
|  8|    0|                    fax|             202-225-1314|
+---+-----+----------------------+-------------------------+
only showing top 20 rows

Inspect the DynamicFrame that contains IDs > 10
+---+-----+----------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-------------------------+
| 11|    0|                  phone|             202-225-5476|
| 11|    1|                twitter|            RepDavidYoung|
| 12|    0|                  phone|             202-225-4035|
| 12|    1|                twitter|           RepStephMurphy|
| 13|    0|                    fax|             202-226-0774|
| 13|    1|                  phone|             202-225-6335|
| 14|    0|                    fax|             202-226-0774|
| 14|    1|                  phone|             202-225-6335|
| 15|    0|                    fax|             202-226-0774|
| 15|    1|                  phone|             202-225-6335|
| 16|    0|                    fax|             202-226-0774|
| 16|    1|                  phone|             202-225-6335|
| 17|    0|                    fax|             202-226-0774|
| 17|    1|                  phone|             202-225-6335|
| 18|    0|                    fax|             202-226-0774|
| 18|    1|                  phone|             202-225-6335|
| 19|    0|                    fax|             202-226-0774|
| 19|    1|                  phone|             202-225-6335|
```

```
| 20|    0|                      fax|             202-226-0774|
| 20|    1|                    phone|             202-225-6335|
+---+-----+-------------------------+-------------------------+
only showing top 20 rows
```

unbox

unbox(path, format, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0, **options)

DynamicFrame의 문자열 필드를 열고(포맷 재지정) 개봉된 DynamicRecords를 포함한 새로운 DynamicFrame를 반환합니다.

DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 Apache Spark DataFrame의 열과 비슷합니다.

- path - 열고자 하는 문자열 노드의 전체 경로입니다.
- format - 포맷 사양입니다(선택 사항). 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용할 수 있습니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하세요.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- options - 다음 중 한 개 이상을 수행할 수 있습니다.
  - separator - 구분자 문자를 포함한 문자열입니다.
  - escaper - 에스케이프 문자를 포함한 문자열입니다.
  - skipFirst - 첫 번째 인스턴스를 넘어갈지 여부를 나타내는 부울 값입니다.
  - withSchema - 노드 스키마의 JSON 표현을 포함하는 문자열입니다. 스키마의 JSON 표현 형식은 StructType.json()의 출력에 의해 정의됩니다.
  - withHeader - 헤더가 포함되었는지 여부를 나타내는 부울 값입니다.

예제: 개봉하기를 사용하여 문자열 필드를 구조에 개봉하기

이 코드 예제는 unbox 메서드를 사용하여 DynamicFrame의 문자열 필드를 유형 구조 필드로 개봉하거나 포맷을 다시 지정합니다.

예제 데이터 세트

이 예제에서는 다음과 같은 스키마 및 항목과 함께 mapped_with_string를 호출한 DynamicFrame을 사용합니다.

AddressString로 이름이 지정된 필드를 확인하십시오. 이 필드는 예제에서 구조로 개봉되는 필드입니다.

```
root
|-- Average Total Payments: string
|-- AddressString: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string

+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+----------------+--------------------+
|Average Total Payments|       AddressString|Average Covered Charges|      DRG
 Definition|Average Medicare Payments|Hospital Referral Region Description|
 Address|Provider Id|Total Discharges|       Provider Name|
+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+----------------+--------------------+
|              $5777.24|{"Street": "1108 ...|              $32963.07|039 -
 EXTRACRANIA...|                $4763.73|                         AL - Dothan|[36301,
 DOTHAN, [...|      10001|              91|SOUTHEAST ALABAMA...|
```

```
|              $5787.57|{"Street": "2505 ...|              $15131.85|039 -
 EXTRACRANIA...|                 $4976.71|                    AL - Birmingham|[35957,
 BOAZ, [25...|      10005|              14|MARSHALL MEDICAL ...|
|              $5434.95|{"Street": "205 M...|              $37560.37|039 -
 EXTRACRANIA...|                 $4453.79|                    AL - Birmingham|[35631,
 FLORENCE,...|      10006|              24|ELIZA COFFEE MEMO...|
|              $5417.56|{"Street": "50 ME...|              $13998.28|039 -
 EXTRACRANIA...|                 $4129.16|                    AL - Birmingham|[35235,
 BIRMINGHA...|      10011|              25|   ST VINCENT'S EAST|
...
```

예제 코드

```
# Example: Use unbox to unbox a string field
# into a struct in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

unboxed = mapped_with_string.unbox("AddressString", "json")
unboxed.printSchema()
unboxed.toDF().show()
```

출력

```
root
|-- Average Total Payments: string
|-- AddressString: struct
|    |-- Street: string
|    |-- City: string
|    |-- State: string
|    |-- Zip.Code: string
|    |-- Array: array
|    |    |-- element: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
```

```
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string

+----------------------+--------------------+-----------------------
+--------------------+-------------------------+-----------------------------------
+--------------------+-----------+----------------+--------------------+
|Average Total Payments|       AddressString|Average Covered Charges|      DRG
 Definition|Average Medicare Payments|Hospital Referral Region Description|
 Address|Provider Id|Total Discharges|       Provider Name|
+----------------------+--------------------+-----------------------
+--------------------+-------------------------+-----------------------------------
+--------------------+-----------+----------------+--------------------+
|              $5777.24|[1108 ROSS CLARK ...|              $32963.07|039 -
 EXTRACRANIA...|                $4763.73|                         AL - Dothan|[36301,
 DOTHAN, [...|      10001|              91|SOUTHEAST ALABAMA...|
|              $5787.57|[2505 U S HIGHWAY...|              $15131.85|039 -
 EXTRACRANIA...|                $4976.71|                     AL - Birmingham|[35957,
 BOAZ, [25...|      10005|              14|MARSHALL MEDICAL ...|
|              $5434.95|[205 MARENGO STRE...|              $37560.37|039 -
 EXTRACRANIA...|                $4453.79|                     AL - Birmingham|[35631,
 FLORENCE,...|      10006|              24|ELIZA COFFEE MEMO...|
|              $5417.56|[50 MEDICAL PARK ...|              $13998.28|039 -
 EXTRACRANIA...|                $4129.16|                     AL - Birmingham|[35235,
 BIRMINGHA...|      10011|              25|   ST VINCENT'S EAST|
|              $5658.33|[1000 FIRST STREE...|              $31633.27|039 -
 EXTRACRANIA...|                $4851.44|                     AL - Birmingham|[35007,
 ALABASTER...|      10016|              18|SHELBY BAPTIST ME...|
|              $6653.80|[2105 EAST SOUTH ...|              $16920.79|039 -
 EXTRACRANIA...|                $5374.14|                     AL - Montgomery|[36116,
 MONTGOMER...|      10023|              67|BAPTIST MEDICAL C...|
|              $5834.74|[2000 PEPPERELL P...|              $11977.13|039 -
 EXTRACRANIA...|                $4761.41|                     AL - Birmingham|[36801,
 OPELIKA, ...|      10029|              51|EAST ALABAMA MEDI...|
|              $8031.12|[619 SOUTH 19TH S...|              $35841.09|039 -
 EXTRACRANIA...|                $5858.50|                     AL - Birmingham|[35233,
 BIRMINGHA...|      10033|              32|UNIVERSITY OF ALA...|
```

```
|              $6113.38|[101 SIVLEY RD, H...|              $28523.39|039 -
 EXTRACRANIA...|                 $5228.40|                     AL - Huntsville|[35801,
 HUNTSVILL...|      10039|             135| HUNTSVILLE HOSPITAL|
|              $5541.05|[1007 GOODYEAR AV...|              $75233.38|039 -
 EXTRACRANIA...|                 $4386.94|                     AL - Birmingham|[35903,
 GADSDEN, ...|      10040|              34|GADSDEN REGIONAL ...|
|              $5461.57|[600 SOUTH THIRD ...|              $67327.92|039 -
 EXTRACRANIA...|                 $4493.57|                     AL - Birmingham|[35901,
 GADSDEN, ...|      10046|              14|RIVERVIEW REGIONA...|
|              $5356.28|[4370 WEST MAIN S...|              $39607.28|039 -
 EXTRACRANIA...|                 $4408.20|                         AL - Dothan|[36305,
 DOTHAN, [...|      10055|              45|    FLOWERS HOSPITAL|
|              $5374.65|[810 ST VINCENT'S...|              $22862.23|039 -
 EXTRACRANIA...|                 $4186.02|                     AL - Birmingham|[35205,
 BIRMINGHA...|      10056|              43|ST VINCENT'S BIRM...|
|              $5366.23|[400 EAST 10TH ST...|              $31110.85|039 -
 EXTRACRANIA...|                 $4376.23|                     AL - Birmingham|[36207,
 ANNISTON,...|      10078|              21|NORTHEAST ALABAMA...|
|              $5282.93|[1613 NORTH MCKEN...|              $25411.33|039 -
 EXTRACRANIA...|                 $4383.73|                         AL - Mobile|[36535,
 FOLEY, [1...|      10083|              15|SOUTH BALDWIN REG...|
|              $5676.55|[1201 7TH STREET ...|               $9234.51|039 -
 EXTRACRANIA...|                 $4509.11|                     AL - Huntsville|[35609,
 DECATUR, ...|      10085|              27|DECATUR GENERAL H...|
|              $5930.11|[6801 AIRPORT BOU...|              $15895.85|039 -
 EXTRACRANIA...|                 $3972.85|                         AL - Mobile|[36608,
 MOBILE, [...|      10090|              27| PROVIDENCE HOSPITAL|
|              $6192.54|[809 UNIVERSITY B...|              $19721.16|039 -
 EXTRACRANIA...|                 $5179.38|                     AL - Tuscaloosa|[35401,
 TUSCALOOS...|      10092|              31|D C H REGIONAL ME...|
|              $4968.00|[750 MORPHY AVENU...|              $10710.88|039 -
 EXTRACRANIA...|                 $3898.88|                         AL - Mobile|[36532,
 FAIRHOPE,...|      10100|              18|     THOMAS HOSPITAL|
|              $5996.00|[701 PRINCETON AV...|              $51343.75|039 -
 EXTRACRANIA...|                 $4962.45|                     AL - Birmingham|[35211,
 BIRMINGHA...|      10103|              33|BAPTIST MEDICAL C...|
+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+----------+---------------+--------------------+
only showing top 20 rows
```

결합

union(frame1, frame2, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

두 개의 DynamicFrame을 결합합니다. 두 입력 DynamicFrames의 모든 레코드를 포함하는 DynamicFrame을 반환합니다. 이 변환은 두 DataFrame을 동일한 데이터와 결합하여 다른 결과를 반환할 수 있습니다. Spark DataFrame 통합 동작이 필요한 경우 toDF을(를) 사용하는 것을 고려해 보세요.

- frame1— 결합할 첫 번째 DynamicFrame.
- frame2— 결합할 두 번째 DynamicFrame.
- transformation_ctx – (선택 사항) 통계/상태 정보를 확인하는 데 사용되는 고유 문자열
- info - (선택 사항) 변환에 따른 오류 관련 문자열
- stageThreshold— (선택 사항) 처리 오류가 발생할 때까지 변환에서 발생한 최대 오류 수
- totalThreshold— (선택 사항) 처리 오류가 발생할 때까지의 총 최대 오류 수.

unnest

unnest(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

DynamicFrame에서 중첩된 객체를 중첩시키지 않고 상위 객체로 만들어서 새로운 중첩되지 않은 DynamicFrame을 반환합니다.

- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.
- totalThreshold - 프로세스에서 오류가 발생해야 하는 이 변환을 포함하여 최대 발생한 오류 수입니다(선택 사항). 기본값은 0이며, 이는 프로세스에 오류가 발생하지 않아야 함을 나타냅니다.

예제: 중첩 해제를 사용하여 중첩된 필드를 최상위 필드로 전환합니다

이 코드 예제는 unnest 메서드를 사용하여 DynamicFrame의 모든 중첩된 필드를 최상위 필드로 병합합니다.

예제 데이터 세트

이 예제에서는 다음 스키마와 함께 mapped_medicare를 호출하는 DynamicFrame을 사용합니다. Address 필드는 중첩된 데이터를 포함하는 유일한 필드라는 점에 유의하십시오.

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

예제 코드

```
# Example: Use unnest to unnest nested
# objects in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Unnest all nested fields
unnested = mapped_medicare.unnest()
unnested.printSchema()
```

출력

```
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
```

```
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address.Zip.Code: string
|-- Address.City: string
|-- Address.Array: array
|    |-- element: string
|-- Address.State: string
|-- Address.Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

unnest_ddb_json

DynamoDB JSON 구조의 DynamicFrame에 있는 중첩된 열을 한정해서 중첩 해제하고, 중첩되지 않은 새 DynamicFrame을 반환합니다. 구조체 유형 배열인 열은 중첩 해제되지 않습니다. 이 변환은 일반적인 unnest 변환과는 다르게 작동하는 특정 유형의 중첩 해제 변환이며, 데이터가 이미 DynamoDB JSON 구조에 있어야 합니다. 자세한 내용은 DynamoDB JSON을 참조하세요.

unnest_ddb_json(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 이 변환에 따른 오류 보고 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생되는 변환 중에 발생하는 오류 수 (선택: 오류가 발생되지 않은 변환을 나타내는 기본값은 0입니다).
- totalThreshold - 오류가 발생되는 변환을 포함하여 최대 발생하는 오류 수 (선택: 오류가 발생되지 않은 변환을 나타내는 기본값은 0입니다).

예를 들어, DynamoDB JSON 구조를 사용하여 내보내기를 읽는 스키마는 다음과 같을 수 있습니다.

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
|    |    |-- N: string
```

```
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

unnest_ddb_json() 변환은 이 구조를 다음과 같이 변환합니다.

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

아래 코드 예제에서는 AWS Glue DynamoDB 내보내기 커넥터를 사용하고, DynamoDB JSON unnest를 호출하고, 파티션 수를 인쇄하는 방법을 보여줍니다.

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

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

dynamicFrame = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": "<test_source>",
        "dynamodb.s3.bucket": "<bucket name>",
        "dynamodb.s3.prefix": "<bucket prefix>",
        "dynamodb.s3.bucketOwner": "<account_id>",
    }
)
unnested = dynamicFrame.unnest_ddb_json()
print(unnested.getNumPartitions())

job.commit()
```

write

write(connection_type, connection_options, format, format_options, accumulator_size)

이 DynamicFrame의 GlueContext 클래스에서 지정된 연결 유형의 DataSink(객체)를 얻어 DynamicFrame 내용을 포맷하고 작성하는 데 사용합니다. 명시된 대로 포맷되고 작성된 새로운 DynamicFrame을 반환합니다.

- connection_type - 사용할 연결 유형입니다. 유효한 값에는 s3, mysql, postgresql, redshift, sqlserver 및 oracle가 있습니다.
- connection_options - 사용할 연결 옵션입니다(선택 사항). s3의 connection_type의 경우, Amazon S3 경로가 정의됩니다.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

JDBC 연결의 경우, 몇 까지 속성이 정의되어야 합니다. 단, 데이터베이스 이름이 URL의 일부여야 합니다. 연결 옵션에 선택적으로 포함될 수 있습니다.

> **Warning**
>
> 스크립트에 암호를 저장하는 것은 권장되지 않습니다. AWS Secrets Manager 또는 AWS Glue 데이터 카탈로그에서 데이터를 검색할 때 boto3 사용을 고려합니다.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

- format - 포맷 사양입니다(선택 사항). 여러 포맷을 지원하는 Amazon Simple Storage Service(Amazon S3) 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- format_options - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- accumulator_size - (선택 사항) 사용할 누적 가능한 크기(바이트)입니다.

- 오류 -

- assertErrorThreshold
- errorsAsDynamicFrame
- errorsCount
- stageErrorsCount

assertErrorThreshold

`assertErrorThreshold( )` - 이 `DynamicFrame`을 생성한 변환의 오류에 대한 어설션입니다. 기본 `DataFrame`으로부터 `Exception`을 반환합니다.

errorsAsDynamicFrame

`errorsAsDynamicFrame( )` - 내부에 중첩된 오류 기록을 보유한 `DynamicFrame`을 반환합니다.

예: errorsAsDynamicFrame을 사용하여 오류 레코드 보기

다음 코드 예제에서는 `errorsAsDynamicFrame` 메서드를 사용하여 `DynamicFrame`에 대한 오류 레코드를 보는 방법을 보여 줍니다.

예제 데이터 세트

이 예제에서는 Amazon S3에 JSON으로 업로드할 수 있는 다음 데이터 세트를 사용합니다. 두 번째 레코드의 형식이 잘못되었습니다. 잘못된 형식의 데이터는 일반적으로 SparkSQL을 사용할 때 파일 구문 분석을 중단합니다. 그러나 `DynamicFrame`은 잘못된 형식 문제를 인식하고 잘못된 형식 행을 개별적으로 처리할 수 있는 오류 레코드로 바꿉니다.

```
{"id": 1, "name": "george", "surname": "washington", "height": 178}
{"id": 2, "name": "benjamin", "surname": "franklin",
{"id": 3, "name": "alexander", "surname": "hamilton", "height": 171}
{"id": 4, "name": "john", "surname": "jay", "height": 190}
```

예제 코드

```
# Example: Use errorsAsDynamicFrame to view error records.
# Replace s3://DOC-EXAMPLE-S3-BUCKET/error_data.json with your location.

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

```
# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create errors DynamicFrame, view schema
errors = glueContext.create_dynamic_frame.from_options(
    "s3", {"paths": ["s3://DOC-EXAMPLE-S3-BUCKET/error_data.json"]}, "json"
)
print("Schema of errors DynamicFrame:")
errors.printSchema()

# Show that errors only contains valid entries from the dataset
print("errors contains only valid records from the input dataset (2 of 4 records)")
errors.toDF().show()

# View errors
print("Errors count:", str(errors.errorsCount()))
print("Errors:")
errors.errorsAsDynamicFrame().toDF().show()

# View error fields and error data
error_record = errors.errorsAsDynamicFrame().toDF().head()

error_fields = error_record["error"]
print("Error fields: ")
print(error_fields.asDict().keys())

print("\nError record data:")
for key in error_fields.asDict().keys():
    print("\n", key, ": ", str(error_fields[key]))
```

출력

```
Schema of errors DynamicFrame:
root
|-- id: int
|-- name: string
|-- surname: string
|-- height: int

errors contains only valid records from the input dataset (2 of 4 records)
+---+------+----------+------+
```

```
| id|  name|   surname|height|
+---+------+----------+------+
|  1|george|washington|   178|
|  4|  john|       jay|   190|
+---+------+----------+------+

Errors count: 1
Errors:
+--------------------+
|               error|
+--------------------+
|[[  File "/tmp/20...|
+--------------------+

Error fields:
dict_keys(['callsite', 'msg', 'stackTrace', 'input', 'bytesread', 'source',
 'dynamicRecord'])

Error record data:

 callsite :  Row(site='  File "/tmp/2060612586885849088", line 549, in <module>\n
 sys.exit(main())\n  File "/tmp/2060612586885849088", line 523, in main\n    response
 = handler(content)\n  File "/tmp/2060612586885849088", line 197, in execute_request
\n    result = node.execute()\n  File "/tmp/2060612586885849088", line 103, in
 execute\n    exec(code, global_dict)\n  File "<stdin>", line 10, in <module>\n
 File "/opt/amazon/lib/python3.6/site-packages/awsglue/dynamicframe.py", line 625, in
 from_options\n    format_options, transformation_ctx, push_down_predicate, **kwargs)\n
  File "/opt/amazon/lib/python3.6/site-packages/awsglue/context.py", line 233, in
 create_dynamic_frame_from_options\n    source.setFormat(format, **format_options)\n',
 info='')

 msg :  error in jackson reader

 stackTrace :  com.fasterxml.jackson.core.JsonParseException: Unexpected character
 ('{' (code 123)): was expecting either valid name character (for unquoted name) or
 double-quote (for quoted) to start field name
 at [Source: com.amazonaws.services.glue.readers.BufferedStream@73492578; line: 3,
 column: 2]
 at com.fasterxml.jackson.core.JsonParser._constructError(JsonParser.java:1581)
 at
 com.fasterxml.jackson.core.base.ParserMinimalBase._reportError(ParserMinimalBase.java:533)
 at
 com.fasterxml.jackson.core.base.ParserMinimalBase._reportUnexpectedChar(ParserMinimalBase.java
```

```
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser._handleOddName(UTF8StreamJsonParser.java:
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser._parseName(UTF8StreamJsonParser.java:1650
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser.nextToken(UTF8StreamJsonParser.java:740)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun$hasNextGoodToken
$1.apply(JacksonReader.scala:57)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun$hasNextGoodToken
$1.apply(JacksonReader.scala:57)
 at scala.collection.Iterator$$anon$9.next(Iterator.scala:162)
 at scala.collection.Iterator$$anon$16.hasNext(Iterator.scala:599)
 at scala.collection.Iterator$$anon$16.hasNext(Iterator.scala:598)
 at scala.collection.Iterator$class.foreach(Iterator.scala:891)
 at scala.collection.AbstractIterator.foreach(Iterator.scala:1334)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun
$1.apply(JacksonReader.scala:120)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun
$1.apply(JacksonReader.scala:116)
 at
 com.amazonaws.services.glue.DynamicRecordBuilder.handleErr(DynamicRecordBuilder.scala:209)
 at
 com.amazonaws.services.glue.DynamicRecordBuilder.handleErrorWithException(DynamicRecordBuilder
 at
 com.amazonaws.services.glue.readers.JacksonReader.nextFailSafe(JacksonReader.scala:116)
 at com.amazonaws.services.glue.readers.JacksonReader.next(JacksonReader.scala:109)
 at com.amazonaws.services.glue.readers.JSONReader.next(JSONReader.scala:247)
 at
 com.amazonaws.services.glue.hadoop.TapeHadoopRecordReaderSplittable.nextKeyValue(TapeHadoopRec
 at org.apache.spark.rdd.NewHadoopRDD$$anon$1.hasNext(NewHadoopRDD.scala:230)
 at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:462)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:462)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at org.apache.spark.sql.execution.SparkPlan$$anonfun$2.apply(SparkPlan.scala:255)
 at org.apache.spark.sql.execution.SparkPlan$$anonfun$2.apply(SparkPlan.scala:247)
 at org.apache.spark.rdd.RDD$$anonfun$mapPartitionsInternal$1$$anonfun$apply
$24.apply(RDD.scala:836)
```

```
 at org.apache.spark.rdd.RDD$$anonfun$mapPartitionsInternal$1$$anonfun$apply
$24.apply(RDD.scala:836)
 at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
 at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324)
 at org.apache.spark.rdd.RDD.iterator(RDD.scala:288)
 at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
 at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324)
 at org.apache.spark.rdd.RDD.iterator(RDD.scala:288)
 at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
 at org.apache.spark.scheduler.Task.run(Task.scala:121)
 at org.apache.spark.executor.Executor$TaskRunner$$anonfun$10.apply(Executor.scala:408)
 at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
 at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:414)
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
 at java.lang.Thread.run(Thread.java:750)


 input :

 bytesread :  252

 source :

 dynamicRecord :  Row(id=2, name='benjamin', surname='franklin')
```

errorsCount

`errorsCount( )` - DynamicFrame의 전체 오류 수를 반환합니다.

stageErrorsCount

`stageErrorsCount` - 이 DynamicFrame을 생성하는 과정에서 발생한 오류 수를 반환합니다.

DynamicFrameCollection 클래스

`DynamicFrameCollection`는 키가 DynamicFrames의 이름이고 값이 DynamicFrame 객체인 DynamicFrame 클래스 객체의 목록입니다.

__init__

__init__(dynamic_frames, glue_ctx)

- dynamic_frames – DynamicFrame 클래스 객체의 딕셔너리입니다.
- glue_ctx – GlueContext 클래스 객체입니다.

키

keys( ) - 이 컬렉션에 키 목록을 반환합니다. 일반적으로 DynamicFrame 값과 대응하는 이름을 포함합니다.

값

values(key) - 이 컬렉션에 DynamicFrame 값의 목록을 반환합니다.

Select

select(key)

지정된 키(일반적으로 DynamicFrame의 이름)에 해당되는 DynamicFrame를 반환합니다.

- key - DynamicFrameCollection의 키이고 보통 DynamicFrame의 이름을 나타냅니다.

맵

map(callable, transformation_ctx="")

입장 함수를 사용하여 이 컬렉션의 DynamicFrames 기반 새로운 DynamicFrameCollection를 생성하고 반환합니다.

- callable - DynamicFrame과 지정 변환 내용을 파라미터로 여기고 DynamicFrame을 반환하는 함수입니다.
- transformation_ctx - 호출 기능에 사용되는 변환 내용입니다(선택 사항).

flatmap

flatmap(f, transformation_ctx="")

입장 함수를 사용하여 이 컬렉션의 DynamicFrames 기반 새로운 DynamicFrameCollection를 생성하고 반환합니다.

- f – DynamicFrame을 파라미터로 여기고 DynamicFrame 혹은 DynamicFrameCollection을 반환하는 함수입니다.
- transformation_ctx - 함수에 사용되는 변환 내용입니다(선택 사항).

DynamicFrameWriter 클래스

메서드

- __init__
- from_options
- from_catalog
- from_jdbc_conf

__init__

__init__(glue_context)

- glue_context – 사용할 GlueContext 클래스입니다.

from_options

from_options(frame, connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")

지정한 연결 및 포맷을 사용하여 DynamicFrame를 작성합니다.

- frame - 작성할 DynamicFrame입니다.
- connection_type - 연결 유형입니다. 유효한 값에는 s3, mysql, postgresql, redshift, sqlserver 및 oracle가 있습니다.
- connection_options - 경로 및 데이터베이스 테이블과 같은 연결 옵션입니다(선택 사항). s3의 connection_type의 경우, Amazon S3 경로가 정의됩니다.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

JDBC 연결의 경우, 몇 까지 속성이 정의되어야 합니다. 단, 데이터베이스 이름이 URL의 일부여야 합니다. 연결 옵션에 선택적으로 포함될 수 있습니다.

> **Warning**
>
> 스크립트에 암호를 저장하는 것은 권장되지 않습니다. AWS Secrets Manager 또는 AWS Glue 데이터 카탈로그에서 데이터를 검색할 때 boto3 사용을 고려합니다.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

dbtable 속성은 JDBC 테이블의 이름입니다. 데이터베이스 내의 스키마를 지원하는 JDBC 데이터 스토어의 경우 schema.table-name에 대해 지정합니다. 스키마가 제공되지 않으면 기본 "퍼블릭" 스키마가 사용됩니다.

자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 단원을 참조하십시오.

- format - 포맷 사양입니다(선택 사항). 여러 포맷을 지원하는 Amazon Simple Storage Service(Amazon S3) 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- format_options - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).

from_catalog

from_catalog(frame, name_space, table_name, redshift_tmp_dir="", transformation_ctx="")

지정한 카탈로그 데이터베이스 및 테이블 이름을 사용하여 DynamicFrame를 작성합니다.

- frame - 작성할 DynamicFrame입니다.
- name_space - 사용할 데이터베이스입니다.
- table_name – 사용할 table_name입니다.
- redshift_tmp_dir – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).
- additional_options - AWS Glue에 제공되는 추가 옵션입니다.

Lake Formation 관리형 테이블에 쓰려면 다음과 같은 추가 옵션을 사용할 수 있습니다.

- `transactionId` - (문자열) 관리형 테이블에 쓰기를 수행할 트랜잭션 ID입니다. 이 트랜잭션은 이미 커밋되거나 중단되었을 수 없습니다. 그렇지 않으면 쓰기에 실패합니다.
- `callDeleteObjectsOnCancel` - (부울, 선택 사항) `true`(기본값)로 설정된 경우 AWS Glue는 객체가 Amazon S3에 기록된 후 `DeleteObjectsOnCancel` API를 자동으로 호출합니다. 자세한 내용은 AWS Lake Formation 개발자 가이드의 DeleteObjectsOnCancel을 참조하세요.

Example 예: Lake Formation의 관리형 테이블에 쓰기

```
txId = glueContext.start_transaction(read_only=False)
glueContext.write_dynamic_frame.from_catalog(
    frame=dyf,
    database = db,
    table_name = tbl,
    transformation_ctx = "datasource0",
    additional_options={"transactionId":txId})
...
glueContext.commit_transaction(txId)
```

from_jdbc_conf

`from_jdbc_conf(frame, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx="")`

지정한 JDBC 연결 정보를 사용하여 `DynamicFrame`를 작성합니다.

- `frame` - 작성할 `DynamicFrame`입니다.
- `catalog_connection` - 사용할 카탈로그 연결입니다.
- `connection_options` - 경로 및 데이터베이스 테이블과 같은 연결 옵션입니다(선택 사항).
- `redshift_tmp_dir` – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).

write_dynamic_frame의 예

이 예에서는 `connection_options`의 POSIX 경로 인수와 함께 S3의 `connection_type`을 사용하여 로컬 스토리지에 쓰기를 허용하는 출력을 로컬로 씁니다.

```
glueContext.write_dynamic_frame.from_options(\
frame = dyf_splitFields,\
connection_options = {'path': '/home/glue/GlueLocalOutput/'},\
connection_type = 's3',\
format = 'json')
```

DynamicFrameReader 클래스

- 메서드 -

- __init__
- from_rdd
- from_options
- from_catalog

__init__

__init__(glue_context)

- glue_context – 사용할 GlueContext 클래스입니다.

from_rdd

from_rdd(data, name, schema=None, sampleRatio=None)

탄력적 분산형 데이터셋(RDD)로부터 DynamicFrame을 판독합니다.

- data - 읽어야 할 데이터 집합입니다.
- name - 읽어야 할 이름입니다.
- schema - 판독할 스키마입니다(선택 사항).
- sampleRatio - 사용할 예제 비율입니다(선택 사항).

from_options

from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")

지정한 연결 및 포맷을 사용하여 DynamicFrame를 판독합니다.

- `connection_type` - 연결 유형입니다. 유효한 값에는 `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `dynamodb`, `snowflake`가 있습니다.
- `connection_options` - 경로 및 데이터베이스 테이블과 같은 연결 옵션입니다(선택 사항). 자세한 내용은 Spark를 위한 AWS Glue의 ETL 연결 유형 및 옵션을 참조하세요. `s3`의 `connection_type`의 경우, Amazon S3 경로가 배열에 정의됩니다.

```
connection_options = {"paths": [ "s3://amzn-s3-demo-bucket/object_a", "s3://amzn-s3-demo-bucket/object_b"]}
```

JDBC 연결의 경우, 몇 까지 속성이 정의되어야 합니다. 단, 데이터베이스 이름이 URL의 일부여야 합니다. 연결 옵션에 선택적으로 포함될 수 있습니다.

> **Warning**
>
> 스크립트에 암호를 저장하는 것은 권장되지 않습니다. AWS Secrets Manager 또는 AWS Glue 데이터 카탈로그에서 데이터를 검색할 때 `boto3` 사용을 고려합니다.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

병렬 읽기를 수행하는 JDBC 연결의 경우 해시 필드 옵션을 설정할 수 있습니다. 예:

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path" , "hashfield": "month"}
```

자세한 내용은 JDBC 테이블을 병렬로 읽기 섹션을 참조하세요.

- `format` - 포맷 사양입니다(선택 사항). 여러 포맷을 지원하는 Amazon Simple Storage Service(Amazon S3) 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- `format_options` - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).

- push_down_predicate - 데이터 집합 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 자세한 내용은 푸시다운 조건자를 사용하여 사전 필터링을 참조하세요.

from_catalog

from_catalog(database, table_name, redshift_tmp_dir="", transformation_ctx="", push_down_predicate="", additional_options={})

지정한 카탈로그 네임스페이스 및 테이블 이름을 사용하여 DynamicFrame를 판독합니다.

- database - 읽어야 할 데이터베이스입니다.
- table_name - 읽어올 테이블의 이름입니다.
- redshift_tmp_dir – 사용할 Amazon Redshift 임시 디렉터리(Redshift에서 데이터를 읽지 않는 경우 선택 사항)입니다.
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).
- push_down_predicate - 데이터 집합 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 자세한 내용은 푸시다운 조건자를 사용하여 예비 필터링 섹션을 참조하세요.
- additional_options - AWS Glue에 제공되는 추가 옵션입니다.
  - 병렬 읽기를 수행하는 JDBC 연결을 사용하기 위해 hashfield, hashexpression 또는 hashpartitions 옵션을 설정할 수 있습니다. 예:

    ```
    additional_options = {"hashfield": "month"}
    ```

    자세한 내용은 JDBC 테이블을 병렬로 읽기 섹션을 참조하세요.
  - 인덱스 열을 기준으로 필터링할 카탈로그 표현식을 전달하려면 catalogPartitionPredicate 옵션을 볼 수 있습니다.

    catalogPartitionPredicate - 카탈로그 표현식을 전달하여 인덱스 열을 기준으로 필터링할 수 있습니다. 이렇게 하면 필터링이 서버 측으로 푸시됩니다. 자세한 내용은 AWS Glue 파티션 인덱스를 참조하세요. push_down_predicate와 catalogPartitionPredicate는 다른 구문을 사용합니다. 전자는 Spark SQL 표준 구문을 사용하고 후자는 JSQL 구문 분석기를 사용합니다.

    자세한 내용은 AWS Glue에서 ETL 출력의 파티션 관리 섹션을 참조하세요.

GlueContext 클래스

Apache Spark SparkContext 객체를 포함하여 Apache Spark 플랫폼과 상호작용하기 위한 원리를 제공합니다.

__init__

`__init__(sparkContext)`

- `sparkContext` - 사용할 아파치 스파크 내용입니다.

[생성 중]

- __init__
- getSource
- create_dynamic_frame_from_rdd
- create_dynamic_frame_from_catalog
- create_dynamic_frame_from_options
- create_sample_dynamic_frame_from_catalog
- create_sample_dynamic_frame_from_options
- add_ingestion_time_columns
- create_data_frame_from_catalog
- create_data_frame_from_options
- forEachBatch

getSource

`getSource(connection_type, transformation_ctx = "", **options)`

외부 소스에서 `DataSource`를 읽을 때 사용되는 `DynamicFrames` 객체를 생성합니다.

- `connection_type` - Amazon Simple Storage Service(Amazon S3), Amazon Redshift 및 JDBC와 같이 사용할 연결 유형입니다. 유효한 값에는 `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` 및 `dynamodb`가 있습니다.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).
- `options` - 선택적 이름-값 페어의 모음입니다. 자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.

다음은 getSource를 사용한 예입니다.

```
>>> data_source = context.getSource("file", paths=["/in/path"])
>>> data_source.setFormat("json")
>>> myFrame = data_source.getFrame()
```

create_dynamic_frame_from_rdd

create_dynamic_frame_from_rdd(data, name, schema=None, sample_ratio=None, transformation_ctx="")

Apache Spark Resilient Distributed Dataset(RDD)로부터 생성된 DynamicFrame을 반환합니다.

- data - 사용할 데이터 원본입니다.
- name - 사용할 데이터 이름입니다.
- schema - 사용할 스키마입니다(선택 사항).
- sample_ratio - 사용할 예제 비율입니다(선택 사항).
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).

create_dynamic_frame_from_catalog

create_dynamic_frame_from_catalog(database, table_name, redshift_tmp_dir, transformation_ctx = "", push_down_predicate= "", additional_options = {}, catalog_id = None)

데이터 카탈로그 데이터베이스 및 테이블 이름을 사용하여 생성된 DynamicFrame을 반환합니다. 이 메서드를 사용하는 경우 지정된 AWS Glue 데이터 카탈로그 테이블의 테이블 속성을 통해 format_options를 제공하고, additional_options 인수를 통해 기타 옵션을 제공합니다.

- Database - 읽어야 할 데이터베이스입니다.
- table_name - 읽어올 테이블의 이름입니다.
- redshift_tmp_dir – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).
- push_down_predicate - 데이터 집합 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 지원되는 소스 및 제한 사항은 AWS Glue ETL의 푸시다운으로 읽기 최적화를 참조하세요. 자세한 내용은 푸시다운 조건자를 사용하여 예비 필터링 섹션을 참조하세요.

- additional_options - 선택적 이름-값 페어의 모음입니다. 가능한 옵션에는 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션에 나열된 옵션이 포함됩니다(endpointUrl, streamName, bootstrap.servers, security.protocol, topicName, classification 및 delimiter 제외). 지원되는 또 다른 옵션은 catalogPartitionPredicate입니다.

  catalogPartitionPredicate - 카탈로그 표현식을 전달하여 인덱스 열을 기준으로 필터링할 수 있습니다. 이렇게 하면 필터링이 서버 측으로 푸시됩니다. 자세한 내용은 AWS Glue 파티션 인덱스를 참조하세요. push_down_predicate와 catalogPartitionPredicate는 다른 구문을 사용합니다. 전자는 Spark SQL 표준 구문을 사용하고 후자는 JSQL 구문 분석기를 사용합니다.
- catalog_id - 액세스 중인 데이터 카탈로그의 카탈로그 ID(계정 ID)입니다. None인 경우 호출자의 기본 계정 ID가 사용됩니다.

create_dynamic_frame_from_options

create_dynamic_frame_from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")

지정된 연결 및 포맷으로 생성된 DynamicFrame을 반환합니다.

- connection_type - Amazon S3, Amazon Redshift 및 JDBC 등의 연결 유형입니다. 유효한 값에는 s3, mysql, postgresql, redshift, sqlserver, oracle 및 dynamodb가 있습니다.
- connection_options - 경로 및 데이터베이스 테이블과 같은 연결 옵션(선택 사항). s3의 connection_type인 경우, Amazon S3 경로 목록이 정의됩니다.

```
connection_options = {"paths": ["s3://aws-glue-target/temp"]}
```

  JDBC 연결의 경우, 몇 까지 속성이 정의되어야 합니다. 단, 데이터베이스 이름이 URL의 일부여야 합니다. 연결 옵션에 선택적으로 포함될 수 있습니다.

> **Warning**
>
> 스크립트에 암호를 저장하는 것은 권장되지 않습니다. AWS Secrets Manager 또는 AWS Glue 데이터 카탈로그에서 데이터를 검색할 때 boto3 사용을 고려합니다.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-
path"}
```

dbtable 속성은 JDBC 테이블의 이름입니다. 데이터베이스 내의 스키마를 지원하는 JDBC 데이터 스토어의 경우 schema.table-name에 대해 지정합니다. 스키마가 제공되지 않으면 기본 "퍼블릭" 스키마가 사용됩니다.

자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.

- format - 형식 사양. 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- format_options - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).
- push_down_predicate - 데이터 집합 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 지원되는 소스 및 제한 사항은 AWS Glue ETL의 푸시다운으로 읽기 최적화를 참조하세요. 자세한 내용은 푸시다운 조건자를 사용하여 사전 필터링을 참조하세요.

create_sample_dynamic_frame_from_catalog

create_sample_dynamic_frame_from_catalog(database, table_name, num, redshift_tmp_dir, transformation_ctx = "", push_down_predicate= "", additional_options = {}, sample_options = {}, catalog_id = None)

데이터 카탈로그 데이터베이스와 테이블 이름을 사용하여 생성된 샘플 DynamicFrame을 반환합니다. DynamicFrame에는 데이터 소스의 첫 번째 num 레코드만 포함됩니다.

- database - 읽어야 할 데이터베이스입니다.
- table_name - 읽어올 테이블의 이름입니다.
- num - 반환된 샘플 동적 프레임의 최대 레코드 수입니다.
- redshift_tmp_dir – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).
- push_down_predicate - 데이터 집합 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 자세한 내용은 푸시다운 조건자를 사용하여 예비 필터링 섹션을 참조하세요.

- additional_options - 선택적 이름-값 페어의 모음입니다. 가능한 옵션에는 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션에 나열된 옵션이 포함됩니다(endpointUrl, streamName, bootstrap.servers, security.protocol, topicName, classification 및 delimiter 제외).
- sample_options - 샘플링 동작을 제어하는 파라미터(선택 사항)입니다. Amazon S3 소스에 현재 사용 가능한 파라미터:
  - maxSamplePartitions - 샘플링에서 읽을 최대 파티션 수입니다. 기본값은 10입니다.
  - maxSampleFilesPerPartition - 샘플링이 한 파티션에서 읽을 최대 파일 수입니다. 기본값은 10입니다.

    이러한 파라미터는 파일 나열에 소요되는 시간을 줄이는 데 도움이 됩니다. 예를 들어 데이터 세트에 1,000개의 파티션이 있고 각 파티션에 10개의 파일이 있다고 가정합니다. maxSamplePartitions = 10 및 maxSampleFilesPerPartition = 10으로 설정하면 10,000개의 파일을 모두 나열하는 대신 샘플링은 각각에 처음 10개의 파일이 있는 처음 10개의 파티션만 나열하고 읽습니다(10*10 = 총 100개의 파일).
- catalog_id - 액세스 중인 데이터 카탈로그의 카탈로그 ID(데이터 카탈로그의 계정 ID)입니다. 기본적으로 None으로 설정됩니다. None인 경우 서비스에 있는 호출 계정의 카탈로그 ID가 기본 설정됩니다.

create_sample_dynamic_frame_from_options

create_sample_dynamic_frame_from_options(connection_type, connection_options={}, num, sample_options={}, format=None, format_options={}, transformation_ctx = "")

지정된 연결과 형식으로 생성된 샘플 DynamicFrame을 반환합니다. DynamicFrame에는 데이터 소스의 첫 번째 num 레코드만 포함됩니다.

- connection_type - Amazon S3, Amazon Redshift 및 JDBC 등의 연결 유형입니다. 유효한 값에는 s3, mysql, postgresql, redshift, sqlserver, oracle 및 dynamodb가 있습니다.
- connection_options - 경로 및 데이터베이스 테이블과 같은 연결 옵션(선택 사항). 자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.
- num - 반환된 샘플 동적 프레임의 최대 레코드 수입니다.
- sample_options - 샘플링 동작을 제어하는 파라미터(선택 사항)입니다. Amazon S3 소스에 현재 사용 가능한 파라미터:
  - maxSamplePartitions - 샘플링에서 읽을 최대 파티션 수입니다. 기본값은 10입니다.

- `maxSampleFilesPerPartition` - 샘플링이 한 파티션에서 읽을 최대 파일 수입니다. 기본값은 10입니다.

  이러한 파라미터는 파일 나열에 소요되는 시간을 줄이는 데 도움이 됩니다. 예를 들어 데이터 세트에 1,000개의 파티션이 있고 각 파티션에 10개의 파일이 있다고 가정합니다. `maxSamplePartitions` = 10 및 `maxSampleFilesPerPartition` = 10으로 설정하면 10,000개의 파일을 모두 나열하는 대신 샘플링은 각각에 처음 10개의 파일이 있는 처음 10개의 파티션만 나열하고 읽습니다(10*10 = 총 100개의 파일).

- `format` - 형식 사양. 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- `format_options` - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).
- `push_down_predicate` - 데이터 집합 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 자세한 내용은 푸시다운 조건자를 사용하여 예비 필터링 섹션을 참조하세요.

add_ingestion_time_columns

`add_ingestion_time_columns(dataFrame, timeGranularity = "")`

입력 `DataFrame`에 `ingest_year`, `ingest_month`, `ingest_day`, `ingest_hour`, `ingest_minute`와 같은 수집 시간 열을 추가합니다. 이 함수는 Amazon S3을 대상으로 하는 데이터 카탈로그 테이블을 지정할 때 AWS Glue가 생성하는 스크립트에서 자동으로 생성됩니다. 이 함수는 출력 테이블의 수집 시간 열로 파티션을 자동으로 업데이트합니다. 이를 통해 입력 데이터에 명시적인 수집 시간 열을 요구하지 않고도 출력 데이터를 수집 시간에 자동으로 분할할 수 있습니다.

- `dataFrame` - 수집 시간 열을 추가할 `dataFrame`입니다.
- `timeGranularity` - 시간 열의 세분성입니다. 유효 값은 "`day`", "`hour`" 및 "`minute`"입니다. 예를 들어 "`hour`"가 함수에 전달되면 원래 `dataFrame`에 "`ingest_year`", "`ingest_month`", "`ingest_day`" 및 "`ingest_hour`" 시간 열이 추가됩니다.

시간 세분성 열을 추가한 후 데이터 프레임을 반환합니다.

예시:

```
dynamic_frame = DynamicFrame.fromDF(glueContext.add_ingestion_time_columns(dataFrame,
 "hour"))
```

create_data_frame_from_catalog

`create_data_frame_from_catalog(database, table_name, transformation_ctx = "", additional_options = {})`

데이터 카탈로그 테이블의 정보를 사용하여 생성된 DataFrame을 반환합니다.

- `database` - 읽을 데이터 카탈로그 데이터베이스입니다.
- `table_name` - 읽을 데이터 카탈로그 테이블의 이름입니다.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).
- `additional_options` - 선택적 이름-값 페어의 모음입니다. 가능한 옵션에는 스트리밍 소스에 대해 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션에 나열된 옵션(예: `startingPosition`, `maxFetchTimeInMs`, `startingOffsets`)이 있습니다.
  - `useSparkDataSource` - true로 설정하면 AWS Glue는 네이티브 Spark 데이터 소스 API를 사용하여 테이블을 읽도록 강제 적용합니다. Spark 데이터 소스 API는 AVRO, 바이너리, CSV, JSON, ORC, Parquet 및 텍스트 형식을 지원합니다. 데이터 카탈로그 테이블에서는 `classification` 속성을 사용하여 형식을 지정합니다. Spark 데이터 소스 API에 대한 자세한 내용은 공식 Apache Spark 설명서를 참조하세요.

    `create_data_frame_from_catalog`를 `useSparkDataSource`와 함께 사용하면 다음과 같은 이점이 있습니다.

    - DataFrame을 직접 반환하고 `create_dynamic_frame.from_catalog().toDF()`에 대한 대안을 제공합니다.
    - 기본 형식에 대한 AWS Lake Formation 테이블 수준 권한 제어를 지원합니다.
    - AWS Lake Formation 테이블 수준 권한 제어 없이 데이터 레이크 형식 읽기를 지원합니다. 자세한 내용은 AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용 섹션을 참조하세요.

    `useSparkDataSource`를 활성화하면 필요에 따라 `additional_options`에서 Spark 데이터 소스 옵션을 추가할 수도 있습니다. AWS Glue는 이러한 옵션을 Spark 리더에 직접 전달합니다.
  - `useCatalogSchema` - true로 설정하면 AWS Glue는 결과 DataFrame에 데이터 카탈로그 스키마를 적용합니다. 그렇지 않으면 리더는 데이터에서 스키마를 추론합니다. 또한 `useCatalogSchema`를 활성화할 경우 `useSparkDataSource`를 true로 설정해야 합니다.

제한 사항

`useSparkDataSource` 옵션을 사용할 경우 다음과 같은 제한 사항을 고려하세요.

- useSparkDataSource를 사용하면 AWS Glue에서는 원래 Spark 세션과 다른 별도의 Spark 세션에 새 DataFrame을 만듭니다.
- Spark DataFrame 파티션 필터링은 다음 AWS Glue 기능에서 작동하지 않습니다.
  - 작업 북마크
  - Amazon S3 스토리지 클래스 제외
  - 카탈로그 파티션 조건자

  이러한 기능에 파티션 필터링을 사용하려면 AWS Glue 푸시다운 조건자를 사용할 수 있습니다. 자세한 내용은 푸시다운 조건자를 사용하여 예비 필터링 섹션을 참조하세요. 분할되지 않은 열에 대한 필터링에는 영향을 주지 않습니다.

  다음 예제 스크립트는 excludeStorageClasses 옵션을 사용하여 파티션 필터링을 수행하는 잘못된 방법을 보여줍니다.

```
// Incorrect partition filtering using Spark filter with excludeStorageClasses
read_df = glueContext.create_data_frame.from_catalog(
    database=database_name,
    table_name=table_name,
    additional_options = {
      "useSparkDataSource": True,
      "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)

//  Suppose year and month are partition keys.
//  Filtering on year and month won't work, the filtered_df will still
//  contain data with other year/month values.
filtered_df = read_df.filter("year == '2017 and month == '04' and 'state == 'CA'")
```

  다음 예제 스크립트는 excludeStorageClasses 옵션을 사용하여 파티션 필터링을 수행하기 위해 푸시다운 조건자를 사용하는 올바른 방법을 보여줍니다.

```
// Correct partition filtering using the AWS Glue pushdown predicate
// with excludeStorageClasses
read_df = glueContext.create_data_frame.from_catalog(
    database=database_name,
    table_name=table_name,
    //  Use AWS Glue pushdown predicate to perform partition filtering
    push_down_predicate = "(year=='2017' and month=='04')"
    additional_options = {
```

```
      "useSparkDataSource": True,
      "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)

//  Use Spark filter only on non-partitioned columns
filtered_df = read_df.filter("state == 'CA'")
```

예: Spark 데이터 소스 리더를 사용하여 CSV 테이블 생성

```
//  Read a CSV table with '\t' as separator
read_df = glueContext.create_data_frame.from_catalog(
    database=<database_name>,
    table_name=<table_name>,
    additional_options = {"useSparkDataSource": True,  "sep": '\t'}
)
```

create_data_frame_from_options

create_data_frame_from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")

이 API는 이제 더 이상 사용되지 않습니다. 대신에 `getSource()` API를 사용하십시오. 지정된 연결 및 포맷으로 생성된 `DataFrame`을 반환합니다. 이 함수는 AWS Glue 스트리밍 소스에만 사용합니다.

- `connection_type` - 스트리밍 연결 유형입니다. 유효한 값에는 `kinesis` 및 `kafka`(이)가 있습니다.
- `connection_options` - Kinesis 및 Kafka에 대해 서로 다른 연결 옵션입니다. AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션에서 각 스트리밍 데이터 원본에 대한 모든 연결 옵션 목록을 찾을 수 있습니다. 스트리밍 연결 옵션에는 다음과 같은 차이점이 있습니다.
  - Kinesis 스트리밍 소스에는 `streamARN`, `startingPosition`, `inferSchema` 및 `classification`이 필요합니다.
  - Kafka 스트리밍 소스에는 `connectionName`, `topicName`, `startingOffsets`, `inferSchema` 및 `classification`이 필요합니다.
- `format` - 형식 사양. 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷에 관한 내용은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션섹션을 참조하세요.

- `format_options` - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷 옵션에 대한 자세한 내용은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션 섹션을 참조하세요.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).

Amazon Kinesis 스트리밍 소스의 예:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Kafka 스트리밍 소스의 예:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

forEachBatch

`forEachBatch(frame, batch_function, options)`

스트리밍 소스에서 읽는 모든 마이크로 배치에 전달된 `batch_function`을 적용합니다.

- `frame` - 현재 마이크로 배치를 포함하는 DataFrame입니다.
- `batch_function` - 모든 마이크로 배치에 적용될 함수입니다.
- `options` - 마이크로 배치를 처리하는 방법에 대한 정보가 들어 있는 키-값 페어 컬렉션입니다. 다음 옵션이 필요합니다.

- `windowSize` - 각 배치를 처리하는 데 소요되는 시간입니다.
- `checkpointLocation` - 스트리밍 ETL 작업에 대해 체크포인트가 저장되는 위치입니다.
- `batchMaxRetries` – 실패한 경우 배치를 다시 시도할 수 있는 최대 횟수입니다. 기본값은 3입니다. 이 옵션은 Glue 버전 2.0 이상에서만 구성할 수 있습니다.

예시:

```
glueContext.forEachBatch(
    frame = data_frame_datasource0,
    batch_function = processBatch,
    options = {
        "windowSize": "100 seconds",
        "checkpointLocation": "s3://kafka-auth-dataplane/confluent-test/output/
checkpoint/"
    }
)

def processBatch(data_frame, batchId):
    if (data_frame.count() > 0):
        datasource0 = DynamicFrame.fromDF(
          glueContext.add_ingestion_time_columns(data_frame, "hour"),
          glueContext, "from_data_frame"
        )
        additionalOptions_datasink1 = {"enableUpdateCatalog": True}
        additionalOptions_datasink1["partitionKeys"] = ["ingest_yr", "ingest_mo",
 "ingest_day"]
        datasink1 = glueContext.write_dynamic_frame.from_catalog(
          frame = datasource0,
          database = "tempdb",
          table_name = "kafka-auth-table-output",
          transformation_ctx = "datasink1",
          additional_options = additionalOptions_datasink1
        )
```

Amazon S3의 데이터 집합 작업

- purge_table
- purge_s3_path
- transition_table
- transition_s3_path

purge_table

purge_table(catalog_id=None, database="", table_name="", options={}, transformation_ctx="")

지정된 카탈로그의 데이터베이스 및 테이블에 대한 파일을 Amazon S3에서 삭제합니다. 파티션의 모든 파일을 삭제하면 해당 파티션도 카탈로그에서 삭제됩니다. Lake Formation에 등록된 테이블에 대해서는 purge_table 작업을 지원하지 않습니다.

삭제된 객체를 복구하려면 Amazon S3 버킷에서 객체 버전 관리를 설정할 수 있습니다. 객체 버전 관리가 활성화되어 있지 않은 버킷에서 객체를 삭제하는 경우에는 객체를 복구할 수 없습니다. 버전 관리가 사용되는 버킷에서 삭제된 객체를 복구하는 방법에 대한 자세한 내용은 AWS Support 지식 센터에서 삭제된 Amazon S3 객체를 검색하려면 어떻게 해야 합니까?를 참조하세요.

- `catalog_id` - 액세스 중인 데이터 카탈로그의 카탈로그 ID(데이터 카탈로그의 계정 ID)입니다. 기본적으로 None으로 설정됩니다. None인 경우 서비스에 있는 호출 계정의 카탈로그 ID가 기본 설정됩니다.
- `database` - 사용할 데이터베이스입니다.
- `table_name` - 사용할 테이블의 이름입니다.
- `options` - 삭제할 파일 필터링 및 매니페스트 파일 생성을 위한 옵션입니다.
  - `retentionPeriod` - 파일을 보존할 기간(시간)을 지정합니다. 보존 기간 내의 파일은 유지됩니다. 기본적으로 168시간(7일)으로 설정됩니다.
  - `partitionPredicate` - 이 조건자를 충족하는 파티션이 삭제됩니다. 이러한 파티션에서 보존 기간 내에 있는 파일은 삭제되지 않습니다. 기본적으로 ""(비움)로 설정합니다.
  - `excludeStorageClasses` – `excludeStorageClasses` 집합에 스토리지 클래스가 있는 파일은 삭제되지 않습니다. 기본값은 `Set()`(빈 집합)입니다.
  - `manifestFilePath` - 매니페스트 파일 생성을 위한 선택적 경로입니다. 성공적으로 제거된 모든 파일은 `Success.csv`에 기록되고 실패한 파일은 `Failed.csv`에 기록됩니다.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항). 매니페스트 파일 경로에 사용됩니다.

Example

```
glueContext.purge_table("database", "table", {"partitionPredicate": "(month=='march')",
 "retentionPeriod": 1, "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath":
 "s3://bucketmanifest/"})
```

purge_s3_path

purge_s3_path(s3_path, options={}, transformation_ctx="")

지정된 Amazon S3 경로에서 파일을 재귀적으로 삭제합니다.

삭제된 객체를 복구하려면 Amazon S3 버킷에서 객체 버전 관리를 설정할 수 있습니다. 객체 버전 관리가 설정되어 있지 않은 버킷에서 객체를 삭제하는 경우에는 객체를 복구할 수 없습니다. 버전 관리를 사용하여 버킷에서 삭제된 객체를 복구하는 방법에 대한 자세한 내용은 지원 지식 센터에서 삭제된 Amazon S3 객체를 검색하려면 어떻게 해야 합니까?를 참조하세요.

- s3_path - s3://<*bucket*>/<*prefix*>/ 포맷으로 삭제할 파일에 대한 Amazon S3의 경로입니다.
- options - 삭제할 파일 필터링 및 매니페스트 파일 생성을 위한 옵션입니다.
  - retentionPeriod - 파일을 보존할 기간(시간)을 지정합니다. 보존 기간 내의 파일은 유지됩니다. 기본적으로 168시간(7일)으로 설정됩니다.
  - excludeStorageClasses – excludeStorageClasses 집합에 스토리지 클래스가 있는 파일은 삭제되지 않습니다. 기본값은 Set()(빈 집합)입니다.
  - manifestFilePath - 매니페스트 파일 생성을 위한 선택적 경로입니다. 성공적으로 제거된 모든 파일은 Success.csv에 기록되고 실패한 파일은 Failed.csv에 기록됩니다.
- transformation_ctx - 사용할 변환 내용입니다(선택 사항). 매니페스트 파일 경로에 사용됩니다.

Example

```
glueContext.purge_s3_path("s3://bucket/path/", {"retentionPeriod": 1,
 "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath": "s3://bucketmanifest/"})
```

transition_table

transition_table(database, table_name, transition_to, options={}, transformation_ctx="", catalog_id=None)

지정된 카탈로그의 데이터베이스 및 테이블에 대해 Amazon S3에 저장된 파일의 스토리지 클래스를 전환합니다.

두 스토리지 클래스 간에 전환할 수 있습니다. GLACIER 및 DEEP_ARCHIVE 스토리지 클래스의 경우 이러한 클래스로 전환할 수 있습니다. 하지만 S3 RESTORE를 사용하여 GLACIER 및 DEEP_ARCHIVE 스토리지 클래스에서 전환할 수 있습니다.

Amazon S3에서 파일 또는 파티션을 읽는 AWS Glue ETL 작업을 실행하는 경우 일부 Amazon S3 스토리지 클래스 유형을 제외할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 클래스 제외를 참조하세요.

- `database` - 사용할 데이터베이스입니다.
- `table_name` - 사용할 테이블의 이름입니다.
- `transition_to` - 전환할 Amazon S3 스토리지 클래스입니다.
- `options` - 삭제할 파일 필터링 및 매니페스트 파일 생성을 위한 옵션입니다.
  - `retentionPeriod` - 파일을 보존할 기간(시간)을 지정합니다. 보존 기간 내의 파일은 유지됩니다. 기본적으로 168시간(7일)으로 설정됩니다.
  - `partitionPredicate` - 이 조건자를 충족하는 파티션이 전환됩니다. 이러한 파티션에서 보존 기간 내에 있는 파일은 전환되지 않습니다. 기본적으로 ""(비움)로 설정합니다.
  - `excludeStorageClasses` – `excludeStorageClasses` 집합에 스토리지 클래스가 있는 파일은 전환되지 않습니다. 기본값은 `Set()`(빈 집합)입니다.
  - `manifestFilePath` - 매니페스트 파일 생성을 위한 선택적 경로입니다. 성공적으로 전환된 모든 파일은 `Success.csv`에 기록되고 실패한 파일은 `Failed.csv`에 기록됩니다.
  - `accountId` - 전환 변환을 실행할 Amazon Web Services 계정 ID입니다. 이 변환에 대해 필수입니다.
  - `roleArn` - 전환/변환을 실행할 AWS 역할입니다. 이 변환에 대해 필수입니다.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항). 매니페스트 파일 경로에 사용됩니다.
- `catalog_id` - 액세스 중인 데이터 카탈로그의 카탈로그 ID(데이터 카탈로그의 계정 ID)입니다. 기본적으로 None으로 설정됩니다. None인 경우 서비스에 있는 호출 계정의 카탈로그 ID가 기본 설정됩니다.

Example

```
glueContext.transition_table("database", "table", "STANDARD_IA", {"retentionPeriod": 1,
 "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath": "s3://bucketmanifest/",
 "accountId": "12345678901", "roleArn": "arn:aws:iam::123456789012:user/example-
username"})
```

transition_s3_path

`transition_s3_path(s3_path, transition_to, options={}, transformation_ctx="")`

지정된 Amazon S3 경로에 있는 파일의 스토리지 클래스를 재귀적으로 전환합니다.

두 스토리지 클래스 간에 전환할 수 있습니다. GLACIER 및 DEEP_ARCHIVE 스토리지 클래스의 경우 이러한 클래스로 전환할 수 있습니다. 하지만 S3 RESTORE를 사용하여 GLACIER 및 DEEP_ARCHIVE 스토리지 클래스에서 전환할 수 있습니다.

Amazon S3에서 파일 또는 파티션을 읽는 AWS Glue ETL 작업을 실행하는 경우 일부 Amazon S3 스토리지 클래스 유형을 제외할 수 있습니다. 자세한 내용은 Amazon S3 스토리지 클래스 제외를 참조하세요.

- `s3_path` - `s3://<bucket>/<prefix>/` 포맷으로 전환할 파일에 대한 Amazon S3의 경로입니다.
- `transition_to` - 전환할 Amazon S3 스토리지 클래스입니다.
- `options` - 삭제할 파일 필터링 및 매니페스트 파일 생성을 위한 옵션입니다.
  - `retentionPeriod` - 파일을 보존할 기간(시간)을 지정합니다. 보존 기간 내의 파일은 유지됩니다. 기본적으로 168시간(7일)으로 설정됩니다.
  - `partitionPredicate` - 이 조건자를 충족하는 파티션이 전환됩니다. 이러한 파티션에서 보존 기간 내에 있는 파일은 전환되지 않습니다. 기본적으로 `""`(비움)로 설정합니다.
  - `excludeStorageClasses` – `excludeStorageClasses` 집합에 스토리지 클래스가 있는 파일은 전환되지 않습니다. 기본값은 `Set()`(빈 집합)입니다.
  - `manifestFilePath` - 매니페스트 파일 생성을 위한 선택적 경로입니다. 성공적으로 전환된 모든 파일은 `Success.csv`에 기록되고 실패한 파일은 `Failed.csv`에 기록됩니다.
  - `accountId` - 전환 변환을 실행할 Amazon Web Services 계정 ID입니다. 이 변환에 대해 필수입니다.
  - `roleArn` - 전환/변환을 실행할 AWS 역할입니다. 이 변환에 대해 필수입니다.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항). 매니페스트 파일 경로에 사용됩니다.

Example

```
glueContext.transition_s3_path("s3://bucket/prefix/", "STANDARD_IA",
 {"retentionPeriod": 1, "excludeStorageClasses": ["STANDARD_IA"],
 "manifestFilePath": "s3://bucketmanifest/", "accountId": "12345678901", "roleArn":
 "arn:aws:iam::123456789012:user/example-username"})
```

추출

- extract_jdbc_conf

extract_jdbc_conf

`extract_jdbc_conf(connection_name, catalog_id = None)`

데이터 카탈로그에 있는 AWS Glue 연결 개체의 구성 속성이 있는 `dict`와 키를 반환합니다.

- `user` – 데이터베이스 사용자 이름입니다.
- `password` – 데이터베이스 암호입니다.
- `vendor` – 공급업체를 지정합니다 (`mysql`, `postgresql`, `oracle`, `sqlserver` 등.).
- `enforceSSL` – 보안 연결이 필요한지 여부를 나타내는 부울 문자열입니다.
- `customJDBCCert` – 표시된 Amazon S3 경로의 특정 클라이언트 인증서를 사용합니다.
- `skipCustomJDBCCertValidation` – CA에서 `customJDBCCert`의 유효성을 검사해야 하는지 여부를 나타내는 부울 문자열입니다.
- `customJDBCCertString` – 드라이버 유형에 맞는 사용자 지정 인증서에 대한 추가 정보입니다.
- `url` – (더 이상 사용되지 않음) 프로토콜, 서버 및 포트만 있는 JDBC URL입니다.
- `fullUrl` - 연결이 생성될 때 입력한 JDBC URL입니다(AWS Glue 버전 3.0 이상에서 사용 가능).

JDBC 구성 검색의 예:

```
jdbc_conf = glueContext.extract_jdbc_conf(connection_name="your_glue_connection_name")
print(jdbc_conf)
>>> {'enforceSSL': 'false', 'skipCustomJDBCCertValidation': 'false', 'url':
 'jdbc:mysql://myserver:3306', 'fullUrl': 'jdbc:mysql://myserver:3306/mydb',
 'customJDBCCertString': '', 'user': 'admin', 'customJDBCCert': '', 'password': '1234',
 'vendor': 'mysql'}
```

트랜잭션

- start_transaction
- commit_transaction
- cancel_transaction

start_transaction

`start_transaction(read_only)`

새 트랜잭션을 시작합니다. 내부적으로 Lake Formation startTransaction API를 호출합니다.

- `read_only` - (부울) 이 트랜잭션이 읽기 전용인지 또는 읽기/쓰기인지를 나타냅니다. 읽기 전용 트랜잭션 ID를 사용하여 수행된 쓰기는 거부됩니다. 읽기 전용 트랜잭션은 커밋할 필요가 없습니다.

트랜잭션 ID를 반환합니다.

commit_transaction

`commit_transaction(transaction_id, wait_for_commit = True)`

지정된 트랜잭션을 커밋하려는 시도입니다. 트랜잭션이 커밋을 완료하기 전에 `commit_transaction`이 반환될 수 있습니다. 내부적으로 Lake Formation commitTransaction API를 호출합니다.

- `transaction_id` - (문자열) 커밋할 트랜잭션입니다.
- `wait_for_commit` - (부울) `commit_transaction`이 즉시 반환되는지 여부를 결정합니다. 기본값은 true입니다. false인 경우 `commit_transaction`은 폴링한 후 트랜잭션이 커밋될 때까지 기다립니다. 대기 시간은 최대 재시도 횟수가 6회인 지수 백오프를 사용하여 1분으로 제한됩니다.

커밋이 수행되었는지 여부를 나타내는 부울을 반환합니다.

cancel_transaction

`cancel_transaction(transaction_id)`

지정된 트랜잭션을 취소하려는 시도입니다. 트랜잭션이 이전에 커밋된 경우 `TransactionCommittedException` 예외를 반환합니다. 내부적으로 Lake Formation CancelTransaction API를 호출합니다.

- `transaction_id` - (문자열) 취소할 트랜잭션입니다.

작성

- getSink

- write_dynamic_frame_from_options
- write_from_options
- write_dynamic_frame_from_catalog
- write_data_frame_from_catalog
- write_dynamic_frame_from_jdbc_conf
- write_from_jdbc_conf

getSink

`getSink(connection_type, format = None, transformation_ctx = "", **options)`

외부 소스로 `DataSink`를 읽을 때 사용되는 `DynamicFrames` 객체를 얻습니다. 먼저 SparkSQL `format`를 확인하여 예상 싱크를 얻도록 합니다.

- `connection_type` - Amazon S3, Amazon Redshift 및 JDBC와 같이 사용할 연결 유형입니다. 유효한 값에는 `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `kinesis`, `kafka`가 있습니다.
- `format` - 사용할 SparkSQL 포맷입니다(선택 사항).
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).
- `options` - 연결 옵션을 지정하는 데 사용되는 이름-값 페어의 컬렉션입니다. 몇 가지 가능한 값은 다음과 같습니다.
  - `user` 및 `password`: 권한 부여용
  - `url`: 데이터 스토어에 대한 엔드포인트
  - `dbtable`: 대상 테이블의 이름
  - `bulkSize`: 삽입 작업의 병렬 처리 수준

지정할 수 있는 옵션은 연결 유형에 따라 다릅니다. 추가 값과 예제는 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.

예시:

```
>>> data_sink = context.getSink("s3")
>>> data_sink.setFormat("json"),
>>> data_sink.writeFrame(myFrame)
```

write_dynamic_frame_from_options

write_dynamic_frame_from_options(frame, connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")

지정한 연결 및 포맷을 사용하여 DynamicFrame를 작성하고 반환합니다.

- frame - 작성할 DynamicFrame입니다.
- connection_type - Amazon S3, Amazon Redshift 및 JDBC 등의 연결 유형입니다. 유효한 값에는 s3, mysql, postgresql, redshift, sqlserver, oracle, kinesis, kafka가 있습니다.
- connection_options - 경로 및 데이터베이스 테이블과 같은 연결 옵션입니다(선택 사항). s3의 connection_type의 경우, Amazon S3 경로가 정의됩니다.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

JDBC 연결의 경우, 몇 까지 속성이 정의되어야 합니다. 단, 데이터베이스 이름이 URL의 일부여야 합니다. 연결 옵션에 선택적으로 포함될 수 있습니다.

> **Warning**
>
> 스크립트에 암호를 저장하는 것은 권장되지 않습니다. AWS Secrets Manager 또는 AWS Glue 데이터 카탈로그에서 데이터를 검색할 때 boto3 사용을 고려합니다.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

dbtable 속성은 JDBC 테이블의 이름입니다. 데이터베이스 내의 스키마를 지원하는 JDBC 데이터 스토어의 경우 schema.table-name에 대해 지정합니다. 스키마가 제공되지 않으면 기본 "퍼블릭" 스키마가 사용됩니다.

자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.

- format - 형식 사양. 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.

- `format_options` - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).

write_from_options

`write_from_options(frame_or_dfc, connection_type, connection_options={}, format={}, format_options={}, transformation_ctx = "")`

지정한 연결 및 포맷 정보로 생성된 DynamicFrame 또는 DynamicFrameCollection을 작성하고 반환합니다.

- `frame_or_dfc` – 작성할 DynamicFrame 또는 DynamicFrameCollection입니다.
- `connection_type` - Amazon S3, Amazon Redshift 및 JDBC 등의 연결 유형입니다. 유효한 값에는 `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` 및 `oracle`가 있습니다.
- `connection_options` - 경로 및 데이터베이스 테이블과 같은 연결 옵션입니다(선택 사항). `s3`의 `connection_type`의 경우, Amazon S3 경로가 정의됩니다.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

JDBC 연결의 경우, 몇 까지 속성이 정의되어야 합니다. 단, 데이터베이스 이름이 URL의 일부여야 합니다. 연결 옵션에 선택적으로 포함될 수 있습니다.

> **Warning**
>
> 스크립트에 암호를 저장하는 것은 권장되지 않습니다. AWS Secrets Manager 또는 AWS Glue 데이터 카탈로그에서 데이터를 검색할 때 boto3 사용을 고려합니다.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

`dbtable` 속성은 JDBC 테이블의 이름입니다. 데이터베이스 내의 스키마를 지원하는 JDBC 데이터 스토어의 경우 `schema.table-name`에 대해 지정합니다. 스키마가 제공되지 않으면 기본 "퍼블릭" 스키마가 사용됩니다.

자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.

- `format` - 형식 사양. 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- `format_options` - 지정된 포맷에 대한 포맷 옵션입니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하십시오.
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).

write_dynamic_frame_from_catalog

`write_dynamic_frame_from_catalog(frame, database, table_name, redshift_tmp_dir, transformation_ctx = "", additional_options = {}, catalog_id = None)`

데이터 카탈로그 데이터베이스 및 테이블의 정보를 사용하여 `DynamicFrame`을 작성하고 반환합니다.

- `frame` - 작성할 `DynamicFrame`입니다.
- `Database` - 테이블이 들어 있는 데이터 카탈로그 데이터베이스입니다.
- `table_name` - 대상과 연결된 데이터 카탈로그 테이블의 이름입니다.
- `redshift_tmp_dir` – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).
- `additional_options` - 선택적 이름-값 페어의 모음입니다.
- `catalog_id` - 액세스 중인 데이터 카탈로그의 카탈로그 ID(계정 ID)입니다. None인 경우 호출자의 기본 계정 ID가 사용됩니다.

write_data_frame_from_catalog

`write_data_frame_from_catalog(frame, database, table_name, redshift_tmp_dir, transformation_ctx = "", additional_options = {}, catalog_id = None)`

데이터 카탈로그 데이터베이스 및 테이블의 정보를 사용하여 `DataFrame`을 작성하고 반환합니다. 이 메서드는 데이터 레이크 형식(Hudi, Iceberg, Delta Lake) 쓰기를 지원합니다. 자세한 내용은 AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용 섹션을 참조하세요.

- `frame` - 작성할 `DataFrame`입니다.

- `Database` - 테이블이 들어 있는 데이터 카탈로그 데이터베이스입니다.
- `table_name` - 대상과 연결된 데이터 카탈로그 테이블의 이름입니다.
- `redshift_tmp_dir` – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).
- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).
- `additional_options` - 선택적 이름-값 페어의 모음입니다.
  - `useSparkDataSink` - true로 설정하면 AWS Glue가 네이티브 Spark 데이터 싱크 API를 사용하여 테이블에 쓰도록 강제 적용합니다. 이 옵션을 활성화하면 필요에 따라 Spark 데이터 소스 옵션을 `additional_options`에 추가할 수 있습니다. AWS Glue는 이러한 옵션을 Spark 라이터에 직접 전달합니다.
- `catalog_id` - 액세스 중인 데이터 카탈로그의 카탈로그 ID(계정 ID)입니다. 값을 지정하지 않으면 발신자의 기본 계정 ID가 사용됩니다.

제한 사항

`useSparkDataSink` 옵션을 사용할 경우 다음과 같은 제한 사항을 고려하세요.

- `useSparkDataSink` 옵션을 사용하는 경우 `enableUpdateCatalog` 옵션이 지원되지 않습니다.

예: Spark 데이터 소스 라이터를 사용하여 Hudi 테이블에 쓰기

```
hudi_options = {
    'useSparkDataSink': True,
    'hoodie.table.name': <table_name>,
    'hoodie.datasource.write.storage.type': 'COPY_ON_WRITE',
    'hoodie.datasource.write.recordkey.field': 'product_id',
    'hoodie.datasource.write.table.name': <table_name>,
    'hoodie.datasource.write.operation': 'upsert',
    'hoodie.datasource.write.precombine.field': 'updated_at',
    'hoodie.datasource.write.hive_style_partitioning': 'true',
    'hoodie.upsert.shuffle.parallelism': 2,
    'hoodie.insert.shuffle.parallelism': 2,
    'hoodie.datasource.hive_sync.enable': 'true',
    'hoodie.datasource.hive_sync.database': <database_name>,
    'hoodie.datasource.hive_sync.table': <table_name>,
    'hoodie.datasource.hive_sync.use_jdbc': 'false',
    'hoodie.datasource.hive_sync.mode': 'hms'}

glueContext.write_data_frame.from_catalog(
```

```
    frame = <df_product_inserts>,
    database = <database_name>,
    table_name = <table_name>,
    additional_options = hudi_options
)
```

write_dynamic_frame_from_jdbc_conf

write_dynamic_frame_from_jdbc_conf(frame, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx = "", catalog_id = None)

지정한 JDBC 연결 정보를 사용하여 DynamicFrame를 작성하고 반환합니다.

- frame - 작성할 DynamicFrame입니다.
- catalog_connection - 사용할 카탈로그 연결입니다.
- connection_options - 경로 및 데이터베이스 테이블과 같은 연결 옵션입니다(선택 사항). 자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.
- redshift_tmp_dir – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).
- transformation_ctx - 사용할 변환 내용입니다(선택 사항).
- catalog_id - 액세스 중인 데이터 카탈로그의 카탈로그 ID(계정 ID)입니다. None인 경우 호출자의 기본 계정 ID가 사용됩니다.

write_from_jdbc_conf

write_from_jdbc_conf(frame_or_dfc, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx = "", catalog_id = None)

지정한 JDBC 연결 정보를 사용하여 DynamicFrame 또는 DynamicFrameCollection을 작성하고 반환합니다.

- frame_or_dfc – 작성할 DynamicFrame 또는 DynamicFrameCollection입니다.
- catalog_connection - 사용할 카탈로그 연결입니다.
- connection_options - 경로 및 데이터베이스 테이블과 같은 연결 옵션입니다(선택 사항). 자세한 내용은 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션 섹션을 참조하세요.
- redshift_tmp_dir – 사용할 Amazon Redshift 임시 디렉터리입니다(선택 사항).

- `transformation_ctx` - 사용할 변환 내용입니다(선택 사항).
- `catalog_id` - 액세스 중인 데이터 카탈로그의 카탈로그 ID(계정 ID)입니다. None인 경우 호출자의 기본 계정 ID가 사용됩니다.

AWS Glue PySpark 변환 참조

AWS Glue는 PySpark ETL 작동에서 사용할 수 있는 다음과 같은 내장형 변환을 제공합니다. 데이터가 DynamicFrame이라는 데이터 구조 내의 변환에서 변환으로 전달됩니다. 이는 Apache Spark SQL `DataFrame`을 확장한 것입니다. `DynamicFrame`은 데이터를 포함하고 데이터 스키마를 참조하여 데이터를 진행합니다.

이러한 변환의 대부분은 `DynamicFrame` 클래스의 메서드로도 존재합니다. 자세한 내용은 DynamicFrame 변환을 참조하세요.

- GlueTransform 베이스 클래스
- ApplyMapping 클래스
- DropFields 클래스
- DropNullField 클래스
- ErrorsAsDynamicFrame 클래스
- EvaluateDataQuality 클래스
- FillMissingValues 클래스
- Filter 클래스
- FindIncrementalMatches 클래스
- FindMatches 클래스
- FlatMap 클래스
- Join 클래스
- Map 클래스
- MapToCollection 클래스
- mergeDynamicFrame
- Relationalize 클래스
- RenameField 클래스
- ResolveChoice 클래스

- SelectFields 클래스
- SelectFromCollection 클래스
- Simplify_ddb_json 클래스
- Spigot 클래스
- SplitFields 클래스
- SplitRows 클래스
- Unbox 클래스
- UnnestFrame 클래스

GlueTransform 베이스 클래스

이 베이스 클래스에서 모든 `awsglue.transforms` 클래스가 물려받습니다.

이 클래스는 `__call__` 방법을 정의합니다. 이것들은 다음 섹션에 있는 `GlueTransform` 클래스 방법을 재정의하거나 기본값으로 클래스 이름을 사용하여 호출됩니다.

메서드

- apply(cls, *args, **kwargs)
- name(cls)
- describeArgs(cls)
- describeReturn(cls)
- describeTransform(cls)
- describeErrors(cls)
- describe(cls)

apply(cls, *args, **kwargs)

변환 클래스를 호출하여 변환을 적용하고 결과를 반환합니다.

- `cls` – `self` 클래스 객체입니다.

name(cls)

파생된 변환 클래스의 이름을 반환합니다.

- `cls` – `self` 클래스 객체입니다.

describeArgs(cls)

- `cls` – `self` 클래스 객체입니다.

명명된 논제와 관련된 딕셔너리 목록을 다음과 같은 포맷으로 반환합니다.

```
[
  {
    "name": "(name of argument)",
    "type": "(type of argument)",
    "description": "(description of argument)",
    "optional": "(Boolean, True if the argument is optional)",
    "defaultValue": "(Default value string, or None)(String; the default value, or
 None)"
  },
...
]
```

파생된 변환이 실행되지 않고 호출되지 않으면 `NotImplementedError` 예외가 발생합니다.

describeReturn(cls)

- `cls` – `self` 클래스 객체입니다.

반환 유형에 대한 정보와 함께 다음 포맷으로 딕셔너리를 반환합니다.

```
{
  "type": "(return type)",
  "description": "(description of output)"
}
```

파생된 변환이 실행되지 않고 호출되지 않으면 `NotImplementedError` 예외가 발생합니다.

describeTransform(cls)

변환을 나타내는 문자열을 반환합니다.

- `cls` – `self` 클래스 객체입니다.

파생된 변환이 실행되지 않고 호출되지 않으면 `NotImplementedError` 예외가 발생합니다.

describeErrors(cls)

- `cls` – `self` 클래스 객체입니다.

변환에 따른 예외를 나타내는 딕셔너리 목록을 다음과 같은 포맷으로 반환합니다.

```
[
  {
    "type": "(type of error)",
    "description": "(description of error)"
  },
...
]
```

describe(cls)

- `cls` – `self` 클래스 객체입니다.

다음의 형식으로 객체를 반환합니다.

```
{
  "transform" : {
    "name" : cls.name( ),
    "args" : cls.describeArgs( ),
    "returns" : cls.describeReturn( ),
    "raises" : cls.describeErrors( ),
    "location" : "internal"
  }
}
```

ApplyMapping 클래스

`DynamicFrame`에서 매핑을 적용합니다.

예

`DynamicFrame`에 매핑을 적용하기 위해서는 `DynamicFrame.apply_mapping()` 메서드를 사용하는 것을 권장합니다. 코드에 대한 예제는 예: apply_maping을 사용하여 필드 이름을 바꾸고 필드 유형을 변경합니다. 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(frame, mappings, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

지정 DynamicFrame에 서술식 매핑을 적용합니다.

- frame – DynamicFrame은 매핑을 적용할 대상(필수).
- mappings - 매핑 튜플 목록(필수). 각각 (소스 열, 소스 유형, 대상 열, 대상 유형)으로 구성된 매핑 튜플 목록입니다.

  소스 열에 점(“.”)이 있는 경우 주위에 백틱(“``”)을 넣어야 합니다. 예를 들어 this.old.name(문자열)을 thisNewName에 매핑하려면 다음 튜플을 사용합니다.

  ```
  ("`this.old.name`", "string", "thisNewName", "string")
  ```
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

“매핑” 튜플에 지정된 DynamicFrame의 필드만 반환합니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

DropFields 클래스

`DynamicFrame` 내에서 필드를 드롭합니다.

예

`DynamicFrame.drop_fields()`에서 필드를 드롭하기 위해 `DynamicFrame` 방법을 사용하는 것이 좋습니다. 코드에 대한 예제는 예: drop_fields를 사용하여 `DynamicFrame`에서 필드를 제거합니다. 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn

- describeTransform
- describeErrors
- 설명

__call__(frame, paths, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

DynamicFrame 내에서 노드를 드롭합니다.

- frame – DynamicFrame은 노드를 드롭합니다(필수).
- paths - 드롭할 노드의 전체 경로 목록입니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

지정된 필드없이 새로운 DynamicFrame을 반환합니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.

describeTransform(cls)

GlueTransform describeTransform에서 상속됩니다.

describeErrors(cls)

GlueTransform describeErrors에서 상속됩니다.

describe(cls)

GlueTransform describe에서 상속됩니다.

DropNullField 클래스

DynamicFrame 에서 NullType 인 유형의 모든 널 필드를 드롭합니다. DynamicFrame 데이터 세트의 모든 기록에서 누락된 혹은 null 값인 필드입니다.

예

이 예제에서는 NullType 유형의 필드가 삭제된 새 DynamicFrame을 만드는 데 DropNullFields를 사용합니다. DropNullFields를 보여 주기 위해 이미 로드된 persons 데이터 세트에 null 유형의 empty_column라는 이름의 새 열을 추가합니다.

> **Note**
>
> 이 예제에서 사용되는 데이터 세트에 액세스하려면 코드 예: 데이터 조인 및 관계화 항목을 참조하고 1단계: Amazon S3 버킷에서 데이터 크롤의 지침을 따르세요.

```
# Example: Use DropNullFields to create a new DynamicFrame without NullType fields

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from pyspark.sql.functions import lit
from pyspark.sql.types import NullType
from awsglue.dynamicframe import DynamicFrame
from awsglue.transforms import DropNullFields

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create DynamicFrame
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
```

```
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Add new column "empty_column" with NullType
persons_with_nulls = persons.toDF().withColumn("empty_column",
 lit(None).cast(NullType()))
persons_with_nulls_dyf = DynamicFrame.fromDF(persons_with_nulls, glueContext,
 "persons_with_nulls")
print("Schema for the persons_with_nulls_dyf DynamicFrame:")
persons_with_nulls_dyf.printSchema()

# Remove the NullType field
persons_no_nulls = DropNullFields.apply(persons_with_nulls_dyf)
print("Schema for the persons_no_nulls DynamicFrame:")
persons_no_nulls.printSchema()
```

출력

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
```

```
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the persons_with_nulls_dyf DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
|-- empty_column: null

null_fields ['empty_column']
Schema for the persons_no_nulls DynamicFrame:
root
|-- family_name: string
```

```
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(frame, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

DynamicFrame 에서 NullType 인 유형의 모든 널 필드를 드롭합니다. DynamicFrame 데이터 세트의 모든 기록에서 누락된 혹은 null 값인 필드입니다.

- frame – DynamicFrame은 null 필드를 드롭합니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

널값이 없는 새로운 DynamicFrame을 반환합니다.

apply(cls, *args, **kwargs)

- cls – cls

name(cls)

- cls – cls

describeArgs(cls)

- cls – cls

describeReturn(cls)

- cls – cls

describeTransform(cls)

- cls – cls

describeErrors(cls)

- `cls` – cls

describe(cls)

- `cls` – cls

ErrorsAsDynamicFrame 클래스

소스 `DynamicFrame`이 생성되는 동안 발생한 오류에 대한 중첩 레코드가 포함된 `DynamicFrame`을 반환합니다.

예

오류 레코드를 조회하고 보는 데는 `DynamicFrame.errorsAsDynamicFrame()` 메서드를 권장합니다. 코드에 대한 예제는 예: errorsAsDynamicFrame을 사용하여 오류 레코드 보기 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(frame)

`DynamicFrame` 소스 생성과 관련된 중첩된 오류 기록을 포함하는 `DynamicFrame`을 반환합니다.

- `frame` – 소스 `DynamicFrame`입니다(필수).

apply(cls, *args, **kwargs)

- `cls` – cls

name(cls)

- `cls` – cls

describeArgs(cls)

- `cls` – cls

describeReturn(cls)

- `cls` – cls

describeTransform(cls)

- `cls` – cls

describeErrors(cls)

- `cls` – cls

describe(cls)

- `cls` – cls

EvaluateDataQuality 클래스

`DynamicFrame`을 기준으로 데이터 품질 규칙 세트를 평가하고 새 `DynamicFrame`을 평가 결과와 함께 반환합니다.

예

다음 예제 코드는 `DynamicFrame`의 데이터 품질을 평가한 다음 데이터 품질 결과를 확인하는 방법을 보여줍니다.

```
from awsglue.transforms import *
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsgluedq.transforms import EvaluateDataQuality
```

```
#Create Glue context
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Define DynamicFrame
legislatorsAreas = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="areas_json")

# Create data quality ruleset
ruleset = """Rules = [ColumnExists "id", IsComplete "id"]"""

# Evaluate data quality
dqResults = EvaluateDataQuality.apply(
    frame=legislatorsAreas,
    ruleset=ruleset,
    publishing_options={
        "dataQualityEvaluationContext": "legislatorsAreas",
        "enableDataQualityCloudWatchMetrics": True,
        "enableDataQualityResultsPublishing": True,
        "resultsS3Prefix": "amzn-s3-demo-bucket1",
    },
)


# Inspect data quality results
dqResults.printSchema()
dqResults.toDF().show()
```

출력

```
root
|-- Rule: string
|-- Outcome: string
|-- FailureReason: string
|-- EvaluatedMetrics: map
|    |-- keyType: string
|    |-- valueType: double


+----------------------+-------+-------------+---------------------------------------+
|Rule                  |Outcome|FailureReason|EvaluatedMetrics                       |
+----------------------+-------+-------------+---------------------------------------+
|ColumnExists "id"     |Passed |null         |{}                                     |
```

```
|IsComplete "id"        |Passed |null         |{Column.first_name.Completeness -> 1.0}|
+-----------------------+-------+-------------+---------------------------------------+
```

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, ruleset, publishing_options = {})

- `frame` - 데이터 품질을 평가하려는 `DynamicFrame`입니다.
- `ruleset` - 문자열 형식의 DQDL(데이터 품질 정의 언어) 규칙 세트입니다. DQDL에 대한 자세한 내용은 데이터 품질 정의 언어(DQDL) 참조 안내서를 참조하세요.
- `publishing_options` - 평가 결과 및 지표를 게시하기 위해 다음 옵션을 지정하는 사전입니다.
  - `dataQualityEvaluationContext` - AWS Glue가 Amazon CloudWatch 지표와 데이터 품질 결과를 게시할 네임스페이스를 지정하는 문자열입니다. 집계된 지표는 CloudWatch에 표시되고 전체 결과는 AWS Glue Studio 인터페이스에 표시됩니다.
    - 필수 항목 여부: 아니요
    - 기본 값: `default_context`
  - `enableDataQualityCloudWatchMetrics` - 데이터 품질 평가 결과를 CloudWatch에 게시할지 여부를 지정합니다. `dataQualityEvaluationContext` 옵션을 사용하여 지표의 네임스페이스를 지정합니다.
    - 필수 항목 여부: 아니요
    - 기본값: False
  - `enableDataQualityResultsPublishing` - AWS Glue Studio 인터페이스의 Data Quality(데이터 품질) 탭에 데이터 품질 결과를 표시할지 여부를 지정합니다.
    - 필수 항목 여부: 아니요

  - 기본값: True
- `resultsS3Prefix` - AWS Glue가 데이터 품질 평가 결과를 기록할 수 있는 Amazon S3 위치를 지정합니다.
  - 필수 항목 여부: 아니요
  - 기본값: ""(빈 문자열)

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

FillMissingValues 클래스

`FillMissingValues` 클래스는 지정된 `DynamicFrame`에서 null 값과 빈 문자열을 찾고 선형 회귀 및 랜덤 포레스트와 같은 기계 학습 방법을 사용하여 누락된 값을 예측합니다. ETL 작업은 입력 데이터 집합의 값을 사용하여 기계 학습 모델을 훈련합니다. 이 모델은 누락 된 값이 무엇인지 예측합니다.

> **Tip**
>
> 증분 데이터 집합을 사용하는 경우 각 증분 집합은 기계 학습 모델의 훈련 데이터로 사용되므로 결과가 정확하지 않을 수 있습니다.

가져오려면

```
from awsglueml.transforms import FillMissingValues
```

메서드

- Apply

apply(frame, missing_values_column, output_column ="", transformation_ctx ="", info ="", stageThreshold = 0, totalThreshold = 0)

지정된 열에 동적 프레임의 누락된 값을 채우고 새 열에 추정 값과 함께 새 프레임을 반환합니다. 누락된 값이 없는 행의 경우 지정된 열의 값이 새 열에 복제됩니다.

- `frame` – 누락된 값을 채울 `DynamicFrame`입니다 필수 사항입니다.
- `missing_values_column` - 누락된 값(`null` 값 및 빈 문자열)이 포함된 열입니다. 필수 사항입니다.
- `output_column` - 값이 누락된 모든 행에 대한 예상 값을 포함할 새 열의 이름입니다. 선택 사항이며 기본값은 `"_filled"`가 접미사로 사용된 `missing_values_column`의 이름입니다.
- `transformation_ctx` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- `info` - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- `stageThreshold` - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항, 기본값은 0).
- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항, 기본값은 0).

누락된 값이 있는 행에 대한 추정 값과 다른 행에 대한 현재 값이 포함된 하나의 추가 열이 있는 새 `DynamicFrame`을 반환합니다.

Filter 클래스

지정된 술어 함수를 충족하는 DynamicFrame 입력의 레코드를 포함하는 새 DynamicFrame 항목을 빌드합니다.

예

DynamicFrame에서 레코드를 필터링하기 위해 DynamicFrame.filter() 방법을 사용하는 것이 좋습니다. 코드에 대한 예제는 예: 필터를 사용하여 필터링된 필드 선택 가져오기 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0))

지정된 조건자 함수를 만족하는 입력 DynamicFrame에서 기록을 선택하여 만든 새로운 DynamicFrame을 반환합니다.

- frame – 소스 DynamicFrame으로 지정된 필터 함수를 적용합니다(필수).
- f - DynamicFrame의 각 DynamicRecord를 적용하는 조건자 함수입니다. 함수는 DynamicRecord를 인수로 받아들이고 DynamicRecord가 필터 요구 사항과 맞으면 True을 반환하고 아니면 False를 반환합니다 (필수).

  DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 스파크 DataFrame의 열과 비슷합니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.

- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

FindIncrementalMatches 클래스

기존 및 증분 `DynamicFrame`에서 일치하는 레코드를 식별하고 일치하는 레코드의 각 그룹에 할당된 고유 식별자로 새 `DynamicFrame`을 생성합니다.

가져오려면

```
from awsglueml.transforms import FindIncrementalMatches
```

메서드

- Apply

apply(existingFrame, incrementalFrame, transformId, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0, enforcedMatches = none, computeMatchConfidenceScores = 0)

입력 DynamicFrame에서 일치하는 레코드를 식별하고 일치하는 레코드의 각 그룹에 할당된 고유 식별자를 사용하여 새 DynamicFrame을 생성합니다.

- existingFrame - FindIncrementalMatches 변환을 적용하기 위한 기존 및 사전 일치 DynamicFrame입니다. 필수 사항입니다.
- incrementalFrame- existingFrame과 일치하도록 FindIncrementalMatches 변환을 적용하는 증분 DynamicFrame입니다. 필수 사항입니다.
- transformId - DynamicFrames의 레코드에 적용할 FindIncrementalMatches 변환과 연결된 고유 ID입니다. 필수 사항입니다.
- transformation_ctx – 고유 문자열을 통해 통계/상태 정보를 확인합니다. 선택 사항입니다.
- info - 변환에 따른 오류 관련 문자열입니다. 선택 사항입니다.
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다. 선택 사항입니다. 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다. 선택 사항입니다. 기본값은 0입니다.
- enforcedMatches - 일치를 적용하는 데 사용되는 DynamicFrame입니다. 선택 사항입니다. 기본값은 없습니다.
- computeMatchConfidenceScores - 각 일치 레코드 그룹의 신뢰도 점수를 컴퓨팅할지 여부를 나타내는 부울 값입니다. 선택 사항입니다. 기본값은 false입니다.

일치하는 레코드의 각 그룹에 고유 식별자가 할당된 새 DynamicFrame을 반환합니다.

FindMatches 클래스

입력 DynamicFrame에서 일치하는 레코드를 식별하고 일치하는 레코드의 각 그룹에 할당된 고유 식별자를 사용하여 새 DynamicFrame을 생성합니다.

가져오려면

```
from awsglueml.transforms import FindMatches
```

메서드

- Apply

apply(frame, transformId, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0, enforcedMatches = none, computeMatchConfidenceScores = 0)

입력 DynamicFrame에서 일치하는 레코드를 식별하고 일치하는 레코드의 각 그룹에 할당된 고유 식별자를 사용하여 새 DynamicFrame을 생성합니다.

- frame – FindMatches 변환을 적용할 DynamicFrame입니다. 필수 사항입니다.
- transformId – DynamicFrame의 레코드에 적용할 FindMatches 변환과 연결된 고유 ID입니다. 필수 사항입니다.
- transformation_ctx – 고유 문자열을 통해 통계/상태 정보를 확인합니다. 선택 사항입니다.
- info - 변환에 따른 오류 관련 문자열입니다. 선택 사항입니다.
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다. 선택 사항입니다. 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다. 선택 사항입니다. 기본값은 0입니다.
- enforcedMatches - 일치를 적용하는 데 사용되는 DynamicFrame입니다. 선택 사항입니다. 기본값은 없습니다.
- computeMatchConfidenceScores - 각 일치 레코드 그룹의 신뢰도 점수를 컴퓨팅할지 여부를 나타내는 부울 값입니다. 선택 사항입니다. 기본값은 false입니다.

일치하는 레코드의 각 그룹에 고유 식별자가 할당된 새 DynamicFrame을 반환합니다.

FlatMap 클래스

컬렉션에서 각 DynamicFrame에 변환을 적용합니다. 결과는 하나의 DynamicFrame으로 병합되지 않고 컬렉션으로 유지됩니다.

FlatMap 예제

다음 스니펫 예제에서는 FlatMap에 적용할 때 동적 프레임 컬렉션에서 ResolveChoice 변환을 사용하는 방법을 보여줍니다. 입력에 사용되는 데이터는 JSON에서 Amazon S3 주소 s3://bucket/path-for-data/sample.json 자리 표시자에 있으며, 다음 데이터를 포함합니다.

예시 JSON 데이터

```
[{
    "firstname": "Arnav",
    "lastname": "Desai",
    "address": {
        "street": "6 Anyroad Avenue",
        "city": "London",
        "state": "England",
        "country": "UK"
    },
    "phone": 17235550101,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Independent Research",
        "Government Department of Examples"
    ]
},
{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
    },
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
        "Example.io"
    ]
},
{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    },
    "phone": 12175550181,
```

```
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
    ]
}]
```

Example ResolveChoice를 DynamicFrameCollection에 적용하고 출력을 표시합니다.

```
#Read DynamicFrame
datasource = glueContext.create_dynamic_frame_from_options("s3", connection_options =
 {"paths":["s3://bucket/path/to/file/mysamplejson.json"]}, format="json")
datasource.printSchema()
datasource.show()

## Split to create a DynamicFrameCollection
split_frame=datasource.split_fields(["firstname","lastname","address"],"personal_info","busines
split_frame.keys()
print("---")

## Use FlatMap to run ResolveChoice
kwargs = {"choice": "cast:string"}
flat = FlatMap.apply(split_frame, ResolveChoice, frame_name="frame",
 transformation_ctx='tcx', **kwargs)
flat.keys()

##Select one of the DynamicFrames
personal_info = flat.select("personal_info")
personal_info.printSchema()
personal_info.show()
print("---")

business_info = flat.select("business_info")
business_info.printSchema()
business_info.show()
```

Important

`FlatMap.apply` 직접 호출 시 `frame_name` 파라미터는 반드시 `"frame"`이어야 합니다. 현재 다른 값은 허용되지 않습니다.

출력 예시

```
root
|-- firstname: string
|-- lastname: string
|-- address: struct
|    |-- street: string
|    |-- city: string
|    |-- state: string
|    |-- country: string
|-- phone: long
|-- affiliations: array
|    |-- element: string
---
{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
    },
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
        "Example.io"
    ]
}

{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    },
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
```

```
    ]
}
---
root
|-- firstname: string
|-- lastname: string
|-- address: struct
|    |-- street: string
|    |-- city: string
|    |-- state: string
|    |-- country: string

{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
    }
}

{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    }
}
---
root
|-- phone: long
|-- affiliations: array
|    |-- element: string

{
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
```

```
        "Example.io"
    ]
}

{
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
    ]
}
```

메서드

- __call__
- Apply
- 명칭
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(dfc, BaseTransform, frame_name, transformation_ctx = "", **base_kwargs)

컬렉션의 각 DynamicFrame에 변환을 적용하고 결과를 플랫합니다.

- dfc – 플랫맵할 DynamicFrameCollection입니다(필수).
- BaseTransform – GlueTransform로부터 얻어진 변환을 사용하여 컬렉션의 각 멤버에게 적용합니다(필수).
- frame_name - 컬렉션의 요소를 전달하기 위한 인수 이름입니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- base_kwargs - 기본 변환으로 전달을 위한 인수입니다(필수).

DynamicFrameCollection 원본의 변환을 각 DynamicFrame에 적용하여 생성한 새로운 DynamicFrameCollection을 반환합니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

Join 클래스

두 개의 `DynamicFrames`에 동일한 조인을 실행합니다.

예

`DynamicFrames`에 조인하려면 `DynamicFrame.join()` 메서드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예: 조인을 사용하여 `DynamicFrames` 결합 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs

- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(frame1, frame2, keys1, keys2, transformation_ctx = "")

두 개의 DynamicFrames에 동일한 조인을 실행합니다.

- frame1 - 조인할 첫 번째 DynamicFrame입니다(필수).
- frame2 - 조인할 두 번째 DynamicFrame입니다(필수).
- keys1 - 첫 번째 프레임을 조인할 키입니다(필수).
- keys2 - 두 번째 프레임을 조인할 키입니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).

두 DynamicFrames를 조인하여 생성된 새 DynamicFrame 항목을 반환합니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.

describeTransform(cls)

GlueTransform describeTransform에서 상속됩니다.

describeErrors(cls)

GlueTransform describeErrors에서 상속됩니다.

describe(cls)

GlueTransform describe에서 상속됩니다.

Map 클래스

입력 DynamicFrame의 모든 기록에 함수를 적용하여 새로운 DynamicFrame을 설계합니다.

예

DynamicFrame의 모든 레코드에 함수를 적용하는 DynamicFrame.map() 메서드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예: map을 사용하여 DynamicFrame의 모든 레코드에 함수를 적용합니다. 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

기존 DynamicFrame의 모든 DynamicRecords에 지정된 함수를 적용한 결과로써 새로운 DynamicFrame을 반환합니다.

- frame - 원본 DynamicFrame은 매핑 함수를 적용합니다(필수).
- f - DynamicFrame에 모든 DynamicRecords를 적용하려는 함수입니다. 함수는 매핑 결과에 따라 DynamicRecord를 논증으로 받아들이고 새로운 DynamicRecord를 반환합니다(필수).

  DynamicRecord는 DynamicFrame내 논리적 기록입니다. 자기 설명적이고 고정 스키마를 준수하지 않은 데이터에 사용될 수 있다는 점을 제외하면 Apache Spark DataFrame의 열과 비슷합니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).

- `stageThreshold` - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

기존 `DynamicFrame`의 모든 `DynamicRecords`에 지정된 함수를 적용한 결과로써 새로운 `DynamicFrame`을 반환합니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

MapToCollection 클래스

지정된 `DynamicFrameCollection`의 각 `DynamicFrame`에 변환을 적용합니다.

메서드

- __call__
- Apply

- 명칭
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(dfc, BaseTransform, frame_name, transformation_ctx = "", **base_kwargs)

지정된 DynamicFrameCollection의 각 DynamicFrame에 변환 함수를 적용합니다.

- dfc - 원본 DynamicFrameCollection은 변환 함수를 적용합니다(필수).
- callable - 호출 가능한 변환 함수는 각 컬렉션의 수에 적용됩니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).

DynamicFrameCollection 원본의 변환을 각 DynamicFrame에 적용하여 생성한 새로운 DynamicFrameCollection을 반환합니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.

describeTransform(cls)

GlueTransform describeTransform에서 상속됩니다.

describeErrors(cls)

GlueTransform describeErrors에서 상속됩니다.

describe(cls)

GlueTransform describe에서 상속됩니다.

Relationalize 클래스

중첩된 스키마를 DynamicFrame에 평면화하고 평면화된 프레임에서 배열의 열을 회전합니다.

예

DynamicFrame을 관계형으로 만들려면 DynamicFrame.relationalize() 메서드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예: 관계형 만들기를 사용하여 DynamicFrame 내 중첩된 스키마를 플랫하게 만들기 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, staging_path=None, name='roottable', options=None, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

DynamicFrame를 관계화하고 중첩되지 않은 열과 회전되는 배열 열이 생성한 프레임 목록을 생산합니다. 회전된 배열 열은 중첩되지 않는 상태에서 생성된 조인 키를 사용하여 루트 테이블에 연결될 수 있습니다.

- frame - 관계화할 DynamicFrame입니다(필수).
- staging_path - 메소드가 CSV 포맷으로 회전 테이블의 파티션을 저장하는 경로입니다(선택 사항). 회전 테이블은 이 경로부터 다시 읽습니다.
- name - 루트 테이블의 이름입니다(선택 사항).
- options - 선택적 파라미터의 딕셔너리입니다. 현재 사용되지 않습니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).

- `info` - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- `stageThreshold` - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

RenameField 클래스

`DynamicFrame` 내에서 노드 이름을 바꿉니다.

예

`DynamicFrame`에서 필드 이름을 다시 지정하기 위해 `DynamicFrame.rename_field()` 메소드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예: rename_field를 사용하여 `DynamicFrame`의 필드 이름을 변경합니다 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, old_name, new_name, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

DynamicFrame 내에서 노드 이름을 바꿉니다.

- frame – DynamicFrame은 노드 이름을 바꿉니다(필수).
- old_name - 이름을 바꿀 노드의 전체 경로입니다(필수).

  기존 이름에 점이 있으면 그 주위로 억음 부호(``)를 하지 않는 한 RenameField는 실행되지 않습니다. 예를 들어, this.old.name를 thisNewName로 바꾸려면 다음과 같이 RenameField를 불러올 수 있습니다.

  ```
  newDyF = RenameField(oldDyF, "`this.old.name`", "thisNewName")
  ```
- new_name - 전체 경로를 포함한 새로운 이름입니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

ResolveChoice 클래스

선택 유형을 `DynamicFrame` 내에서 해결합니다.

예

`DynamicFrame.resolveChoice()` 메서드를 사용하여 `DynamicFrame`에 여러 유형이 포함된 필드를 처리하는 것이 좋습니다. 코드에 대한 예제는 예: resolveChoice를 사용하여 여러 유형이 포함된 열을 처리합니다 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform

- describeErrors
- describe

__call__(frame, specs = none, choice = "", transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

모호한 유형을 DynamicFrame 내에서 해결하기 위한 정보를 제공합니다. 결과 DynamicFrame를 반환합니다.

- frame – DynamicFrame은 선택 유형을 해결합니다(필수).
- specs - 해결할 특정 모호성 목록이며, 각각은 (path, action)의 튜플 형식입니다. path 가치는 특정 모호한 요소를 확인하고 action 가치는 관련 해결 방안을 제안합니다.

  spec 및 choice 파라미터 중 하나만 사용할 수 있습니다. spec 파라미터가 None이 아니면 choice 파라미터는 빈 문자열이어야 합니다. 반대로 choice 파라미터가 빈 문자열이 아니면, spec 파라미터는 None이어야 합니다. 어떠한 파라미터도 제공되지 않을 경우, AWS Glue는 스키마를 구문분석하고 모호함을 해결하기 위해 사용합니다.

  specs 튜플의 action 부분에서 다음 해결 전략 중 하나를 지정할 수 있습니다.

  - cast - 보내버릴 유형을 명시할 수 있도록 도와줍니다. (예를 들어, cast:int).
  - make_cols - 데이터를 평면화하여 잠재적 모호성을 해결합니다. 예를 들어, columnA가 int 혹은 string이면 해결 방안은 DynamicFrame에서 columnA_int와 columnA_string으로 된 두 열을 생산하는 것입니다.
  - make_struct - 데이터를 나타내도록 구조를 사용하여 잠재적 모호성을 해결합니다. 예를 들어, 열에서 데이터가 int 혹은 string이면 make_struct 작업은 각 DynamicFrame와 int을 포함하는 결과인 string의 구조 열을 생성합니다.
  - project - 결과 DynamicFrame에 지정된 유형의 값만 유지하여 잠재적 모호성을 해결합니다. 예를 들어 ChoiceType 열에서 데이터가 int 혹은 string이면 project:string 작업은 string 유형이 아닌 결과 DynamicFrame의 값을 삭제합니다.

  만약 path가 배열을 확인하면, 빈 대괄호를 배열 이름 다음에 만들어 모호성을 피합니다. 예를 들어, 다음과 같은 구조화된 데이터와 작업한다고 가정합시다.

```
"myList": [
  { "price": 100.00 },
  { "price": "$100.00" }
]
```

path를 "myList[].price"로 설정하고, action을 "cast:double"로 설정하여 가격의 문자열 버전보다 숫자 버전을 선택할 수 있습니다.

- choice - specs 파라미터가 None인 경우의 기본 해결 작업입니다. specs 파라미터가 None이 아니면 이것은 어떠한 것에도 설정되지 않고 빈 문자열이어야 합니다.

  이 인수는 앞에서 설명된 specs 작업 외에 다음 작업을 지원합니다.

  - MATCH_CATALOG – 각 ChoiceType을 지정된 Data Catalog 테이블의 해당 유형에 캐스팅해봅니다.
- database — MATCH_CATALOG 선택 항목과 함께 사용할 AWS Glue 데이터 카탈로그 데이터베이스입니다(MATCH_CATALOG에 대해 필수).
- table_name— MATCH_CATALOG 작업에 사용할 AWS Glue 데이터 카탈로그 테이블 이름 입니다(MATCH_CATALOG에 대해 필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.

describeTransform(cls)

GlueTransform describeTransform에서 상속됩니다.

describeErrors(cls)

GlueTransform describeErrors에서 상속됩니다.

describe(cls)

GlueTransform describe에서 상속됩니다.

SelectFields 클래스

SelectFields 클래스는 기존 DynamicFrame에서 새 DynamicFrame 항목을 만들고 지정한 필드만 유지합니다. SelectFields는 SQL SELECT 문과 유사한 기능을 제공합니다.

예

DynamicFrame에서 필드를 선택하기 위해 DynamicFrame.select_fields() 방법을 사용하는 것이 좋습니다. 코드에 대한 예제는 예: select_fields를 사용하여 선택한 필드로 새 DynamicFrame 생성 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- 설명

__call__(frame, paths, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

DynamicFrame의 필드(노드)를 얻습니다.

- frame – DynamicFrame에서 필드를 선택합니다(필수).
- paths - 선택할 필드의 전체 경로 목록입니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).

- `stageThreshold` - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

특정 필드만 포함한 새로운 `DynamicFrame`을 반환합니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

SelectFromCollection 클래스

`DynamicFrameCollection`에서 `DynamicFrame` 하나를 선택합니다.

예

이 예제에서는 `SelectFromCollection`을 사용하여 `DynamicFrameCollection`에서 `DynamicFrame`를 선택합니다.

예제 데이터 세트

예제에서는 split_rows_collection을 호출한 DynamicFrameCollection에서 두 개의 DynamicFrames을 선택합니다. 다음은 split_rows_collection에서의 키 목록입니다.

```
dict_keys(['high', 'low'])
```

예제 코드

```
# Example: Use SelectFromCollection to select
# DynamicFrames from a DynamicFrameCollection

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import SelectFromCollection

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Select frames and inspect entries
frame_low = SelectFromCollection.apply(dfc=split_rows_collection, key="low")
frame_low.toDF().show()

frame_high = SelectFromCollection.apply(dfc=split_rows_collection, key="high")
frame_high.toDF().show()
```

출력

```
+---+-----+------------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+------------------------+-------------------------+
|  1|    0|                     fax|             202-225-3307|
|  1|    1|                   phone|             202-225-5731|
|  2|    0|                     fax|             202-225-3307|
|  2|    1|                   phone|             202-225-5731|
|  3|    0|                     fax|             202-225-3307|
|  3|    1|                   phone|             202-225-5731|
|  4|    0|                     fax|             202-225-3307|
|  4|    1|                   phone|             202-225-5731|
|  5|    0|                     fax|             202-225-3307|
|  5|    1|                   phone|             202-225-5731|
|  6|    0|                     fax|             202-225-3307|
|  6|    1|                   phone|             202-225-5731|
```

```
|  7|    0|                     fax|             202-225-3307|
|  7|    1|                   phone|             202-225-5731|
|  8|    0|                     fax|             202-225-3307|
|  8|    1|                   phone|             202-225-5731|
|  9|    0|                     fax|             202-225-3307|
|  9|    1|                   phone|             202-225-5731|
| 10|    0|                     fax|             202-225-6328|
| 10|    1|                   phone|             202-225-4576|
+---+-----+----------------------+-------------------------+
only showing top 20 rows

+---+-----+----------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-------------------------+
| 11|    0|                     fax|             202-225-6328|
| 11|    1|                   phone|             202-225-4576|
| 11|    2|                 twitter|           RepTrentFranks|
| 12|    0|                     fax|             202-225-6328|
| 12|    1|                   phone|             202-225-4576|
| 12|    2|                 twitter|           RepTrentFranks|
| 13|    0|                     fax|             202-225-6328|
| 13|    1|                   phone|             202-225-4576|
| 13|    2|                 twitter|           RepTrentFranks|
| 14|    0|                     fax|             202-225-6328|
| 14|    1|                   phone|             202-225-4576|
| 14|    2|                 twitter|           RepTrentFranks|
| 15|    0|                     fax|             202-225-6328|
| 15|    1|                   phone|             202-225-4576|
| 15|    2|                 twitter|           RepTrentFranks|
| 16|    0|                     fax|             202-225-6328|
| 16|    1|                   phone|             202-225-4576|
| 16|    2|                 twitter|           RepTrentFranks|
| 17|    0|                     fax|             202-225-6328|
| 17|    1|                   phone|             202-225-4576|
+---+-----+----------------------+-------------------------+
only showing top 20 rows
```

메서드

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(dfc, key, transformation_ctx = "")

`DynamicFrameCollection`에서 `DynamicFrame` 하나를 가져옵니다.

- `dfc` – `DynamicFrame`을 선택해야 하는 `DynamicFrameCollection`입니다(필수).
- `key` - 선택할 `DynamicFrame`의 키입니다(필수).
- `transformation_ctx` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

Simplify_ddb_json 클래스

DynamicFrame의 중첩된 열 중 특히 DynamoDB JSON 구조에 있는 열을 단순화하고 단순화된 새로운 DynamicFrame을 반환합니다.

예

특히 DynamoDB JSON 구조에 있는 DynamicFrame의 중첩 열을 단순화하려면 DynamicFrame.simplify_ddb_json() 메서드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예제: simplify_ddb_json을 사용하여 DynamoDB JSON 단순화 호출 단원을 참조하세요.

Spigot 클래스

AWS Glue 작업에 의해 수행된 변환을 확인하는 데 도움이 되도록 샘플 레코드를 지정된 대상에 기록합니다.

예

DynamicFrame.spigot() 메서드를 사용하여 DynamicFrame에서 지정한 대상에 레코드의 하위 집합을 쓰는 것이 좋습니다. 코드에 대한 예제는 예: spigot을 사용하여 DynamicFrame에서 Amazon S3에 샘플 필드를 작성합니다 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, path, options, transformation_ctx = "")

변환 중에 특정 대상 주소에 샘플 기록을 씁니다.

- frame - spigot으로 DynamicFrame입니다(필수).
- path - 작성할 기본 대상 주소 경로입니다(필수).

- `options` - 옵션을 지정하는 JSON 키-값 페어입니다(선택 사항). `"topk"` 옵션은 첫 번째 k 레코드가 작성되어야 한다는 것을 명시합니다. `"prob"` 옵션은 특정 레코드를 선택할 확률(10진수)을 명시합니다. 작성할 레코드를 선택할 때 사용합니다.
- `transformation_ctx` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

SplitFields 클래스

지정된 필드로 `DynamicFrame`을 두 개의 새로운 것으로 쪼갭니다.

예

`DynamicFrame`에서 레코드를 분할하기 위해 `DynamicFrame.split_fields()` 메소드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예제: split_fields를 사용하여 선택한 필드를 별도의 `DynamicFrame`로 분할합니다 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, paths, name1 = none, name2 = none, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

DynamicFrame의 하나 이상의 필드를 새로운 DynamicFrame으로 분할하고 남겨진 필드를 포함하는 새로운 다른 DynamicFrame을 생성합니다.

- frame – 소스 DynamicFrame으로 새로운 두 개로 스플릿합니다(필수).
- paths - 스플릿할 필드의 전체 경로 목록입니다(필수).
- name1 - 스플릿될 필드를 포함할 DynamicFrame에 지정되는 이름입니다. 이름이 제공되지 않으면 소스 프레임의 이름은 "1"이 붙여져 사용됩니다.
- name2 - 지정된 필드가 스플릿된 후 남겨진 필드를 포함하는 DynamicFrame에 지정되는 이름입니다. 이름이 제공되지 않으면 소스 프레임의 이름은 "2"이 붙여져 사용됩니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.

describeTransform(cls)

GlueTransform describeTransform에서 상속됩니다.

describeErrors(cls)

GlueTransform describeErrors에서 상속됩니다.

describe(cls)

GlueTransform describe에서 상속됩니다.

SplitRows 클래스

두 개의 DynamicFrames가 포함된 DynamicFrameCollection를 만듭니다. DynamicFrame은 분할 예정으로 지정된 행만 포함하고 다른 하나는 남겨진 모든 행을 포함합니다.

예

DynamicFrame의 행을 분할하려면 DynamicFrame.split_rows() 메서드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예제: split_rows를 사용하여 DynamicFrame 행을 분할합니다 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn

- describeTransform
- describeErrors
- describe

__call__(frame, comparison_dict, name1="frame1", name2="frame2", transformation_ctx = "", info = none, stageThreshold = 0, totalThreshold = 0)

새로운 DynamicFrame으로 DynamicFrame의 하나 이상의 열을 스플릿합니다.

- frame – 소스 DynamicFrame으로 새로운 두 개로 스플릿합니다(필수).
- comparison_dict - 열까지 전체 경로의 키와 열 값이 비교된 값의 매핑 비교기를 위한 다른 딕셔너리인 값의 딕셔너리입니다. 예를 들어, {"age": {">": 10, "<": 20}}는 "수명"이 범위를 벗어난 행에서 "수명" 값이 10과 20 사이인 행을 제외하고 스플릿합니다.
- name1 - 스플릿될 행을 포함할 DynamicFrame에 지정되는 이름입니다.
- name2 - 지정된 행이 스플릿된 후 남겨진 DynamicFrame에 지정되는 이름입니다.
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- stageThreshold - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- totalThreshold - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

Unbox 클래스

`DynamicFrame`의 문자열 필드를 개봉합니다(다시 포맷합니다).

예

`DynamicFrame`에서 필드를 레코드를 개봉하기 위해 `DynamicFrame.unbox()` 메소드를 사용하는 것이 좋습니다. 코드에 대한 예제는 예제: 개봉하기를 사용하여 문자열 필드를 구조에 개봉하기 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, path, format, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0, **options)

`DynamicFrame`의 문자열 필드를 개봉합니다.

- `frame` – `DynamicFrame`에서 필드를 개봉합니다(필수).

- `path` - 개봉할 `StringNode`로의 전체 경로입니다(필수).
- `format` - 포맷 사양입니다(선택 사항). 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용됩니다. 지원되는 포맷은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션를 참조하세요.
- `transformation_ctx` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- `info` - 변환에 따른 오류 관련 문자열입니다(선택 사항).
- `stageThreshold` - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.
- `separator` - 구분자 토큰입니다(선택 사항).
- `escaper` - 이스케이프 토큰입니다(선택 사항).
- `skipFirst` – 첫 번째 데이터 행을 건너뛰어야 한다면 `True`이고 건너뛰지 않아야 한다면 `False`입니다(선택 사항).
- withSchema – 개봉할 데이터 스키마를 포함하는 문자열입니다(선택 사항). 이것은 `StructType.json`을 사용하여 생성되어야 합니다.
- `withHeader` – 헤더를 포함해 데이터가 개봉되면 `True`이고 그렇지 않으면 `False`입니다(선택 사항).

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

GlueTransform describeErrors에서 상속됩니다.

describe(cls)

GlueTransform describe에서 상속됩니다.

UnnestFrame 클래스

DynamicFrame을 내보내고 중첩된 객체를 최상위 요소로 평면화하여 배열 객체의 조인 키를 생성합니다.

예

DynamicFrame.unnest() 메소드를 사용하여 DynamicFrame에서 중첩된 구조를 평면화하는 것이 좋습니다. 코드에 대한 예제는 예제: 중첩 해제를 사용하여 중첩된 필드를 최상위 필드로 전환합니다 단원을 참조하세요.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0)

DynamicFrame을 내보내고 중첩된 객체를 최상위 요소로 평면화하여 배열 객체의 조인 키를 생성합니다.

- frame - 중첩하지 않을 DynamicFrame입니다(필수).
- transformation_ctx - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- info - 변환에 따른 오류 관련 문자열입니다(선택 사항).

- `stageThreshold` - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항). 기본값은 0입니다.
- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항). 기본값은 0입니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

FlagDuplicatesInColumn 클래스

`FlagDuplicatesInColumn` 변환은 각 행에 지정된 값이 있는 새 열을 반환합니다. 이 열은 행의 소스 열 값이 소스 열의 이전 행 값과 일치하는지 여부를 나타냅니다. 일치 항목이 발견되면 중복으로 플래그가 지정됩니다. 초기 발생은 이전 행과 일치하지 않으므로 플래그가 지정되지 않습니다.

예

```
from pyspark.context import SparkContext
```

```
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

datasource1 = spark.read.json("s3://${BUCKET}/json/zips/raw/data")

try:
    df_output = column.FlagDuplicatesInColumn.apply(
        data_frame=datasource1,
        spark_context=sc,
        source_column="city",
        target_column="flag_col",
        true_string="True",
        false_string="False"
    )
except:
    print("Unexpected Error happened ")
    raise
```

출력

`FlagDuplicatesInColumn` 변환은 `df_output` DataFrame에 새 열 `flag_col`을 추가합니다. 이 열에는 해당 행의 `city` 열에 중복 값이 있는지 여부를 나타내는 문자열 값이 포함됩니다. 행에 중복된 `city` 값이 있는 경우 `flag_col`에는 `true_string` 값 'True'가 포함됩니다. 행에 고유한 `city` 값이 있는 경우 `flag_col`에는 `false_string` 값 'False'가 포함됩니다.

결과 `df_output` DataFrame에는 원래 `datasource1` DataFrame의 모든 열과 중복 `city` 값을 나타내는 추가 `flag_col` 열이 포함됩니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform

- describeErrors
- describe

__call__(spark_context, data_frame, source_column, target_column, true_string=DEFAULT_TRUE_STRING, false_string=DEFAULT_FALSE_STRING)

`FlagDuplicatesInColumn` 변환은 각 행에 지정된 값이 있는 새 열을 반환합니다. 이 열은 행의 소스 열 값이 소스 열의 이전 행 값과 일치하는지 여부를 나타냅니다. 일치 항목이 발견되면 중복으로 플래그가 지정됩니다. 초기 발생은 이전 행과 일치하지 않으므로 플래그가 지정되지 않습니다.

- `source_column` - 소스 열의 이름.
- `target_column` - 대상 열의 이름.
- `true_string` - 소스 열 값이 해당 열의 이전 값을 복제할 때 대상 열에 삽입할 문자열.
- `false_string` - 소스 열 값이 해당 열의 이전 값과 다를 때 대상 열에 삽입할 문자열.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

FormatPhoneNumber 클래스

`FormatPhoneNumber` 변환은 전화번호 문자열을 형식이 지정된 값으로 변환하는 열을 반환합니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        ("408-341-5669",),
        ("4083415669",)
    ],
    ["phone"],
)

try:
    df_output = column_formatting.FormatPhoneNumber.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="phone",
        default_region="US"
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 다음과 같습니다.

```
```

```
+---------------+
| phone|
+---------------+
|(408) 341-5669|
|(408) 341-5669|
+---------------+
```

`FormatPhoneNumber` 변환은 `source_column`을 `"phone"`으로, `default_region`을 `"US"`로 가져옵니다.

변환은 초기 형식에 관계없이 두 전화번호의 형식을 표준 미국 형식 `(408) 341-5669`로 지정합니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, source_column, phone_number_format=None, default_region=None, default_region_column=None)

`FormatPhoneNumber` 변환은 전화번호 문자열을 형식이 지정된 값으로 변환하는 열을 반환합니다.

- `source_column` – 기존 열의 이름입니다.
- `phone_number_format` - 전화번호를 변환할 형식. 형식을 지정하지 않은 경우 기본값은 국제적으로 인정되는 표준 전화번호 형식(`E.164`)입니다. 유효한 값은 다음과 같습니다.
  - E164(E 뒤의 마침표 생략)
- `default_region` - 번호 자체에 국가 코드가 없는 경우 전화번호의 리전을 지정하는 두 개 또는 세 개의 대문자로 구성된 유효한 리전 코드. 최대 `defaultRegion` 또는 `defaultRegionColumn` 중 하나를 제공할 수 있습니다.

- `default_region_column` - 고급 데이터 유형 `Country`의 열 이름. 지정된 열의 리전 코드는 번호 자체에 국가 코드가 없는 경우 전화번호의 국가 코드를 결정하는 데 사용됩니다. 최대 `defaultRegion` 또는 `defaultRegionColumn` 중 하나를 제공할 수 있습니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

FormatCase 클래스

`FormatCase` 변환은 열의 각 문자열을 지정된 케이스 유형으로 변경합니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
```

```
spark = SparkSession(sc)

datasource1 = spark.read.json("s3://${BUCKET}/json/zips/raw/data")

try:
    df_output = data_cleaning.FormatCase.apply(
        data_frame=datasource1,
        spark_context=sc,
        source_column="city",
        case_type="LOWER"
    )
except:
    print("Unexpected Error happened ")
    raise
```

출력

FormatCase 변환은 `case_type="LOWER"` 파라미터를 기반으로 `city` 열의 값을 소문자로 변환합니다. 결과 `df_output` DataFrame에는 원래 `datasource1` DataFrame의 모든 열이 포함되지만 `city` 열 값은 소문자로 표시됩니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, source_column, case_type)

FormatCase 변환은 열의 각 문자열을 지정된 케이스 유형으로 변경합니다.

- source_column – 기존 열의 이름입니다.
- case_type – 지원되는 케이스 유형은 CAPITAL, LOWER, UPPER, SENTENCE입니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

FillWithMode 클래스

`FillWithMode` 변환은 지정한 전화번호 형식에 따라 열의 형식을 지정합니다. 일부 값이 동일한 타이 브레이커 로직을 지정할 수도 있습니다. 예를 들어 다음과 같은 입력 값을 고려합니다. `1 2 2 3 3 4`

modeType이 `MINIMUM`이면 `FillWithMode`에서 모드 값으로 2를 반환합니다. modeType이 `MAXIMUM`이면 모드는 3입니다. `AVERAGE`의 경우 모드는 2.5입니다.

예

```
from awsglue.context import *
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)
```

```
input_df = spark.createDataFrame(
    [
        (105.111, 13.12),
        (1055.123, 13.12),
        (None, 13.12),
        (13.12, 13.12),
        (None, 13.12),
    ],
    ["source_column_1", "source_column_2"],
)

try:
    df_output = data_quality.FillWithMode.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_1",
        mode_type="MAXIMUM"
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

출력

지정된 코드의 출력은 다음과 같습니다.

```
```
+---------------+---------------+
|source_column_1|source_column_2|
+---------------+---------------+
105.111	13.12
1055.123	13.12
1055.123	13.12
13.12	13.12
1055.123	13.12
+---------------+---------------+
```
```

`awsglue.data_quality` 모듈의 `FillWithMode` 변환은 `input_df` DataFrame에 적용됩니다. `source_column_1` 열의 `null` 값을 해당 열의 null이 아닌 값 중 최댓값(`mode_type="MAXIMUM"`)으로 바꿉니다.

이 경우 source_column_1 열의 최댓값은 `1055.123`입니다. 따라서 source_column_1의 `null` 값은 출력 DataFrame `df_output`에서 `1055.123`으로 바뀝니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, source_column, mode_type)

FillWithMode 변환은 열의 문자열 케이스 형식을 지정합니다.

- source_column – 기존 열의 이름입니다.
- mode_type - 데이터의 타이 값을 확인하는 방법. 이 값은 MINIMUM, NONE, AVERAGE 또는 MAXIMUM 중 하나여야 합니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

FlagDuplicateRows 클래스

`FlagDuplicateRows` 변환은 각 행에 지정된 값이 있는 새 열을 반환합니다. 이 열은 해당 행이 데이터세트의 이전 행과 정확히 일치하는지 여부를 나타냅니다. 일치 항목이 발견되면 중복으로 플래그가 지정됩니다. 초기 발생은 이전 행과 일치하지 않으므로 플래그가 지정되지 않습니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (105.111, 13.12),
        (13.12, 13.12),
        (None, 13.12),
        (13.12, 13.12),
        (None, 13.12),
    ],
    ["source_column_1", "source_column_2"],
)

try:
    df_output = data_quality.FlagDuplicateRows.apply(
        data_frame=input_df,
        spark_context=sc,
        target_column="flag_row",
        true_string="True",
```

```
        false_string="False",
        target_index=1
    )
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 source_column_1 열을 기반으로 행의 중복 여부를 나타내는 추가 열 flag_row를 포함하는 PySpark DataFrame입니다. 결과 `df_output` DataFrame에는 다음 행이 포함됩니다.

```
```
+---------------+---------------+--------+
|source_column_1|source_column_2|flag_row|
+---------------+---------------+--------+
105.111	13.12	False
13.12	13.12	True
null	13.12	True
13.12	13.12	True
null	13.12	True
+---------------+---------------+--------+
```
```

flag_row 열은 행이 중복인지 여부를 나타냅니다. `true_string`은 'True'로 설정되고 `false_string`은 'False'로 설정됩니다. `target_index`는 1로 설정됩니다. 즉, 출력 DataFrame의 두 번째 위치(인덱스 1)에 flag_row 열이 삽입됩니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, target_column, true_string=DEFAULT_TRUE_STRING, false_string=DEFAULT_FALSE_STRING, target_index=None)

`FlagDuplicateRows` 변환은 각 행에 지정된 값이 있는 새 열을 반환합니다. 이 열은 해당 행이 데이터세트의 이전 행과 정확히 일치하는지 여부를 나타냅니다. 일치 항목이 발견되면 중복으로 플래그가 지정됩니다. 초기 발생은 이전 행과 일치하지 않으므로 플래그가 지정되지 않습니다.

- `true_string` - 행이 이전 행과 일치하는 경우 삽입할 값.
- `false_string` - 행이 고유할 경우 삽입할 값.
- `target_column` - 데이터세트에 삽입된 새 열의 이름.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

RemoveDuplicates 클래스

`RemoveDuplicates` 변환은 선택한 소스 열에서 중복 값이 발생하는 경우 전체 행을 삭제합니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (105.111, 13.12),
        (13.12, 13.12),
        (None, 13.12),
        (13.12, 13.12),
        (None, 13.12),
    ],
    ["source_column_1", "source_column_2"],
)

try:
    df_output = data_quality.RemoveDuplicates.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_1"
    )
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 source_column_1 열에 따라 중복이 제거된 PySpark DataFrame입니다. 결과 `df_output` DataFrame에는 다음 행이 포함됩니다.

```
```
+---------------+---------------+
|source_column_1|source_column_2|
+---------------+---------------+
105.111	13.12
13.12	13.12
null	13.12
```
```

```
+---------------+---------------+
```
```

source_column_1 열에 따라 중복이 제거되므로 source_column_1 값이 `13.12` 및 `null`인 행은 출력 DataFrame에 한 번만 표시됩니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, source_column)

RemoveDuplicates 변환은 선택한 소스 열에서 중복 값이 발생하는 경우 전체 행을 삭제합니다.

- source_column – 기존 열의 이름입니다.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.

describeArgs(cls)

GlueTransform describeArgs에서 상속됩니다.

describeReturn(cls)

GlueTransform describeReturn에서 상속됩니다.
```

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

MonthName 클래스

`MonthName` 변환은 날짜를 나타내는 문자열에서 월의 이름이 포함된 새 열을 생성합니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")

input_df = spark.createDataFrame(
    [
        ("20-2018-12",),
        ("2018-20-12",),
        ("20182012",),
        ("12202018",),
        ("20122018",),
        ("20-12-2018",),
        ("12/20/2018",),
        ("02/02/02",),
        ("02 02 2009",),
        ("02/02/2009",),
        ("August/02/2009",),
        ("02/june/2009",),
        ("02/2020/june",),
        ("2013-02-21 06:35:45.658505",),
        ("August 02 2009",),
```

```
        ("2013/02/21",),
        (None,),
    ],
    ["column_1"],
)

try:
    df_output = datetime_functions.MonthName.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="column_1",
        target_column="target_column"
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 다음과 같습니다.

```
```
+------------+------------+
| column_1|target_column|
+------------+------------+
20-2018-12	December
2018-20-12	null
20182012	null
12202018	null
20122018	null
20-12-2018	December
12/20/2018	December
02/02/02	February
02 02 2009	February
02/02/2009	February
August/02/2009	August
02/june/2009	null
02/2020/june	null
2013-02-21 06:35:45.658505	February
August 02 2009	August
2013/02/21	February
null	null
```
```

```
+------------+------------+
```
```

MonthName 변환은 `source_column`을 `"column_1"`로, `target_column`을 `"target_column"`으로 가져옵니다. `"column_1"` 열의 날짜/시간 문자열에서 월 이름을 추출하여 `"target_column"` 열에 배치하려고 합니다. 날짜/시간 문자열이 인식할 수 없는 형식이거나 구문 분석할 수 없는 경우 `"target_column"` 값이 `null`로 설정됩니다.

변환은 '20-12-2018', '12/20/2018', '02/02/2009', '2013-02-21 06:35:45.658505', 'August 02 2009'와 같은 다양한 날짜/시간 형식에서 월 이름을 성공적으로 추출합니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, target_column, source_column=None, value=None)

MonthName 변환은 날짜를 나타내는 문자열에서 월의 이름이 포함된 새 열을 생성합니다.

- source_column – 기존 열의 이름입니다.
- value - 평가할 문자열.
- target_column - 새로 생성된 열의 이름.

apply(cls, *args, **kwargs)

GlueTransform apply에서 상속됩니다.

name(cls)

GlueTransform name에서 상속됩니다.
```

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

IsEven 클래스

`IsEven` 변환은 소스 열 또는 값이 짝수인지 여부를 나타내는 부울 값을 새 열에서 반환합니다. 소스 열 또는 값이 소수인 경우 결과는 false입니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [(5,), (0,), (-1,), (2,), (None,)],
    ["source_column"],
)

try:
    df_output = math_functions.IsEven.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column",
```

```
        target_column="target_column",
        value=None,
        true_string="Even",
        false_string="Not even",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 다음과 같습니다.

```
```
+------------+------------+
|source_column|target_column|
+------------+------------+
5	Not even
0	Even
-1	Not even
2	Even
null	null
+------------+------------+
```
```

IsEven 변환은 `source_column`을 'source_column'으로, `target_column`을 'target_column'으로 가져옵니다. `"source_column"`의 값이 짝수인지 확인합니다. 값이 짝수인 경우 `"target_column"`값을 `true_string` 'Even'으로 설정합니다. 값이 홀수인 경우 `"target_column"`값을 `false_string` 'Not even'으로 설정합니다. `"source_column"` 값이 `null`인 경우 `"target_column"` 값은 `null`로 설정됩니다.

변환은 짝수(0 및 2)를 올바르게 식별하고 "target_column"` 값을 'Even'으로 설정합니다. 홀수(5 및 -1)의 경우 "target_column"` 값을 'Not even'으로 설정합니다. `"source_column"` 값이 `null`인 경우 `"target_column"` 값은 `null`로 설정됩니다.

메서드

- __call__
- apply

- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, target_column, source_column=None, true_string=DEFAULT_TRUE_STRING, false_string=DEFAULT_FALSE_STRING, value=None)

`IsEven` 변환은 소스 열 또는 값이 짝수인지 여부를 나타내는 부울 값을 새 열에서 반환합니다. 소스 열 또는 값이 소수인 경우 결과는 false입니다.

- `source_column` – 기존 열의 이름입니다.
- `target_column` - 생성할 새 열의 이름.
- `true_string` - 값이 짝수인지 여부를 나타내는 문자열.
- `false_string` - 값이 짝수가 아닌지 여부를 나타내는 문자열.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

CryptographicHash 클래스

`CryptographicHash` 변환은 열에서 값을 해시하는 데 알고리즘을 적용합니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

secret = "${SECRET}"
sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (1, "1234560000"),
        (2, "1234560001"),
        (3, "1234560002"),
        (4, "1234560003"),
        (5, "1234560004"),
        (6, "1234560005"),
        (7, "1234560006"),
        (8, "1234560007"),
        (9, "1234560008"),
        (10, "1234560009"),
    ],
    ["id", "phone"],
)

try:
    df_output = pii.CryptographicHash.apply(
        data_frame=input_df,
        spark_context=sc,
        source_columns=["id", "phone"],
        secret_id=secret,
```

```
        algorithm="HMAC_SHA256",
        output_format="BASE64",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 다음과 같습니다.

```
```
+---+------------+------------------+------------------+
| id| phone | id_hashed | phone_hashed |
+---+------------+------------------+------------------+
1	1234560000	QUI1zXTJiXmfIb...	juDBAmiRnnO3g...
2	1234560001	ZAUWiZ3dVTzCo...	vC8lgUqBVDMNQ...
3	1234560002	ZP4VvZWkqYifu...	Kl3QAkgswYpzB...
4	1234560003	3u8vO3wQ8EQfj...	CPBzK1P8PZZkV...
5	1234560004	eWkQJk4zAOIzx...	aLf7+mHcXqbLs...
6	1234560005	xtI9fZCJZCvsa...	dy2DFgdYWmr0p...
7	1234560006	iW9hew7jnHuOf...	wwfGMCOEv6oOv...
8	1234560007	H9V1pqvgkFhfS...	g9WKhagIXy9ht...
9	1234560008	xDhEuHaxAUbU5...	b3uQLKPY+Q5vU...
10	1234560009	GRN6nFXkxk349...	VJdsKt8VbxBbt...
+---+------------+------------------+------------------+
```
```

변환은 지정된 알고리즘과 시크릿 키를 사용하여 `id` 및 `phone` 열 값의 암호화 해시를 계산하고 Base64 형식으로 해시를 인코딩합니다. 결과 `df_output` DataFrame에는 원래 `input_df` DataFrame의 모든 열과 계산된 해시가 포함된 추가 `id_hashed` 및 `phone_hashed` 열이 있습니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn

- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, source_columns, secret_id, algorithm=None, secret_version=None, create_secret_if_missing=False, output_format=None, entity_type_filter=None)

`CryptographicHash` 변환은 열에서 값을 해시하는 데 알고리즘을 적용합니다.

- `source_columns` - 기존 열의 배열.
- `secret_id` – Secrets Manager 시크릿 키의 ARN. 소스 열을 해시하기 위해 해시 기반 메시지 인증 코드(HMAC) 접두사 알고리즘에 사용되는 키.
- `secret_version` – 선택 사항입니다. 기본적으로 최신 시크릴 버전으로 설정됩니다.
- `entity_type_filter` - 엔터티 유형의 선택적 배열. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.
- `create_secret_if_missing` - 선택적 부울. true인 경우 호출자를 대신하여 시크릿을 생성하려고 시도합니다.
- `algorithm` – 데이터를 해시하는 데 사용되는 알고리즘. 유효한 열거형 값: MD5, SHA1, SHA256, SHA512, HMAC_MD5, HMAC_SHA1, HMAC_SHA256, HMAC_SHA512.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

Decrypt 클래스

`Decrypt` 변환은 AWS Glue 내부에서 복호화를 수행합니다. AWS 암호화 SDK를 사용하여 AWS Glue 외부에서 데이터를 복호화할 수도 있습니다. 제공된 KMS 키 ARN이 열을 암호화하는 데 사용된 것과 일치하지 않으면 복호화 작업이 실패합니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

kms = "${KMS}"
sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (1, "1234560000"),
        (2, "1234560001"),
        (3, "1234560002"),
        (4, "1234560003"),
        (5, "1234560004"),
        (6, "1234560005"),
        (7, "1234560006"),
        (8, "1234560007"),
        (9, "1234560008"),
        (10, "1234560009"),
    ],
    ["id", "phone"],
)

try:
    df_encrypt = pii.Encrypt.apply(
        data_frame=input_df,
        spark_context=sc,
```

```
        source_columns=["phone"],
        kms_key_arn=kms
    )
    df_decrypt = pii.Decrypt.apply(
        data_frame=df_encrypt,
        spark_context=sc,
        source_columns=["phone"],
        kms_key_arn=kms
    )
    df_decrypt.show()
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 원본 `id` 열과 복호화된 `phone` 열이 있는 PySpark DataFrame입니다.

```
```
+---+------------+
| id| phone|
+---+------------+
1	1234560000
2	1234560001
3	1234560002
4	1234560003
5	1234560004
6	1234560005
7	1234560006
8	1234560007
9	1234560008
10	1234560009
+---+------------+
```
```

`Encrypt` 변환은 `source_columns`를 `["phone"]`으로, `kms_key_arn`을 `${KMS}` 환경 변수의 값으로 가져옵니다. 변환은 지정된 KMS 키를 사용하여 `phone` 열의 값을 암호화합니다. 그런 다음, 암호화된 DataFrame `df_encrypt`가 `awsglue.pii` 모듈에서 `Decrypt` 변환으로 전달됩니다. 이때 `source_columns`를 `["phone"]`으로, `kms_key_arn`을 `${KMS}` 환경 변수의 값으로 가져옵니다. 변환은 동일한 KMS 키를 사용하여 `phone` 열의 암호화된 값을 복호화합니다. 결과 `df_decrypt` DataFrame에는 원본 `id` 열과 복호화된 `phone` 열이 포함되어 있습니다.

메서드

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, source_columns, kms_key_arn)

`Decrypt` 변환은 AWS Glue 내부에서 복호화를 수행합니다. AWS 암호화 SDK를 사용하여 AWS Glue 외부에서 데이터를 복호화할 수도 있습니다. 제공된 KMS 키 ARN이 열을 암호화하는 데 사용된 것과 일치하지 않으면 복호화 작업이 실패합니다.

- `source_columns` - 기존 열의 배열.
- `kms_key_arn` - 소스 열을 복호화하는 데 사용할 AWS Key Management Service 키의 키 ARN.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

Encrypt 클래스

`Encrypt` 변환은 AWS Key Management Service 키를 사용하여 소스 열을 암호화합니다. `Encrypt` 변환은 셀당 최대 128MiB를 암호화할 수 있습니다. 복호화할 때 형식을 보존하려고 시도합니다. 데이터 유형을 보존하려면 데이터 유형 메타데이터를 1KB 미만으로 직렬화해야 합니다. 그렇지 않으면 `preserve_data_type` 파라미터를 false로 설정해야 합니다. 데이터 유형 메타데이터는 암호화 컨텍스트에서 일반 텍스트로 저장됩니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

kms = "${KMS}"
sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (1, "1234560000"),
        (2, "1234560001"),
        (3, "1234560002"),
        (4, "1234560003"),
        (5, "1234560004"),
        (6, "1234560005"),
        (7, "1234560006"),
        (8, "1234560007"),
        (9, "1234560008"),
        (10, "1234560009"),
    ],
    ["id", "phone"],
)

try:
```

```
    df_encrypt = pii.Encrypt.apply(
        data_frame=input_df,
        spark_context=sc,
        source_columns=["phone"],
        kms_key_arn=kms
    )
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 원래 `id` 열과 `phone` 열의 암호화된 값이 포함된 추가 열이 있는 PySpark DataFrame입니다.

```
```
+---+------------+------------------------+
| id| phone | phone_encrypted |
+---+------------+------------------------+
1	1234560000	EncryptedData1234...abc
2	1234560001	EncryptedData5678...def
3	1234560002	EncryptedData9012...ghi
4	1234560003	EncryptedData3456...jkl
5	1234560004	EncryptedData7890...mno
6	1234560005	EncryptedData1234...pqr
7	1234560006	EncryptedData5678...stu
8	1234560007	EncryptedData9012...vwx
9	1234560008	EncryptedData3456...yz0
10	1234560009	EncryptedData7890...123
+---+------------+------------------------+
```
```

`Encrypt` 변환은 `source_columns`를 `["phone"]`으로, `kms_key_arn`을 `${KMS}` 환경 변수의 값으로 가져옵니다. 변환은 지정된 KMS 키를 사용하여 `phone` 열의 값을 암호화합니다. 결과 `df_encrypt` DataFrame에는 원래 `id` 열, 원래 `phone` 열, `phone_encrypted` 열의 암호화된 값이 포함된 `phone_encrypted`라는 추가 열이 포함되어 있습니다.

메서드

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, source_columns, kms_key_arn, entity_type_filter=None, preserve_data_type=None)

`Encrypt` 변환은 AWS Key Management Service 키를 사용하여 소스 열을 암호화합니다.

- `source_columns` - 기존 열의 배열.
- `kms_key_arn` - 소스 열을 암호화하는 데 사용할 AWS Key Management Service 키의 키 ARN.
- `entity_type_filter` - 엔터티 유형의 선택적 배열. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.
- `preserve_data_type` - 선택적 부울. 기본값은 true입니다. false인 경우 데이터 유형이 저장되지 않습니다.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

IntToIp 클래스

`IntToIp` 변환은 소스 열의 정수 값 또는 기타 값을 대상 열의 대응하는 IPv4 값으로 변환한 다음, 새 열에서 결과를 반환합니다.

예

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (3221225473,),
        (0,),
        (1,),
        (100,),
        (168430090,),
        (4294967295,),
        (4294967294,),
        (4294967296,),
        (-1,),
        (None,),
    ],
    ["source_column_int"],
)

try:
    df_output = web_functions.IntToIp.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_int",
        target_column="target_column",
```

```
        value=None
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

출력

출력은 다음과 같습니다.

```
```
+---------------+---------------+
|source_column_int|target_column|
+---------------+---------------+
3221225473	192.0.0.1
0	0.0.0.0
1	0.0.0.1
100	0.0.0.100
168430090	10.0.0.10
4294967295	255.255.255.255
4294967294	255.255.255.254
4294967296	null
-1	null
null	null
+---------------+---------------+
```
```

`IntToIp.apply` 변환은 `source_column`을 `"source_column_int"`로, `target_column`을 `"target_column"`으로 변환하고 `source_column_int` 열의 정수 값을 대응하는 IPv4 주소 표현으로 변환한 다음, `target_column` 열에 결과를 저장합니다.

IPv4 주소 범위(0~4294967295) 내 유효한 정수 값의 경우 변환은 해당 값을 IPv4 주소 표현(예: 192.0.0.1, 0.0.0.0, 10.0.0.10, 255.255.255.255)으로 변환합니다.

유효한 범위를 벗어난 정수 값(예: 4294967296, -1)의 경우 `target_column` 값이 `null`로 설정됩니다. `source_column_int` 열에 `null` 값이 있는 경우 `target_column` 값도 `null`로 설정됩니다.

메서드

- __call__

- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, target_column, source_column=None, value=None)

`IntToIp` 변환은 소스 열의 정수 값 또는 기타 값을 대상 열의 대응하는 IPv4 값으로 변환한 다음, 새 열에서 결과를 반환합니다.

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

IpToInt 클래스

`IpToInt` 변환은 소스 열의 Internet Protocol 버전 4(IPv4) 값 또는 기타 값을 대상 열의 대응하는 정수 값으로 변환한 다음, 새 열에서 결과를 반환합니다.

예

AWS Glue 4.0 이상의 경우 `key: --enable-glue-di-transforms, value: true`를 사용하여 작업 인수를 생성하거나 업데이트합니다.

```
from pyspark.context import SparkContext
from awsgluedi.transforms import *

sc = SparkContext()

input_df = spark.createDataFrame(
    [
        ("192.0.0.1",),
        ("10.10.10.10",),
        ("1.2.3.4",),
        ("1.2.3.6",),
        ("http://12.13.14.15",),
        ("https://16.17.18.19",),
        ("1.2.3.4",),
        (None,),
        ("abc",),
        ("abc.abc.abc.abc",),
        ("321.123.123.123",),
        ("244.4.4.4",),
        ("255.255.255.255",),
    ],
    ["source_column_ip"],
)

    df_output = web_functions.IpToInt.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_ip",
        target_column="target_column",
        value=None
```

```
    )
    df_output.show()
```

출력

출력은 다음과 같습니다.

```
```
+----------------+---------------+
|source_column_ip| target_column|
+----------------+---------------+
192.0.0.1	3221225473
10.10.10.10	168427722
1.2.3.4	16909060
1.2.3.6	16909062
http://12.13.14.15	null
https://16.17.18.19	null
1.2.3.4	16909060
null	null
abc	null
abc.abc.abc.abc	null
321.123.123.123	null
244.4.4.4	4102444804
255.255.255.255	4294967295
+----------------+---------------+
```
```

`IpToInt` 변환은 `source_column`을 `"source_column_ip"`로, `target_column`을 `"target_column"`으로 변환하고 `source_column_ip` 열의 유효한 IPv4 주소 문자열을 대응하는 32비트 정수 표현으로 변환한 다음, `target_column` 열에 결과를 저장합니다.

유효한 IPv4 주소 문자열(예: '192.0.0.1', '10.10.10.10', '1.2.3.4')의 경우 변환은 해당 값을 정수 표현(예: 3221225473, 168427722, 16909060)으로 변환합니다. 유효한 IPv4 주소가 아닌 문자열(예: URL , 'abc'와 같은 IP 외 문자열, 'abc.abc.abc.abc'와 같은 유효하지 않은 IP 형식)의 경우 `target_column` 값은 `null`로 설정됩니다. `source_column_ip` 열에 `null` 값이 있는 경우 `target_column` 값도 `null`로 설정됩니다.

메서드

- __call__
- apply

- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(spark_context, data_frame, target_column, source_column=None, value=None)

`IpToInt` 변환은 소스 열의 Internet Protocol 버전 4(IPv4) 값 또는 기타 값을 대상 열의 대응하는 정수 값으로 변환한 다음, 새 열에서 결과를 반환합니다.

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

apply(cls, *args, **kwargs)

`GlueTransform` apply에서 상속됩니다.

name(cls)

`GlueTransform` name에서 상속됩니다.

describeArgs(cls)

`GlueTransform` describeArgs에서 상속됩니다.

describeReturn(cls)

`GlueTransform` describeReturn에서 상속됩니다.

describeTransform(cls)

`GlueTransform` describeTransform에서 상속됩니다.

describeErrors(cls)

`GlueTransform` describeErrors에서 상속됩니다.

describe(cls)

`GlueTransform` describe에서 상속됩니다.

데이터 통합 변환

AWS Glue 4.0 이상의 경우 `key: --enable-glue-di-transforms, value: true`를 사용하여 작업 인수를 생성하거나 업데이트하세요.

작업 스크립트 예제:

```
from pyspark.context import SparkContext

from awsgluedi.transforms import *
sc = SparkContext()

input_df = spark.createDataFrame(
    [(5,), (0,), (-1,), (2,), (None,)],
    ["source_column"],
)

try:
    df_output = math_functions.IsEven.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column",
        target_column="target_column",
        value=None,
        true_string="Even",
        false_string="Not even",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

노트북을 사용한 세션 예제

```
%idle_timeout 2880
%glue_version 4.0
%worker_type G.1X
%number_of_workers 5
```

```
%region eu-west-1
```

```
%%configure
{
    "--enable-glue-di-transforms": "true"
}
```

```
from pyspark.context import SparkContext
from awsgluedi.transforms import *

sc = SparkContext()

input_df = spark.createDataFrame(
    [(5,), (0,), (-1,), (2,), (None,)],
    ["source_column"],
)

try:
    df_output = math_functions.IsEven.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column",
        target_column="target_column",
        value=None,
        true_string="Even",
        false_string="Not even",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

AWS CLI를 사용한 세션 예제

```
aws glue create-session --default-arguments "--enable-glue-di-transforms=true"
```

DI 변환:

- FlagDuplicatesInColumn 클래스
- FormatPhoneNumber 클래스
- FormatCase 클래스

- FillWithMode 클래스
- FlagDuplicateRows 클래스
- RemoveDuplicates 클래스
- MonthName 클래스
- IsEven 클래스
- CryptographicHash 클래스
- Decrypt 클래스
- Encrypt 클래스
- IntToIp 클래스
- IpToInt 클래스

Maven: Spark 애플리케이션과 플러그인 번들링

Spark 애플리케이션을 로컬에서 개발하는 동안 Maven pom.xml에 플러그인 종속성을 추가하여 Spark 애플리케이션 및 Spark 배포(버전 3.3)와 변환 종속성을 번들링할 수 있습니다.

```
<repositories>
   ...
    <repository>
        <id>aws-glue-etl-artifacts</id>
        <url>https://aws-glue-etl-artifacts.s3.amazonaws.com/release/ </url>
    </repository>
</repositories>
...
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>AWSGlueTransforms</artifactId>
    <version>4.0.0</version>
</dependency>
```

또는 다음과 같이 AWS Glue Maven 아티팩트에서 바이너리를 직접 다운로드하여 Spark 애플리케이션에 포함할 수도 있습니다.

```
#!/bin/bash
sudo wget -v https://aws-glue-etl-artifacts.s3.amazonaws.com/release/com/amazonaws/
AWSGlueTransforms/4.0.0/AWSGlueTransforms-4.0.0.jar -P /usr/lib/spark/jars/
```

Scala의 AWS Glue ETL 스크립트 프로그래밍

AWS Glue용 Scala 코드 예제와 유틸리티는 GitHub 웹 사이트의 AWS Glue 샘플 리포지토리에서 찾을 수 있습니다.

AWS Glue는 추출, 변환, 로드 작업 스크립트의 PySpark Scala의 확장 언어를 지원합니다. 다음 섹션은 AWS GlueScala 라이브러리와 ETL 스크립트의 AWS Glue API를 사용하고 라이브러리용 참조 문서를 제공하는 방법에 대해 설명합니다.

목차

- Scala 사용하여 AWS Glue ETL 스크립트 프로그래밍
  - Scala ETL 프로그램을 개발 엔드포인트의 Jupyter Notebook에서 테스트
  - Scala REPL에서 Scala ETL 프로그램 테스트
- Scala 스크립트 예 - 스트리밍 ETL
- AWS Glue Scala 라이브러리의 API
  - com.amazonaws.services.glue
  - com.amazonaws.services.glue.ml
  - com.amazonaws.services.glue.dq
  - com.amazonaws.services.glue.types
  - com.amazonaws.services.glue.util
  - AWS Glue Scala ChoiceOption API
    - ChoiceOption 특성
    - ChoiceOption 객체
      - 정의 적용
    - 케이스 클래스 ChoiceOptionWithResolver
    - 케이스 클래스 MatchCatalogSchemaChoiceOption
  - 추상 DataSink 클래스
    - writeDynamicFrame 정의
    - pyWriteDynamicFrame 정의
    - Def writeDataFrame
    - Def pyWriteDataFrame
    - setCatalogInfo 정의
    - supportsFormat 정의

    - setFormat 정의
    - withFormat 정의
    - setAccumulableSize 정의
    - getOutputErrorRecordsAccumulable 정의
    - errorsAsDynamicFrame 정의
    - DataSink 객체
        - recordMetrics 정의
- AWS Glue Scala DataSource 특성
- AWS Glue Scala DynamicFrame API
    - AWS Glue Scala DynamicFrame 클래스
        - val errorsCount
        - def applyMapping
        - def assertErrorThreshold
        - def count
        - def dropField
        - def dropFields
        - def dropNulls
        - def errorsAsDynamicFrame
        - def filter
        - def getName
        - def getNumPartitions
        - def getSchemaIfComputed
        - def isSchemaComputed
        - def javaToPython
        - def join
        - def map
        - def mergeDynamicFrames
        - def printSchema
        - def recomputeSchema
        - def relationalize

        - def renameField
        - def repartition
        - def resolveChoice
        - def schema
        - def selectField
        - def selectFields
        - def show
        - Def simplifyDDBJson
        - def spigot
        - def splitFields
        - def splitRows
        - def stageErrorsCount
        - def toDF
        - def unbox
        - def unnest
        - def unnestDDBJson
        - def withFrameSchema
        - def withName
        - def withTransformationContext
    - DynamicFrame 객체
        - def apply
        - def emptyDynamicFrame
        - def fromPythonRDD
        - def ignoreErrors
        - def inlineErrors
        - def newFrameWithErrors
- AWS Glue Scala DynamicRecord 클래스
    - def addField
    - def dropField
    - def setError

- def isError
- def getError
- def clearError
- def write
- def readFields
- def clone
- def schema
- def getRoot
- def toJson
- def getFieldNode
- def getField
- def hashCode
- def equals
- DynamicRecord 객체
  - 정의 적용
- RecordTraverser 특성

- AWS Glue Scala GlueContext API
  - def addIngestionColumns
  - def createDataFrameFromOptions
  - forEachBatch
  - def getCatalogSink
  - def getCatalogSource
  - def getJDBCSink
  - def getSink
  - def getSinkWithFormat
  - def getSource
  - def getSourceWithFormat
  - def getSparkSession
  - def startTransaction
  - def commitTransaction

    - def cancelTransaction
    - def this
    - def this
    - def this
- MappingSpec
    - MappingSpec 케이스 클래스
    - MappingSpec 객체
    - Val orderingByTarget
    - 정의 적용
    - 정의 적용
    - 정의 적용
- AWS Glue Scala ResolveSpec API
    - ResolveSpec 객체
        - def
        - def
    - ResolveSpec 케이스 클래스
        - ResolveSpec def 메서드
- AWS Glue Scala ArrayNode API
    - ArrayNode 케이스 클래스
        - ArrayNode 정의 메서드
- AWS Glue Scala BinaryNode API
    - BinaryNode 케이스 클래스
        - BinaryNode val 필드
        - BinaryNode 정의 메서드
- AWS Glue Scala BooleanNode API
    - BooleanNode 케이스 클래스
        - BooleanNode val 필드
        - BooleanNode 정의 메서드
- AWS Glue Scala ByteNode API
    - ByteNode 케이스 클래스

        - ByteNode val 필드
        - ByteNode 정의 메서드
- AWS Glue Scala DateNode API
    - DateNode 케이스 클래스
        - DateNode val 필드
        - ByteNode def 메서드
- AWS Glue Scala DecimalNode API
    - DecimalNode 케이스 클래스
        - DecimalNode val 필드
        - DecimalNode def 메서드
- AWS Glue Scala DoubleNode API
    - DoubleNode 케이스 클래스
        - DoubleNode val 필드
        - DoubleNode def 메서드
- AWS Glue Scala DynamicNode API
    - DynamicNode 클래스
        - DynamicNode def 메서드
    - DynamicNode 객체
        - DynamicNode def 메서드
- EvaluateDataQuality 클래스
    - def apply
    - 예
- AWS Glue Scala FloatNode API
    - FloatNode 케이스 클래스
        - FloatNode val 필드
        - FloatNode def 메서드
- FillMissingValues 클래스
    - def apply
- FindMatches 클래스
    - def apply

- FindIncrementalMatches 클래스
  - def apply
- AWS Glue Scala IntegerNode API
  - IntegerNode 케이스 클래스
    - IntegerNode val 필드
    - IntegerNode def 메서드
- AWS Glue Scala LongNode API
  - LongNode 케이스 클래스
    - LongNode val 필드
    - LongNode def 메서드
- AWS Glue Scala MapLikeNode API
  - MapLikeNode 클래스
    - MapLikeNode def 메서드
- AWS Glue Scala MapNode API
  - MapNode 케이스 클래스
    - MapNode def 메서드
- AWS Glue Scala NullNode API
  - NullNode 클래스
  - NullNode 케이스 객체
- AWS Glue Scala ObjectNode API
  - ObjectNode 객체
    - ObjectNode def 메서드
  - ObjectNode 케이스 클래스
    - ObjectNode def 메서드
- AWS Glue Scala ScalarNode API
  - ScalarNode 클래스
    - ScalarNode def 메서드
  - ScalarNode 객체
    - ScalarNode def 메서드

- AWS Glue Scala ShortNode API

    - ShortNode 케이스 클래스
        - ShortNode val 필드
        - ShortNode def 메서드
- AWS Glue Scala StringNode API
    - StringNode 케이스 클래스
        - StringNode val 필드
        - StringNode def 메서드
- AWS Glue Scala TimestampNode API
    - TimestampNode 케이스 클래스
        - TimestampNode val 필드
        - TimestampNode def 메서드
- AWS Glue Scala GlueArgParser API
    - GlueArgParser 객체
        - GlueArgParser def 메서드
- AWS Glue Scala 작업 API
    - 작업 객체
        - 작업 def 메서드

Scala 사용하여 AWS Glue ETL 스크립트 프로그래밍

AWS Glue 콘솔을 사용하여 Scala ETL(extract, transform, load: 추출, 변환, 로드) 프로그램을 자동으로 생성하고, 필요에 따라 수정한 후 작업에 할당할 수 있습니다. 또는 처음부터 자체 프로그램을 작성할 수 있습니다. 자세한 내용은 AWS Glue에서 Spark 작업에 대한 작업 속성 구성를 참조하십시오. 그런 다음 AWS Glue는 관련 작업을 실행하기 전에 Scala 프로그램을 서버에 포함합니다.

프로그램이 오류없이 포함되고 예상대로 실행되는지 확인합니다. 이런 과정은 읽기, 평가, 출력 루프(REPL) 혹은 Jupyter Notebook의 개발 엔드포인트에서 프로그램을 로드하고 작업에서 실행하기 전에 프로그램을 테스트할 때 중요합니다. 서버에서 포함하는 과정이 발생하기 때문에 여기에서 진행되는 어떠한 문제도 눈으로 볼 수 없습니다.

Scala ETL 프로그램을 개발 엔드포인트의 Jupyter Notebook에서 테스트

AWS Glue 개발 엔드포인트에서 Scala 프로그램을 테스트하려면 개발 엔드포인트 추가에 나오는 설명대로 개발 엔드포인트를 설정합니다.

그런 다음, 사용자의 시스템에서 로컬로 실행되거나 Amazon EC2 노트북 서버에서 원격으로 실행되는 Jupyter Notebook에 연결합니다. 자습서: JupyterLab의 Jupyter Notebook의 지시에 따라 Jupyter Notebook의 로컬 버전을 설치합니다.

노트북의 Scala 코드를 실행하는 것과 PySpark 코드를 실행하는 것의 단 한 가지 차이는 다음을 통해 Notebook의 각 단락을 시작해야 합니다.

```
%spark
```

이런 과정은 노트북 서버가 Spark 인터프리터의 PySpark에 맞게 초기화되는 것을 방지합니다.

Scala REPL에서 Scala ETL 프로그램 테스트

AWS Glue Scala REPL을 사용하여 개발 엔드포인트의 Scala 프로그램을 테스트할 수 있습니다. 자습서: SageMaker AI 노트북 사용 지침에 따라 SSH-to-REPL 명령 끝부분을 제외하고 `-t gluepyspark`를 `-t glue-spark-shell`로 바꿉니다. 그러면 AWS Glue Scala REPL이 호출됩니다.

완료되면 REPL을 닫아서 `sys.exit`을 입력합니다.

Scala 스크립트 예 - 스트리밍 ETL

Example

다음 예제 스크립트는 Amazon Kinesis Data Streams에 연결하고 Data Catalog의 스키마를 사용하여 데이터 스트림을 구문 분석하고, Amazon S3의 정적 데이터 집합에 스트림을 조인하고, 조인된 결과를 parquet 포맷의 Amazon S3에 출력합니다.

```
// This script connects to an Amazon Kinesis stream, uses a schema from the data
 catalog to parse the stream,
// joins the stream to a static dataset on Amazon S3, and outputs the joined results to
 Amazon S3 in parquet format.
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import java.util.Calendar
import org.apache.spark.SparkContext
import org.apache.spark.sql.Dataset
import org.apache.spark.sql.Row
import org.apache.spark.sql.SaveMode
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.from_json
```

```
import org.apache.spark.sql.streaming.Trigger
import scala.collection.JavaConverters._

object streamJoiner {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val sparkSession: SparkSession = glueContext.getSparkSession
    import sparkSession.implicits._
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val staticData = sparkSession.read          // read() returns type DataFrameReader
      .format("csv")
      .option("header", "true")
      .load("s3://amzn-s3-demo-bucket/inputs/productsStatic.csv")  // load() returns a
 DataFrame

    val datasource0 = sparkSession.readStream   // readstream() returns type
 DataStreamReader
      .format("kinesis")
      .option("streamName", "stream-join-demo")
      .option("endpointUrl", "https://kinesis.us-east-1.amazonaws.com")
      .option("startingPosition", "TRIM_HORIZON")
      .load                                     // load() returns a DataFrame

    val selectfields1 = datasource0.select(from_json($"data".cast("string"),
 glueContext.getCatalogSchemaAsSparkSchema("stream-demos", "stream-join-demo2")) as
 "data").select("data.*")

    val datasink2 = selectfields1.writeStream.foreachBatch { (dataFrame: Dataset[Row],
 batchId: Long) => {   //foreachBatch() returns type DataStreamWriter
      val joined = dataFrame.join(staticData, "product_id")
      val year: Int = Calendar.getInstance().get(Calendar.YEAR)
      val month :Int = Calendar.getInstance().get(Calendar.MONTH) + 1
      val day: Int = Calendar.getInstance().get(Calendar.DATE)
      val hour: Int = Calendar.getInstance().get(Calendar.HOUR_OF_DAY)

      if (dataFrame.count() > 0) {
        joined.write                            // joined.write returns type
 DataFrameWriter
          .mode(SaveMode.Append)
          .format("parquet")
```

```
          .option("quote", " ")
          .save("s3://amzn-s3-demo-bucket/output/" + "/year=" + "%04d".format(year)
 + "/month=" + "%02d".format(month) + "/day=" + "%02d".format(day) + "/hour=" +
 "%02d".format(hour) + "/")
      }
    }
    }  // end foreachBatch()
      .trigger(Trigger.ProcessingTime("100 seconds"))
      .option("checkpointLocation", "s3://amzn-s3-demo-bucket/checkpoint/")
      .start().awaitTermination()              // start() returns type StreamingQuery
    Job.commit()
  }
}
```

AWS Glue Scala 라이브러리의 API

AWS Glue는 추출, 변환 및 로드(ETL) 작업 스크립트의 PySpark Scala의 확장 언어를 지원합니다. 다음 단원에서는 AWS Glue Scala 라이브러리의 API를 설명합니다.

com.amazonaws.services.glue

AWS Glue Scala 라이브러리의 com.amazonaws.services.glue 패키지에는 다음 API가 포함되어 있습니다.

- ChoiceOption
- DataSink
- DataSource 특성
- DynamicFrame
- DynamicRecord
- GlueContext
- MappingSpec
- ResolveSpec

com.amazonaws.services.glue.ml

AWS Glue Scala 라이브러리의 com.amazonaws.services.glue.ml 패키지에는 다음 API가 포함되어 있습니다.

- FillMissingValues

- FindIncrementalMatches
- FindMatches

com.amazonaws.services.glue.dq

AWS Glue Scala 라이브러리의 com.amazonaws.services.glue.dq 패키지에는 다음 API가 포함되어 있습니다.

- EvaluateDataQuality

com.amazonaws.services.glue.types

AWS Glue Scala 라이브러리의 com.amazonaws.services.glue.types 패키지에는 다음 API가 포함되어 있습니다.

- ArrayNode
- BinaryNode
- BooleanNode
- ByteNode
- DateNode
- DecimalNode
- DoubleNode
- DynamicNode
- FloatNode
- IntegerNode
- LongNode
- MapLikeNode
- MapNode
- NullNode
- ObjectNode
- ScalarNode
- ShortNode
- StringNode

- TimestampNode

com.amazonaws.services.glue.util

AWS Glue Scala 라이브러리의 com.amazonaws.services.glue.util 패키지에는 다음 API가 포함되어 있습니다.

- GlueArgParser
- 작업

AWS Glue Scala ChoiceOption API

주제

- ChoiceOption 특성
- ChoiceOption 객체
- 케이스 클래스 ChoiceOptionWithResolver
- 케이스 클래스 MatchCatalogSchemaChoiceOption

패키지: com.amazonaws.services.glue

ChoiceOption 특성

```
trait ChoiceOption extends Serializable
```

ChoiceOption 객체

ChoiceOption

```
object ChoiceOption
```

DynamicFrame의 모든 ChoiceType 노드에 적용 가능한 선택을 확인하기 위한 일반적인 전략입니다.

- `val CAST`
- `val MAKE_COLS`
- `val MAKE_STRUCT`

- `val MATCH_CATALOG`
- `val PROJECT`

정의 적용

```
def apply(choice: String): ChoiceOption
```

케이스 클래스 ChoiceOptionWithResolver

```
case class ChoiceOptionWithResolver(name: String, choiceResolver: ChoiceResolver)
 extends ChoiceOption {}
```

케이스 클래스 MatchCatalogSchemaChoiceOption

```
case class MatchCatalogSchemaChoiceOption() extends ChoiceOption {}
```

추상 DataSink 클래스

주제

- writeDynamicFrame 정의
- pyWriteDynamicFrame 정의
- Def writeDataFrame
- Def pyWriteDataFrame
- setCatalogInfo 정의
- supportsFormat 정의
- setFormat 정의
- withFormat 정의
- setAccumulableSize 정의
- getOutputErrorRecordsAccumulable 정의
- errorsAsDynamicFrame 정의
- DataSink 객체

패키지: com.amazonaws.services.glue

```
abstract class DataSink
```

DataSource에 대한 쓰기 아날로그. DataSink는 DynamicFrame이 기록될 수 있는 대상과 형식을 요약합니다.

writeDynamicFrame 정의

```
def writeDynamicFrame( frame : DynamicFrame,
                       callSite : CallSite = CallSite("Not provided", "")
                     ) : DynamicFrame
```

pyWriteDynamicFrame 정의

```
def pyWriteDynamicFrame( frame : DynamicFrame,
                         site : String = "Not provided",
                         info : String = "" )
```

Def writeDataFrame

```
def writeDataFrame(frame: DataFrame,
                   glueContext: GlueContext,
                   callSite: CallSite = CallSite("Not provided", "")
                   ): DataFrame
```

Def pyWriteDataFrame

```
def pyWriteDataFrame(frame: DataFrame,
                     glueContext: GlueContext,
                     site: String = "Not provided",
                     info: String = ""
                     ): DataFrame
```

setCatalogInfo 정의

```
def setCatalogInfo(catalogDatabase: String,
```

```
                    catalogTableName : String,
                    catalogId : String = "")
```

supportsFormat 정의

```
def supportsFormat( format : String ) : Boolean
```

setFormat 정의

```
def setFormat( format : String,
               options : JsonOptions
             ) : Unit
```

withFormat 정의

```
def withFormat( format : String,
                options : JsonOptions = JsonOptions.empty
              ) : DataSink
```

setAccumulableSize 정의

```
def setAccumulableSize( size : Int ) : Unit
```

getOutputErrorRecordsAccumulable 정의

```
def getOutputErrorRecordsAccumulable : Accumulable[List[OutputError], OutputError]
```

errorsAsDynamicFrame 정의

```
def errorsAsDynamicFrame : DynamicFrame
```

DataSink 객체

```
object DataSink
```

recordMetrics 정의

```
def recordMetrics( frame : DynamicFrame,
                   ctxt : String
                 ) : DynamicFrame
```

AWS Glue Scala DataSource 특성

패키지: com.amazonaws.services.glue

DynamicFrame을 생산하는데 필요한 높은 수준의 인터페이스입니다.

```
trait DataSource {

  def getDynamicFrame : DynamicFrame

  def getDynamicFrame( minPartitions : Int,
                       targetPartitions : Int
                     ) : DynamicFrame
  def getDataFrame : DataFrame

  /** @param num: the number of records for sampling.
    * @param options: optional parameters to control sampling behavior. Current
 available parameter for Amazon S3 sources in options:
    *  1. maxSamplePartitions: the maximum number of partitions the sampling will
 read.
    *  2. maxSampleFilesPerPartition: the maximum number of files the sampling will
 read in one partition.
    */
  def getSampleDynamicFrame(num:Int, options: JsonOptions = JsonOptions.empty):
 DynamicFrame

  def glueContext : GlueContext

  def setFormat( format : String,
                 options : String
               ) : Unit

  def setFormat( format : String,
                 options : JsonOptions
               ) : Unit
```

```
  def supportsFormat( format : String ) : Boolean

  def withFormat( format : String,
                  options : JsonOptions = JsonOptions.empty
                ) : DataSource
}
```

AWS Glue Scala DynamicFrame API

패키지: com.amazonaws.services.glue

목차

- AWS Glue Scala DynamicFrame 클래스
  - val errorsCount
  - def applyMapping
  - def assertErrorThreshold
  - def count
  - def dropField
  - def dropFields
  - def dropNulls
  - def errorsAsDynamicFrame
  - def filter
  - def getName
  - def getNumPartitions
  - def getSchemaIfComputed
  - def isSchemaComputed
  - def javaToPython
  - def join
  - def map
  - def mergeDynamicFrames
  - def printSchema
  - def recomputeSchema
  - def relationalize
  - def renameField

- def repartition
- def resolveChoice
- def schema
- def selectField
- def selectFields
- def show
- Def simplifyDDBJson
- def spigot
- def splitFields
- def splitRows
- def stageErrorsCount
- def toDF
- def unbox
- def unnest
- def unnestDDBJson
- def withFrameSchema
- def withName
- def withTransformationContext

- DynamicFrame 객체
  - def apply
  - def emptyDynamicFrame
  - def fromPythonRDD
  - def ignoreErrors
  - def inlineErrors
  - def newFrameWithErrors

AWS Glue Scala DynamicFrame 클래스

패키지: com.amazonaws.services.glue

```
class DynamicFrame extends Serializable with Logging  (
        val glueContext : GlueContext,
```

```
            _records : RDD[DynamicRecord],
            val name : String = s"",
            val transformationContext : String = DynamicFrame.UNDEFINED,
            callSite : CallSite = CallSite("Not provided", ""),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0,
            prevErrors : => Long = 0,
            errorExpr : => Unit = {} )
```

DynamicFrame은 자기 기술형 DynamicRecord 객체의 분산형 모음입니다.

DynamicFrame은 ETL(extract, transform, load) 작업에 유연한 데이터 모델을 제공할 수 있게끔 설계되어 있습니다. 스키마를 생성할 필요가 없고, 복잡하거나 일관되지 않은 값과 유형을 가진 데이터를 읽고 변환할 때 사용할 수 있습니다. 스키마가 필요한 작업에만 '온 디맨드(필요에 따라)'로 스키마를 계산할 수 있습니다.

DynamicFrame은 데이터 정리 및 ETL에 필요한 다양한 변환을 지원합니다. 또한 DataFrame이 제공하는 많은 분석 작업과 기존 코드를 통합할 수 있도록 SparkSQL DataFrame 변환을 지원합니다.

다음 파라미터는 DynamicFrame을 구성하는 수많은 AWS Glue 변환에서 공유됩니다.

- transformationContext - 이 DynamicFrame의 식별자입니다. transformationContext는 실행 간 지속되는 작업 북마크 상태에 대한 키로 사용됩니다.
- callSite - 오류 보고에 필요한 컨텍스트 정보를 제공합니다. Python에서 호출 할 때 자동으로 이 값이 설정됩니다.
- stageThreshold - 예외가 발생하기 전에 이 DynamicFrame 계산에 허용되는 오류 레코드의 최대 개수입니다(이전 DynamicFrame에 존재하는 레코드 제외).
- totalThreshold - 예외가 발생하기 전 전체 오류 레코드의 최대 개수입니다(이전 프레임의 레코드 포함).

val errorsCount

```
val errorsCount
```

이 DynamicFrame의 오류 레코드 개수입니다. 여기에는 이전 작업의 오류가 포함됩니다.

def applyMapping

```
def applyMapping( mappings : Seq[Product4[String, String, String, String]],
```

```
                  caseSensitive : Boolean = true,
                  transformationContext : String = "",
                  callSite : CallSite = CallSite("Not provided", ""),
                  stageThreshold : Long = 0,
                  totalThreshold : Long = 0
                ) : DynamicFrame
```

- `mappings` - 새 `DynamicFrame`을 구성하는 매핑 시퀀스입니다.
- `caseSensitive` - 소스 열을 대소문자를 구분해서 처리할지 여부입니다. 이것을 false로 설정하면 AWS Glue Data Catalog 같이 대/소문자를 구분하지 않는 스토어 통합에 도움이 됩니다.

매핑 시퀀스에 따라 열을 선택, 프로젝트, 캐스트 합니다.

각 매핑은 소스 열과 유형 및 대상 열과 유형으로 구성되어 있습니다. 매핑을 4개 튜플(`source_path`, `source_type`, `target_path`, `target_type`) 또는 동일한 정보가 있는 MappingSpec 객체로 지정할 수 있습니다.

간단한 프로젝션과 캐스팅에 매핑을 사용하는 것 외에 '.'(마침표)가 있는 경로의 구성 요소를 분리해 필드를 중첩하거나 중첩을 해제할 때도 매핑을 사용할 수 있습니다.

예를 들어 `DynamicFrame`에 다음 스키마가 있다고 가정해보겠습니다.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- zip: int
}}}
```

다음 호출을 통해 `state` 및 `zip` 필드의 중첩을 해제할 수 있습니다.

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
       ("address.state", "string", "state", "string"),
       ("address.zip", "int", "zip", "int")))
}}}
```

결과 스키마는 다음과 같습니다.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- state: string
  |-- zip: int
}}}
```

또한 applyMapping을 사용하여 열을 다시 중첩시킬 수 있습니다. 예를 들어 다음은 이전 변환을 반전시킨 다음에 대상에 address라는 이름의 구조체를 생성합니다.

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
       ("state", "string", "address.state", "string"),
       ("zip", "int", "address.zip", "int")))
}}}
```

'.'(마침표) 문자가 있는 필드 이름은 백틱(``)을 인용 부호로 사용할 수 있습니다.

> **Note**
>
> 현재는 applyMapping 방법을 사용하여 어레이 아래에 중첩된 열을 매핑할 수 없습니다.

def assertErrorThreshold

```
def assertErrorThreshold : Unit
```

계산을 강제 실행하고 오류 레코드의 개수가 stageThreshold 및 totalThreshold 미만인지 확인하는 작업입니다. 두 조건 중 하나가 충족되지 않는 경우 예외가 발생합니다.

def count

```
lazy
def count
```

이 DynamicFrame의 요소 개수를 반환합니다.

def dropField

```
def dropField( path : String,
               transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0
             ) : DynamicFrame
```

지정한 열이 제거된 새 DynamicFrame을 반환합니다.

def dropFields

```
def dropFields( fieldNames : Seq[String],   // The column names to drop.
                transformationContext : String = "",
                callSite : CallSite = CallSite("Not provided", ""),
                stageThreshold : Long = 0,
                totalThreshold : Long = 0
              ) : DynamicFrame
```

지정한 열이 제거된 새 DynamicFrame을 반환합니다.

이 방법을 사용하여 어레이 내부의 열을 포함해 중첩된 열을 삭제할 수 있지만, 지정 어레이 요소를 없앨 수는 없습니다.

def dropNulls

```
def dropNulls( transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0 )
```

null 열이 모두 제거된 새 DynamicFrame을 반환합니다.

> **Note**
>
> NullType 유형의 열만 제거합니다. 다른 열의 개별 null 값은 제거되거나 수정되지 않습니다.

def errorsAsDynamicFrame

```
def errorsAsDynamicFrame
```

이 DynamicFrame의 오류 레코드를 포함하는 새 DynamicFrame을 반환합니다.

def filter

```
def filter( f : DynamicRecord => Boolean,
            errorMsg : String = "",
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

함수 'f'이 true를 반환하는 레코드만 포함하는 새로운 DynamicFrame을 생성합니다. 필터 함수 'f'는 입력 레코드를 바꾸지 않습니다.

def getName

```
def getName : String
```

이 DynamicFrame의 이름을 반환합니다.

def getNumPartitions

```
def getNumPartitions
```

이 DynamicFrame의 파티션 개수를 반환합니다.

def getSchemaIfComputed

```
def getSchemaIfComputed : Option[Schema]
```

계산이 완료된 스키마를 반환합니다. 스키마 계산이 완료되지 않은 경우 데이터를 스캔하지 않습니다.

def isSchemaComputed

```
def isSchemaComputed : Boolean
```

이 DynamicFrame에 대해 스키마가 계산되었으면 true를 반환하고 그렇지 않으면 false를 반환합니다. 이 방법이 false를 반환할 경우 schema 방법을 호출하려면 이 DynamicFrame의 레코드를 한 번 더 전달해야 합니다.

def javaToPython

```
def javaToPython : JavaRDD[Array[Byte]]
```

def join

```
def join( keys1 : Seq[String],
          keys2 : Seq[String],
          frame2 : DynamicFrame,
          transformationContext : String = "",
          callSite : CallSite = CallSite("Not provided", ""),
          stageThreshold : Long = 0,
          totalThreshold : Long = 0
        ) : DynamicFrame
```

- keys1 - 이 DynamicFrame에서 조인에 사용할 수 있는 열입니다.
- keys2 - frame2에서 조인에 사용할 수 있는 열입니다. keys1과 길이가 같아야 합니다.
- frame2 - DynamicFrame 조인 기준입니다.

지정된 키를 사용하여 frame2로 동등 조인을 수행한 결과를 반환합니다.

def map

```
def map( f : DynamicRecord => DynamicRecord,
         errorMsg : String = "",
         transformationContext : String = "",
         callSite : CallSite = CallSite("Not provided", ""),
         stageThreshold : Long = 0,
         totalThreshold : Long = 0
       ) : DynamicFrame
```

이 DynamicFrame의 각 레코드에 지정된 함수 'f'를 적용하여 생성된 새로운 DynamicFrame을 반환합니다.

이 방법은 레코드가 바뀌지 않도록 각 레코드를 복사한 후에 지정된 함수를 적용합니다. 매핑 함수가 특정 레코드에 대해 예외를 발생시킨 경우, 이 레코드를 오류로 표시하고 스택 트레이스를 오류 레코드의 열로 저장합니다.

def mergeDynamicFrames

```
def mergeDynamicFrames( stageDynamicFrame: DynamicFrame,  primaryKeys: Seq[String],
 transformationContext: String = "",
                         options: JsonOptions = JsonOptions.empty, callSite: CallSite =
 CallSite("Not provided"),
                         stageThreshold: Long = 0, totalThreshold: Long = 0):
 DynamicFrame
```

- `stageDynamicFrame` - 병합할 스테이징 `DynamicFrame`입니다.
- `primaryKeys` - 소스 및 스테이징 `DynamicFrame`의 레코드와 일치시킬 기본 키 필드 목록입니다.
- `transformationContext` - 현재 변환에 대한 메타데이터를 검색하는 데 사용되는 고유한 문자열입니다(선택 사항).
- `options` - 이 변환에 대한 추가 정보를 제공하는 JSON 이름-값 페어 문자열입니다.
- `callSite` - 오류 보고에 필요한 컨텍스트 정보를 제공하는 데 사용합니다.
- `stageThreshold` - `Long`입니다. 오류가 발생되는 지정된 변환의 오류 수입니다.
- `totalThreshold` - `Long`입니다. 오류가 발생되는 이 변환의 총 오류 수입니다.

레코드를 식별하기 위해, 지정된 기본 키를 기반으로 이 `DynamicFrame`을 스테이징 `DynamicFrame`과 병합합니다. 중복 레코드(기본 키가 동일한 레코드)는 중복 제거되지 않습니다. 스테이징 프레임에 일치하는 레코드가 없으면 모든 레코드(중복 레코드 포함)가 소스에서 유지됩니다. 스테이징 프레임에 일치하는 레코드가 있으면 스테이징 프레임의 레코드가 AWS Glue의 소스에 있는 레코드를 덮어씁니다.

다음과 같은 경우 반환된 `DynamicFrame`에 레코드 A가 포함되어 있습니다.

1. A가 소스 프레임과 스테이징 프레임 모두에 있는 경우에는 스테이징 프레임의 A가 반환됩니다.
2. A가 소스 테이블에 있고 `A.primaryKeys`가 `stagingDynamicFrame`에 없는 경우에는 A는 스테이징 테이블에서 업데이트되지 않습니다.

소스 프레임과 스테이징 프레임이 동일한 스키마를 가질 필요는 없습니다.

Example

```
val mergedFrame: DynamicFrame = srcFrame.mergeDynamicFrames(stageFrame, Seq("id1",
 "id2"))
```

def printSchema

```
def printSchema : Unit
```

이 DynamicFrame의 스키마를 사람이 읽을 수 있는 형식의 stdout으로 인쇄합니다.

def recomputeSchema

```
def recomputeSchema : Schema
```

스키마 재계산을 강제 실행합니다. 이를 위해 데이터를 스캔해야 하지만, 현재 스키마의 일부 필드가 데이터에 존재하지 않는 경우에는 스키마를 줄여야 할 수도 있습니다.

재계산한 스키마를 반환합니다.

def relationalize

```
def relationalize( rootTableName : String,
                   stagingPath : String,
                   options : JsonOptions = JsonOptions.empty,
                   transformationContext : String = "",
                   callSite : CallSite = CallSite("Not provided"),
                   stageThreshold : Long = 0,
                   totalThreshold : Long = 0
                 ) : Seq[DynamicFrame]
```

- rootTableName - 출력에서 기본 DynamicFrame에 사용하는 이름. 이를 접두사로 시작하는 어레이를 피벗하여 생성되는 DynamicFrame입니다.
- stagingPath - 중간 데이터 쓰기를 위한 Amazon Simple Storage Service(Amazon S3) 경로입니다.
- options - 관계화 옵션 및 구성입니다. 현재 사용되지 않습니다.

모든 중첩된 구조와 피벗 어레이를 별개 테이블로 평면화합니다.

이 작업을 사용하여 크게 중첩된 데이터를 관계형 데이터베이스가 처리하도록 준비할 수 있습니다. 중첩된 구조를 unnest 변환과 동일한 방식으로 평면화합니다. 여기에 더해, 어레이를 각 어레이 요소가 행이 되는 별개 테이블로 피벗합니다. 예를 들어 DynamicFrame에 다음 데이터가 있다고 가정해보겠습니다.

```
{"name": "Nancy", "age": 47, "friends": ["Fred", "Lakshmi"]}
{"name": "Stephanie", "age": 28, "friends": ["Yao", "Phil", "Alvin"]}
{"name": "Nathan", "age": 54, "friends": ["Nicolai", "Karen"]}
```

다음 코드를 실행합니다.

```
{{{
  df.relationalize("people", "s3:/my_bucket/my_path", JsonOptions.empty)
}}}
```

그러면 두 개 테이블이 생성됩니다. 첫 번째 테이블 이름은 "people"이며, 다음이 포함되어 있습니다.

```
{{{
  {"name": "Nancy", "age": 47, "friends": 1}
  {"name": "Stephanie", "age": 28, "friends": 2}
  {"name": "Nathan", "age": 54, "friends": 3)
}}}
```

여기에서는 친구 어레이가 자동 생성된 조인 키로 교체되어 있습니다. 다음 내용이 포함된 people.friends라는 이름의 별개 테이블이 생성됩니다.

```
{{{
  {"id": 1, "index": 0, "val": "Fred"}
  {"id": 1, "index": 1, "val": "Lakshmi"}
  {"id": 2, "index": 0, "val": "Yao"}
  {"id": 2, "index": 1, "val": "Phil"}
  {"id": 2, "index": 2, "val": "Alvin"}
  {"id": 3, "index": 0, "val": "Nicolai"}
  {"id": 3, "index": 1, "val": "Karen"}
}}}
```

이 테이블에서 'id'는 어레이 요소의 출처인 레코드를 식별하는 조인 키이며, 'index'는 원본 어레이에서 그 위치를 나타내고, 'val'는 실제 어레이 항목입니다.

relationalize 방법은 이 프로세스를 모든 어레이에 반복 적용하여 생성되는 DynamicFrame의 시퀀스를 반환합니다.

> **Note**
>
> AWS Glue 라이브러리는 자동으로 새 테이블에 대한 조인 키를 생성합니다. 작업 실행 간 조인 키가 고유하도록 만들려면 작업 북마크를 활성화해야 합니다.

def renameField

```
def renameField( oldName : String,
                 newName : String,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

- `oldName` - 열의 원래 이름입니다.
- `newName` - 열의 새 이름입니다.

이름이 바뀐 지정 필드로 새 `DynamicFrame`을 반환합니다.

이 방법을 사용하여 중첩된 필드의 이름을 바꿀 수 있습니다. 예를 들어 주소 구조체의 `state`라는 이름을 `state_code`로 바꿉니다.

```
{{{
  df.renameField("address.state", "address.state_code")
}}}
```

def repartition

```
def repartition( numPartitions : Int,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

`numPartitions` 파티션이 있는 새 `DynamicFrame`을 반환합니다.

def resolveChoice

```
def resolveChoice( specs : Seq[Product2[String, String]] = Seq.empty[ResolveSpec],
                   choiceOption : Option[ChoiceOption] = None,
                   database : Option[String] = None,
                   tableName : Option[String] = None,
                   transformationContext : String = "",
                   callSite : CallSite = CallSite("Not provided", ""),
                   stageThreshold : Long = 0,
                   totalThreshold : Long = 0
                 ) : DynamicFrame
```

- choiceOption - 사양 시퀀스에 열거되지 않는 모든 ChoiceType 열에 적용되는 작업입니다.
- database - match_catalog 작업에 사용할 Data Catalog 데이터베이스입니다.
- tableName - match_catalog 작업에 사용할 Data Catalog 테이블입니다.

하나 이상의 ChoiceType을 보다 구체적인 유형으로 대체하여 새로운 DynamicFrame을 반환합니다.

resolveChoice를 사용하는 방법은 두 가지입니다. 첫 번째는 지정 열의 시퀀스와 확인 방법을 지정하는 방법입니다. (열과 작업)쌍으로 구성된 튜플로 지정합니다.

가능한 작업은 다음과 같습니다.

- cast:type - 모든 값을 지정된 유형으로 캐스트하는 시도를 합니다.
- make_cols - 각 유형을 columnName_type이라는 이름을 가진 열로 변환합니다.
- make_struct - 열을 각 유형에 대한 키가 있는 구조체로 변환합니다.
- project:type - 지정된 유형의 값만 유지합니다.

resolveChoice의 또 다른 모드는 모든 ChoiceType에 대해 단 하나의 해결책을 지정하는 것입니다. 실행 전에 ChoiceType의 완전한 목록을 모르는 경우에 사용할 수 있습니다. 이 모드는 위에서 나열한 작업 외에 다음 작업을 지원합니다.

- match_catalog - 각 ChoiceType을 지정된 카탈로그 테이블의 해당 유형에 캐스팅해봅니다.

예:

int로 캐스팅을 해서 user.id 열을 확인하고, address 필드에는 구조체만 유지합니다.

```
{{{
  df.resolveChoice(specs = Seq(("user.id", "cast:int"), ("address", "project:struct")))
}}}
```

각 선택을 개별 열로 변환하여 전체 ChoiceType을 확인합니다.

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("make_cols")))
}}}
```

지정된 카탈로그 테이블의 유형으로 캐스팅하여 전체 ChoiceType을 확인합니다.

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("match_catalog")),
                   database = Some("my_database"),
                   tableName = Some("my_table"))
}}}
```

def schema

```
def schema : Schema
```

이 DynamicFrame의 스키마를 반환합니다.

반환되는 스키마에는 이 DynamicFrame의 레코드에 있는 모든 필드가 있습니다. 단, 많지는 않지만 추가 필드가 포함되는 경우도 있을 수 있습니다. unnest 방법을 사용하여 이 DynamicFrame의 레코드에 따라 스키마를 "줄일" 수 있습니다.

def selectField

```
def selectField( fieldName : String,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

단일 필드를 DynamicFrame으로 반환합니다.

def selectFields

```
def selectFields( paths : Seq[String],
                  transformationContext : String = "",
                  callSite : CallSite = CallSite("Not provided", ""),
                  stageThreshold : Long = 0,
                  totalThreshold : Long = 0
                ) : DynamicFrame
```

- `paths` - 선택할 열 이름의 시퀀스입니다.

지정한 열이 포함된 새 `DynamicFrame`을 반환합니다.

> **Note**
>
> 최상위 열에만 `selectFields` 방법을 사용할 수 있습니다. applyMapping 방법을 사용하여 중첩된 열을 선택할 수 있습니다.

def show

```
def show( numRows : Int = 20 ) : Unit
```

- `numRows` - 인쇄할 행의 개수입니다.

이 `DynamicFrame`의 행을 JSON 형식으로 인쇄합니다.

Def simplifyDDBJson

AWS Glue DynamoDB 내보내기 커넥터를 사용하여 DynamoDB 내보내기를 수행하면 특정 중첩 구조의 JSON 파일이 생성됩니다. 자세한 내용은 데이터 객체를 참조하세요. `simplifyDDBJson` 이러한 유형의 데이터에 대한 DynamicFrame의 중첩 열을 단순화하고 단순화된 새로운 DynamicFrame을 반환합니다. 목록 유형에 여러 유형이 있거나 맵 유형이 포함된 경우 목록의 요소는 단순화되지 않습니다. 이 메서드는 DynamoDB 내보내기 JSON 형식의 데이터만 지원합니다. 다른 종류의 데이터에 대해 유사한 변경을 수행하려면 `unnest`를 고려하세요.

```
def simplifyDDBJson() : DynamicFrame
```

이 메서드에서는 파라미터를 사용하지 않습니다.

입력 예

DynamoDB 내보내기로 생성된 다음 스키마를 고려해 보세요.

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

예제 코드

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContextimport scala.collection.JavaConverters._

object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
```

```
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> "ddbTableARN",
        "dynamodb.s3.bucket" -> "exportBucketLocation",
        "dynamodb.s3.prefix" -> "exportBucketPrefix",
        "dynamodb.s3.bucketOwner" -> "exportBucketAccountID",
      ))
    ).getDynamicFrame()

    val simplified = dynamicFrame.simplifyDDBJson()
    simplified.printSchema()

    Job.commit()
  }

}
```

출력 예시

simplifyDDBJson 변환은 이를 다음과 같이 간소화합니다.

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

def spigot

```
def spigot( path : String,
            options : JsonOptions = new JsonOptions("{}"),
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

동일한 레코드를 반환하지만 부수적으로 레코드의 하위 집합을 작성하는 패스스루 변환입니다.

- `path` - `s3://bucket//path` 형식으로 출력을 쓸 Amazon S3의 경로입니다.
- `options` - 샘플링 동작을 설명하는 선택 사항인 `JsonOptions` 맵입니다.

이것과 동일한 레코드가 포함된 `DynamicFrame`을 반환합니다.

`path`로 지정한 위치에 임의의 레코드 100개를 작성하는 것이 기본값입니다. `options` 맵을 사용하여 이 동작을 사용자 지정할 수 있습니다. 유효한 키에는 다음이 포함되어 있습니다.

- `topk` - 작성할 레코드의 총 개수를 지정합니다. 기본값은 100입니다.
- `prob` - 개별 레코드의 포함 확률(십진수)을 지정합니다. 기본값은 1.

예를 들어 다음 호출은 20퍼센트의 확률로 각 레코드를 선택하고 200개의 레코드를 작성한 후에 중단하는 방법으로 데이터 세트를 샘플링합니다.

```
{{{
  df.spigot("s3://my_bucket/my_path", JsonOptions(Map("topk" -&gt; 200, "prob" -&gt;
 0.2)))
}}}
```

def splitFields

```
def splitFields( paths : Seq[String],
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : Seq[DynamicFrame]
```

- paths - 첫 DynamicFrame에 포함될 경로입니다.

두 개 DynamicFrame의 시퀀스를 반환합니다. 첫 번째 DynamicFrame에는 지정 경로가, 두 번째에는 나머지 열 전부가 포함됩니다.

예

이 예제에서는 AWS Glue Data Catalog의 legislators 데이터베이스에 있는 persons 테이블에서 생성된 DynamicFrame을 사용하고 두 개의 DynamicFrame으로 분할합니다. 지정된 필드는 첫 번째 DynamicFrame으로 이동하고, 나머지 필드는 두 번째 DynamicFrame으로 이동합니다. 그런 다음, 예제는 결과에서 첫 번째 DynamicFrame을 선택합니다.

```
val InputFrame = glueContext.getCatalogSource(database="legislators",
 tableName="persons",
transformationContext="InputFrame").getDynamicFrame()

val SplitField_collection = InputFrame.splitFields(paths=Seq("family_name", "name",
 "links.note",
"links.url", "gender", "image", "identifiers.scheme", "identifiers.identifier",
 "other_names.lang",
"other_names.note", "other_names.name"), transformationContext="SplitField_collection")

val ResultFrame = SplitField_collection(0)
```

def splitRows

```
def splitRows( paths : Seq[String],
               values : Seq[Any],
               operators : Seq[String],
               transformationContext : String,
               callSite : CallSite,
               stageThreshold : Long,
               totalThreshold : Long
             ) : Seq[DynamicFrame]
```

열을 상수에 비교하는 조건자에 따라 행을 분할합니다.

- paths - 비교에 사용할 열입니다.
- values - 비교에 사용할 상수 값입니다.
- operators - 비교에 사용할 연산자입니다.

두 개 DynamicFrame의 시퀀스를 반환합니다. 첫 번째에는 조건자가 true, 두 번째에는 조건자가 false인 행이 포함됩니다.

조건자는 다음 세 개의 시퀀스를 사용하여 지정됩니다. 'paths'에는 (중첩될 수 있는) 열 이름이 포함되고, 'values'에는 비교할 상수 값이 포함되며, 'operators'에는 비교에 사용할 연산자가 포함됩니다. 세 가지 시퀀스는 모두 길이가 같아야 합니다. n번째 연산자를 사용하여 n번째 열과 n번째 값을 비교합니다.

각 연산자는 "!=", "=", "<=", "<", ">=" 또는 ">" 중 하나여야 합니다.

예를 들어 다음 호출은 첫 번째 출력 프레임에는 65세 이상의 미국인에 대한 레코드가 포함되고 두 번째에는 나머지 레코드 전부가 포함되도록 DynamicFrame을 분할합니다.

```
{{{
  df.splitRows(Seq("age", "address.country"), Seq(65, "USA"), Seq("&gt;=", "="))
}}}
```

def stageErrorsCount

```
def stageErrorsCount
```

이 DynamicFrame을 계산하는 동안 생성된 오류 레코드의 개수를 반환합니다. 이 DynamicFrame에 입력으로 전달된 이전 작업의 오류는 제외됩니다.

def toDF

```
def toDF( specs : Seq[ResolveSpec] = Seq.empty[ResolveSpec] ) : DataFrame
```

이 DynamicFrame을 스키마와 레코드가 동일한 Apache Spark SQL DataFrame으로 변환합니다.

> **Note**
>
> DataFrame은 ChoiceType을 지원하지 않으므로 이 방법은 ChoiceType 열을 StructType로 자동 변환합니다. 선택할 수 있는 확인(해결) 방법에 대한 자세한 내용은 resolveChoice 단원을 참조하십시오.

def unbox

```
def unbox( path : String,
```

```
            format : String,
            optionString : String = "{}",
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

- `path` - 구문 분석할 열입니다. 문자열 또는 이진수여야 합니다.
- `format` - 구문 분석에 사용할 포맷입니다.
- `optionString` - CSV 구분자 같은 포맷으로 전달되는 옵션입니다.

지정 형식에 따라 포함된 문자열이나 이진수 열을 구문 분석합니다. 구문 분석한 열은 원래 열 이름을 가진 구조체 아래 중첩됩니다.

예를 들어 JSON 열이 포함된 CSV 파일이 있다고 가정해보겠습니다.

```
name, age, address
Sally, 36, {"state": "NE", "city": "Omaha"}
...
```

최초 구문 분석 후에 다음 스키마를 갖는 `DynamicFrame`이 생깁니다.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: string
}}}
```

주소 열에 unbox를 호출해 지정 구성 요소를 구문 분석할 수 있습니다.

```
{{{
  df.unbox("address", "json")
}}}
```

그러면 다음 스키마를 갖는 `DynamicFrame`이 생깁니다.

```
{{{
  root
```

```
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- city: string
}}}
```

def unnest

```
def unnest( transformationContext : String = "",
            callSite : CallSite = CallSite("Not Provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

중첩된 구조가 모두 평면화된 새 DynamicFrame이 반환됩니다. 이름은 '.'(마침표) 문자를 사용하여 생성합니다.

예를 들어 DynamicFrame에 다음 스키마가 있다고 가정해보겠습니다.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- city: string
}}}
```

다음 호출은 주소 구조체의 중첩을 해제합니다.

```
{{{
  df.unnest()
}}}
```

결과 스키마는 다음과 같습니다.

```
{{{
  root
  |-- name: string
  |-- age: int
```

```
  |-- address.state: string
  |-- address.city: string
}}}
```

이 방법도 어레이 내부에 중첩된 구조체의 중첩을 해제합니다. 그러나 기록을 위해 이러한 필드는 포함되는 어레이의 이름과 ".val"이 이름 앞에 붙습니다.

def unnestDDBJson

```
unnestDDBJson(transformationContext : String = "",
            callSite : CallSite = CallSite("Not Provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0): DynamicFrame
```

DynamoDB JSON 구조의 DynamicFrame에 있는 중첩된 열을 한정해서 중첩 해제하고, 중첩되지 않은 새 DynamicFrame을 반환합니다. 구조체 유형 배열인 열은 중첩 해제되지 않습니다. 이 변환은 일반적인 unnest 변환과는 다르게 작동하는 특정 유형의 중첩 해제 변환이며, 데이터가 이미 DynamoDB JSON 구조에 있어야 합니다. 자세한 내용은 DynamoDB JSON을 참조하세요.

예를 들어, DynamoDB JSON 구조를 사용하여 내보내기를 읽는 스키마는 다음과 같을 수 있습니다.

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
|    |    |-- N: string
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

unnestDDBJson() 변환은 이 구조를 다음과 같이 변환합니다.

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

다음 코드 예제에서는 AWS Glue DynamoDB 내보내기 커넥터를 사용하고, DynamoDB JSON unnest를 호출하고, 파티션 수를 인쇄하는 방법을 보여줍니다.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> "<test_source>",
        "dynamodb.s3.bucket" -> "<bucket name>",
        "dynamodb.s3.prefix" -> "<bucket prefix>",
        "dynamodb.s3.bucketOwner" -> "<account_id of bucket>",
      ))
    ).getDynamicFrame()

    val unnested = dynamicFrame.unnestDDBJson()
    print(unnested.getNumPartitions())

    Job.commit()
  }

}
```

def withFrameSchema

```
def withFrameSchema( getSchema : () => Schema ) : DynamicFrame
```

- getSchema - 사용할 스키마를 반환하는 함수입니다. 0이라는 파라미터 함수로 지정을 하면 비용이 많이 드는 계산을 연기할 수 있습니다.

이 DynamicFrame의 스키마를 지정된 값으로 설정합니다. 비용이 많이 드는 스키마 재계산을 피하기 위해 주로 내부에서 사용합니다. 통과된 스키마에는 데이터에 존재하는 모든 열이 포함되어 있어야 합니다.

def withName

```
def withName( name : String ) : DynamicFrame
```

- name - 사용할 새 이름입니다.

새 이름이 있는 이 DynamicFrame의 복사본을 반환합니다.

def withTransformationContext

```
def withTransformationContext( ctx : String ) : DynamicFrame
```

지정된 변환 컨텍스트가 있는 이 DynamicFrame의 복사본을 반환합니다.

DynamicFrame 객체

패키지: com.amazonaws.services.glue

```
object DynamicFrame
```

def apply

```
def apply( df : DataFrame,
           glueContext : GlueContext
         ) : DynamicFrame
```

def emptyDynamicFrame

```
def emptyDynamicFrame( glueContext : GlueContext ) : DynamicFrame
```

def fromPythonRDD

```
def fromPythonRDD( rdd : JavaRDD[Array[Byte]],
                   glueContext : GlueContext
                 ) : DynamicFrame
```

def ignoreErrors

```
def ignoreErrors( fn : DynamicRecord => DynamicRecord ) : DynamicRecord
```

def inlineErrors

```
def inlineErrors( msg : String,
                  callSite : CallSite
                ) : (DynamicRecord => DynamicRecord)
```

def newFrameWithErrors

```
def newFrameWithErrors( prevFrame : DynamicFrame,
                        rdd : RDD[DynamicRecord],
                        name : String = "",
                        transformationContext : String = "",
                        callSite : CallSite,
                        stageThreshold : Long,
                        totalThreshold : Long
                      ) : DynamicFrame
```

AWS Glue Scala DynamicRecord 클래스

주제

- def addField
- def dropField
- def setError
- def isError
- def getError
- def clearError

- def write
- def readFields
- def clone
- def schema
- def getRoot
- def toJson
- def getFieldNode
- def getField
- def hashCode
- def equals
- DynamicRecord 객체
- RecordTraverser 특성

패키지: com.amazonaws.services.glue

```
class DynamicRecord extends Serializable with Writable with Cloneable
```

DynamicRecord는 진행된 데이터 세트의 데이터 행을 나타내는 자체적으로 설명되는 데이터 구조입니다. 기록 자체를 검사한 DynamicRecord에 의해 보여지는 스키마 행을 얻을 수 있다는 의미에서 자체 설명적입니다. DynamicRecord는 Apache Spark의 Row와 유사합니다.

def addField

```
def addField( path : String,
              dynamicNode : DynamicNode
            ) : Unit
```

지정된 경로로 DynamicNode를 추가합니다.

- path - 필드를 추가할 경로입니다.
- dynamicNode - 지정된 경로에 추가될 DynamicNode입니다.

def dropField

```
 def dropField(path: String, underRename: Boolean = false): Option[DynamicNode]
```

지정된 경로로부터 DynamicNode을 드롭하고 지정된 경로에 배열이 없으면 드롭된 노트를 반환합니다.

- path - 필드를 제거할 경로입니다.
- underRename - dropField가 이름 변경 변환의 일부로 호출되는 경우는 true이고, 아닌 경우는 false입니다(기본값은 false).

scala.Option Option(DynamicNode)을 반환합니다.

def setError

```
def setError( error : Error )
```

error 파라미터에 의해 지정되어 이 기록을 오류 기록으로 설정합니다.

DynamicRecord을 반환합니다.

def isError

```
def isError
```

이 기록이 오류 기록인지 확인합니다.

def getError

```
def getError
```

이 기록이 오류 기록이면 Error를 얻습니다. 이 기록이 오류 기록이면 scala.Some Some(오류)를 반환하고 그렇지 않으면 scala.None입니다.

def clearError

```
def clearError
```

Error을 scala.None.None로 설정합니다.

def write

```
override def write( out : DataOutput ) : Unit
```

def readFields

```
override def readFields( in : DataInput ) : Unit
```

def clone

```
override def clone : DynamicRecord
```

이 레코드를 새 DynamicRecord로 복제하고 반환합니다.

def schema

```
def schema
```

이 기록을 검사하여 Schema를 얻습니다.

def getRoot

```
def getRoot : ObjectNode
```

이 레코드의 루트 ObjectNode를 얻습니다.

def toJson

```
def toJson : String
```

이 레코드의 JSON 문자열을 얻습니다.

def getFieldNode

```
def getFieldNode( path : String ) : Option[DynamicNode]
```

DynamicNode의 옵션으로 지정된 path의 필드 값을 가져옵니다.

필드가 존재하는 경우 scala.Some Some(DynamicNode)을 반환하고, 그렇지 않으면 scala.None.None입니다.

def getField

```
def getField( path : String ) : Option[Any]
```

DynamicNode의 옵션으로 지정된 path의 필드 값을 가져옵니다.

scala.Some Some(값)을 반환합니다.

def hashCode

```
override def hashCode : Int
```

def equals

```
override def equals( other : Any )
```

DynamicRecord 객체

```
object DynamicRecord
```

정의 적용

```
def apply( row : Row,
           schema : SparkStructType )
```

Apache Spark SQL Row를 DynamicRecord로 변환하는 방법을 적용합니다.

- row - Spark SQL Row.
- schema - 행의 Schema입니다.

DynamicRecord을 반환합니다.

RecordTraverser 특성

```
trait RecordTraverser {
  def nullValue(): Unit
```

```
  def byteValue(value: Byte): Unit
  def binaryValue(value: Array[Byte]): Unit
  def booleanValue(value: Boolean): Unit
  def shortValue(value: Short) : Unit
  def intValue(value: Int) : Unit
  def longValue(value: Long) : Unit
  def floatValue(value: Float): Unit
  def doubleValue(value: Double): Unit
  def decimalValue(value: BigDecimal): Unit
  def stringValue(value: String): Unit
  def dateValue(value: Date): Unit
  def timestampValue(value: Timestamp): Unit
  def objectStart(length: Int): Unit
  def objectKey(key: String): Unit
  def objectEnd(): Unit
  def mapStart(length: Int): Unit
  def mapKey(key: String): Unit
  def mapEnd(): Unit
  def arrayStart(length: Int): Unit
  def arrayEnd(): Unit
}
```

AWS Glue Scala GlueContext API

패키지: com.amazonaws.services.glue

```
class GlueContext extends SQLContext(sc) (
           @transient val sc : SparkContext,
           val defaultSourcePartitioner : PartitioningStrategy )
```

`GlueContext`는 Amazon Simple Storage Service(Amazon S3), AWS Glue Data Catalog, JDBC 등에서 DynamicFrame을 읽고 쓰기 위한 진입점입니다. 이 클래스는 DataSource 특성 및 DataSink 객체를 생성하는 유틸리티 함수를 제공하며, 이 객체는 `DynamicFrame`을 읽고 쓸 때 사용할 수 있습니다.

`GlueContext`를 사용하여 `DynamicFrame`에 목표한 수의 파티션(기본값 20)을 설정할 수도 있습니다. 단, 소스에서 생성된 파티션 수가 파티션 최소 임계값(기본값 10) 미만이어야 합니다.

def addIngestionColumns

```
def addIngestionTimeColumns(
         df : DataFrame,
         timeGranularity : String = "") : dataFrame
```

입력 DataFrame에 ingest_year, ingest_month, ingest_day, ingest_hour, ingest_minute와 같은 수집 시간 열을 추가합니다. 이 함수는 Amazon S3을 대상으로 하는 데이터 카탈로그 테이블을 지정할 때 AWS Glue가 생성하는 스크립트에서 자동으로 생성됩니다. 이 함수는 출력 테이블의 수집 시간 열로 파티션을 자동으로 업데이트합니다. 이를 통해 입력 데이터에 명시적인 수집 시간 열을 요구하지 않고도 출력 데이터를 수집 시간에 자동으로 분할할 수 있습니다.

- dataFrame - 수집 시간 열을 추가할 dataFrame입니다.
- timeGranularity - 시간 열의 세분성입니다. 유효 값은 "day", "hour" 및 "minute"입니다. 예를 들어 "hour"가 함수에 전달되면 원래 dataFrame에 "ingest_year", "ingest_month", "ingest_day" 및 "ingest_hour" 시간 열이 추가됩니다.

시간 세분성 열을 추가한 후 데이터 프레임을 반환합니다.

예시

```
glueContext.addIngestionTimeColumns(dataFrame, "hour")
```

def createDataFrameFromOptions

```
def createDataFrameFromOptions( connectionType : String,
                        connectionOptions : JsonOptions,
                        transformationContext : String = "",
                        format : String = null,
                        formatOptions : JsonOptions = JsonOptions.empty
                      ) : DataSource
```

지정된 연결 및 포맷으로 생성된 DataFrame을 반환합니다. 이 함수는 AWS Glue 스트리밍 소스에만 사용합니다.

- connectionType - 스트리밍 연결 유형입니다. 유효한 값에는 kinesis 및 kafka(이)가 있습니다.
- connectionOptions - 연결 옵션이며, Kinesis 및 Kafka에 대해 서로 다릅니다. AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션에서 각 스트리밍 데이터 원본에 대한 모든 연결 옵션 목록을 찾을 수 있습니다. 스트리밍 연결 옵션에는 다음과 같은 차이점이 있습니다.
  - Kinesis 스트리밍 소스에는 streamARN, startingPosition, inferSchema 및 classification이 필요합니다.
  - Kafka 스트리밍 소스에는 connectionName, topicName, startingOffsets, inferSchema 및 classification이 필요합니다.

- `transformationContext` - 사용할 변환 컨텍스트입니다(선택 사항).
- `format` - 형식 사양입니다(선택 사항). 여러 포맷을 지원하는 Amazon S3 또는 AWS Glue 연결에 사용됩니다. 지원되는 형식에 관한 내용은 AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션 섹션을 참조하세요.
- `formatOptions` - 지정된 형식에 대한 형식 옵션입니다. 지원되는 포맷 옵션에 대한 자세한 내용은 데이터 포맷 옵션 섹션을 참조하세요.

Amazon Kinesis 스트리밍 소스의 예:

```
val data_frame_datasource0 =
glueContext.createDataFrameFromOptions(transformationContext = "datasource0",
 connectionType = "kinesis",
connectionOptions = JsonOptions("""{"streamName": "example_stream", "startingPosition":
 "TRIM_HORIZON", "inferSchema": "true", "classification": "json"}}""" ))
```

Kafka 스트리밍 소스의 예:

```
val data_frame_datasource0 =
glueContext.createDataFrameFromOptions(transformationContext = "datasource0",
 connectionType = "kafka",
connectionOptions = JsonOptions("""{"connectionName": "example_connection",
 "topicName": "example_topic", "startingPosition": "earliest", "inferSchema": "false",
 "classification": "json", "schema":"`column1` STRING, `column2` STRING"}""" ))
```

forEachBatch

`forEachBatch(frame, batch_function, options)`

스트리밍 소스에서 읽는 모든 마이크로 배치에 전달된 `batch_function`을 적용합니다.

- `frame` - 현재 마이크로 배치를 포함하는 DataFrame입니다.
- `batch_function` - 모든 마이크로 배치에 적용될 함수입니다.
- `options` - 마이크로 배치를 처리하는 방법에 대한 정보가 들어 있는 키-값 페어 컬렉션입니다. 다음 옵션이 필요합니다.
  - `windowSize` - 각 배치를 처리하는 데 소요되는 시간입니다.
  - `checkpointLocation` - 스트리밍 ETL 작업에 대해 체크포인트가 저장되는 위치입니다.
  - `batchMaxRetries` – 실패한 경우 배치를 다시 시도할 수 있는 최대 횟수입니다. 기본값은 3입니다. 이 옵션은 Glue 버전 2.0 이상에서만 구성할 수 있습니다.

예:

```
glueContext.forEachBatch(data_frame_datasource0, (dataFrame: Dataset[Row], batchId:
 Long) =>
   {
      if (dataFrame.count() > 0)
        {
          val datasource0 = DynamicFrame(glueContext.addIngestionTimeColumns(dataFrame,
 "hour"), glueContext)
          // @type: DataSink
          // @args: [database = "tempdb", table_name = "fromoptionsoutput",
 stream_batch_time = "100 seconds",
          //      stream_checkpoint_location = "s3://from-options-testing-eu-central-1/
fromOptionsOutput/checkpoint/",
          //      transformation_ctx = "datasink1"]
          // @return: datasink1
          // @inputs: [frame = datasource0]
          val options_datasink1 = JsonOptions(
             Map("partitionKeys" -> Seq("ingest_year", "ingest_month","ingest_day",
 "ingest_hour"),
             "enableUpdateCatalog" -> true))
          val datasink1 = glueContext.getCatalogSink(
             database = "tempdb",
             tableName = "fromoptionsoutput",
             redshiftTmpDir = "",
             transformationContext = "datasink1",
             additionalOptions = options_datasink1).writeDynamicFrame(datasource0)
        }
   }, JsonOptions("""{"windowSize" : "100 seconds",
         "checkpointLocation" : "s3://from-options-testing-eu-central-1/
fromOptionsOutput/checkpoint/"}"""))
```

def getCatalogSink

```
def getCatalogSink( database : String,
        tableName : String,
        redshiftTmpDir : String = "",
        transformationContext : String = ""
        additionalOptions: JsonOptions = JsonOptions.empty,
        catalogId: String = null
) : DataSink
```

Data Catalog에 정의된 테이블에 지정된 위치에 쓰는 DataSink를 생성합니다.

- `database` - Data Catalog에 있는 데이터베이스 이름입니다.
- `tableName` - Data Catalog에 있는 테이블 이름입니다.
- `redshiftTmpDir` - 특정 데이터 싱크와 사용되는 임시 준비 디렉터리입니다. 기본적으로 비우도록 설정합니다.
- `transformationContext` - 작업 북마크에서 사용되는 싱크와 관련된 변환 컨텍스트입니다. 기본적으로 비우도록 설정합니다.
- `additionalOptions` - AWS Glue에 제공되는 추가 옵션입니다.
- `catalogId` - 액세스 중인 Data Catalog의 카탈로그 ID(계정 ID)입니다. null인 경우 호출자의 기본 계정 ID가 사용됩니다.

`DataSink`을 반환합니다.

def getCatalogSource

```
def getCatalogSource( database : String,
                      tableName : String,
                      redshiftTmpDir : String = "",
                      transformationContext : String = ""
                      pushDownPredicate : String = " "
                      additionalOptions: JsonOptions = JsonOptions.empty,
                      catalogId: String = null
                    ) : DataSource
```

Data Catalog의 테이블 정의에서 데이터를 읽는 DataSource 특성를 생성합니다.

- `database` - Data Catalog에 있는 데이터베이스 이름입니다.
- `tableName` - Data Catalog에 있는 테이블 이름입니다.
- `redshiftTmpDir` - 특정 데이터 싱크와 사용되는 임시 준비 디렉터리입니다. 기본적으로 비우도록 설정합니다.
- `transformationContext` - 작업 북마크에서 사용되는 싱크와 관련된 변환 컨텍스트입니다. 기본적으로 비우도록 설정합니다.
- `pushDownPredicate` - 데이터 집합 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 자세한 내용은 푸시다운 조건자를 사용하여 예비 필터링 단원을 참조하십시오.
- `additionalOptions` - 선택적 이름-값 페어의 모음입니다. 가능한 옵션에는 AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션에 나열된 옵션이 포함됩니다(`endpointUrl`, `streamName`,

bootstrap.servers, security.protocol, topicName, classification 및 delimiter 제외). 지원되는 또 다른 옵션은 catalogPartitionPredicate입니다.

catalogPartitionPredicate - 카탈로그 표현식을 전달하여 인덱스 열을 기준으로 필터링할 수 있습니다. 이렇게 하면 필터링이 서버 측으로 푸시됩니다. 자세한 내용은 AWS Glue 파티션 인덱스를 참조하세요. push_down_predicate와 catalogPartitionPredicate는 다른 구문을 사용합니다. 전자는 Spark SQL 표준 구문을 사용하고 후자는 JSQL 구문 분석기를 사용합니다.

- catalogId - 액세스 중인 Data Catalog의 카탈로그 ID(계정 ID)입니다. null인 경우 호출자의 기본 계정 ID가 사용됩니다.

DataSource을 반환합니다.

스트리밍 소스의 예

```
val data_frame_datasource0 = glueContext.getCatalogSource(
    database = "tempdb",
    tableName = "test-stream-input",
    redshiftTmpDir = "",
    transformationContext = "datasource0",
    additionalOptions = JsonOptions("""{
        "startingPosition": "TRIM_HORIZON", "inferSchema": "false"}""")
    ).getDataFrame()
```

def getJDBCSink

```
def getJDBCSink( catalogConnection : String,
                 options : JsonOptions,
                 redshiftTmpDir : String = "",
                 transformationContext : String = "",
                 catalogId: String = null
               ) : DataSink
```

Data Catalog의 Connection 객체에 지정된 JDBC 데이터베이스에 쓰는 DataSink를 생성합니다. Connection 객체는 URL, 사용자 이름, 암호, VPC, 서브넷 및 보안 그룹을 포함한 JDBC 싱크로 연결하는 정보를 가지고 있습니다.

- catalogConnection - 작성할 JDBC URL을 포함하는 Data Catalog의 연결 이름입니다.
- options - JDBC 데이터 스토어로 작성할 때 필요한 추가 정보를 제공하는 JSON 이름-값 페어의 문자열입니다. 여기에는 다음이 포함됩니다.

  - dbtable(필수) - JDBC 테이블의 이름입니다. 데이터베이스 내의 스키마를 지원하는 JDBC 데이터 스토어의 경우 `schema.table-name`에 대해 지정합니다. 스키마가 제공되지 않으면 기본 "퍼블릭" 스키마가 사용됩니다. 다음 예제에서는 `test`라는 스키마와 `test_db` 데이터베이스의 `test_table` 테이블을 가리키는 옵션 파라미터를 보여 줍니다.

    ```
    options = JsonOptions("""{"dbtable": "test.test_table", "database": "test_db"}""")
    ```

  - database(필수) - JDBC 데이터베이스의 이름입니다.
  - SparkSQL JDBC 라이터에게 직접 전달되는 추가 옵션. 자세한 내용은 Redshift data source for Spark를 참조하십시오.
- `redshiftTmpDir` - 특정 데이터 싱크와 사용되는 임시 준비 디렉터리입니다. 기본적으로 비우도록 설정합니다.
- `transformationContext` - 작업 북마크에서 사용되는 싱크와 관련된 변환 컨텍스트입니다. 기본적으로 비우도록 설정합니다.
- `catalogId` - 액세스 중인 Data Catalog의 카탈로그 ID(계정 ID)입니다. null인 경우 호출자의 기본 계정 ID가 사용됩니다.

예제 코드:

```
getJDBCSink(catalogConnection = "my-connection-name", options =
 JsonOptions("""{"dbtable": "my-jdbc-table", "database": "my-jdbc-db"}"""),
 redshiftTmpDir = "", transformationContext = "datasink4")
```

`DataSink`을 반환합니다.

def getSink

```
def getSink( connectionType : String,
             connectionOptions : JsonOptions,
             transformationContext : String = ""
           ) : DataSink
```

Amazon Simple Storage Service(S3), JDBC, AWS Glue 데이터 카탈로그나 Apache Kafka 또는 Amazon Kinesis 데이터 스트림과 같은 대상에 데이터를 쓰는 DataSink를 생성합니다.

- `connectionType` - 연결 유형입니다. the section called “연결 파라미터” 섹션을 참조하세요.

- `connectionOptions` - 데이터 싱크 연결하려는 추가 정보를 제공하는 JSON 이름-값 페어 문자열입니다. the section called “연결 파라미터” 섹션을 참조하세요.
- `transformationContext` - 작업 북마크에서 사용되는 싱크와 관련된 변환 컨텍스트입니다. 기본적으로 비우도록 설정합니다.

`DataSink`을 반환합니다.

def getSinkWithFormat

```
def getSinkWithFormat( connectionType : String,
                       options : JsonOptions,
                       transformationContext : String = "",
                       format : String = null,
                       formatOptions : JsonOptions = JsonOptions.empty
                     ) : DataSink
```

Amazon S3, JDBC, 데이터 카탈로그나 Apache Kafka 또는 Amazon Kinesis 데이터 스트림과 같은 대상에 데이터를 쓰는 DataSink를 생성합니다. 또한 대상에 기록할 데이터의 형식을 설정합니다.

- `connectionType` - 연결 유형입니다. the section called “연결 파라미터” 섹션을 참조하세요.
- `options` - 데이터 싱크 연결하려는 추가 정보를 제공하는 JSON 이름-값 페어 문자열입니다. the section called “연결 파라미터” 섹션을 참조하세요.
- `transformationContext` - 작업 북마크에서 사용되는 싱크와 관련된 변환 컨텍스트입니다. 기본적으로 비우도록 설정합니다.
- `format` - 대상에 작성되는 데이터의 포맷입니다.
- `formatOptions` - 대상 주소로 데이터를 구성하는 추가 옵션을 제공하는 JSON 이름-값 페어 문자열입니다. 데이터 포맷 옵션 섹션을 참조하세요.

`DataSink`을 반환합니다.

def getSource

```
def getSource( connectionType : String,
               connectionOptions : JsonOptions,
               transformationContext : String = ""
               pushDownPredicate
             ) : DataSource
```

Amazon S3, JDBC, AWS Glue Data Catalog 같은 소스에서 데이터를 읽는 DataSource 특성를 생성합니다. Kafka 및 Kinesis 스트리밍 데이터 원본도 지원합니다.

- `connectionType` - 데이터 원본의 유형입니다. the section called “연결 파라미터” 섹션을 참조하세요.
- `connectionOptions` - 데이터 원본을 연결하려는 추가 정보를 제공하는 JSON 이름-값 페어 문자열입니다. 자세한 내용은 the section called “연결 파라미터” 단원을 참조하십시오.

  Kinesis 스트리밍 소스에는 `streamARN`, `startingPosition`, `inferSchema` 및 `classification` 연결 옵션이 필요합니다.

  Kafka 스트리밍 소스에는 `connectionName`, `topicName`, `startingOffsets`, `inferSchema` 및 `classification` 연결 옵션이 필요합니다.
- `transformationContext` - 작업 북마크에서 사용되는 싱크와 관련된 변환 컨텍스트입니다. 기본적으로 비우도록 설정합니다.
- `pushDownPredicate` - 파티션 열에 대한 조건자입니다.

`DataSource`을 반환합니다.

Amazon Kinesis 스트리밍 소스의 예:

```
val kinesisOptions = jsonOptions()
data_frame_datasource0 = glueContext.getSource("kinesis",
 kinesisOptions).getDataFrame()

private def jsonOptions(): JsonOptions = {
    new JsonOptions(
      s"""{"streamARN": "arn:aws:kinesis:eu-central-1:123456789012:stream/
fromOptionsStream",
         |"startingPosition": "TRIM_HORIZON",
         |"inferSchema": "true",
         |"classification": "json"}""".stripMargin)
}
```

Kafka 스트리밍 소스의 예:

```
val kafkaOptions = jsonOptions()
val data_frame_datasource0 = glueContext.getSource("kafka",
 kafkaOptions).getDataFrame()
```

```
private def jsonOptions(): JsonOptions = {
    new JsonOptions(
      s"""{"connectionName": "ConfluentKafka",
         |"topicName": "kafka-auth-topic",
         |"startingOffsets": "earliest",
         |"inferSchema": "true",
         |"classification": "json"}""".stripMargin)
 }
```

def getSourceWithFormat

```
def getSourceWithFormat( connectionType : String,
                         options : JsonOptions,
                         transformationContext : String = "",
                         format : String = null,
                         formatOptions : JsonOptions = JsonOptions.empty
                       ) : DataSource
```

Amazon S3, JDBC, AWS Glue Data Catalog 같은 소스에서 데이터를 읽는 DataSource 특성를 생성하고, 소스에 저장된 데이터 포맷도 설정합니다.

- `connectionType` - 데이터 원본의 유형입니다. the section called “연결 파라미터” 섹션을 참조하세요.
- `options` - 데이터 원본을 연결하려는 추가 정보를 제공하는 JSON 이름-값 페어 문자열입니다. the section called “연결 파라미터” 섹션을 참조하세요.
- `transformationContext` - 작업 북마크에서 사용되는 싱크와 관련된 변환 컨텍스트입니다. 기본적으로 비우도록 설정합니다.
- `format` - 소스에 저장된 데이터의 포맷입니다. `connectionType`이 "s3"일 경우에도 `format`을 지정할 수 있습니다. “avro”, “csv”, “grokLog”, “ion”, “json”, “xml”, “parquet”, “orc” 중 하나가 이에 해당합니다.
- `formatOptions` - 소스로 데이터를 파싱하는 추가 옵션을 제공하는 JSON 이름-값 페어 문자열입니다. 데이터 포맷 옵션 섹션을 참조하세요.

`DataSource`을 반환합니다.

예시

Amazon S3에서 CSV(쉼표로 구분된 값) 파일인 데이터 원본에서 DynamicFrame을 생성합니다.

```
val datasource0 = glueContext.getSourceWithFormat(
    connectionType="s3",
    options =JsonOptions(s"""{"paths": [ "s3://csv/nycflights.csv"]}"""),
    transformationContext = "datasource0",
    format = "csv",
    formatOptions=JsonOptions(s"""{"withHeader":"true","separator": ","}""")
    ).getDynamicFrame()
```

JDBC 연결을 사용하여 PostgreSQL인 데이터 원본에서 DynamicFrame을 생성합니다.

```
val datasource0 = glueContext.getSourceWithFormat(
    connectionType="postgresql",
    options =JsonOptions(s"""{
      "url":"jdbc:postgresql://databasePostgres-1.rds.amazonaws.com:5432/testdb",
      "dbtable": "public.company",
      "redshiftTmpDir":"",
      "user":"username",
      "password":"password123"
    }"""),
    transformationContext = "datasource0").getDynamicFrame()
```

JDBC 연결을 사용하여 MySQL인 데이터 원본에서 DynamicFrame을 생성합니다.

```
 val datasource0 = glueContext.getSourceWithFormat(
    connectionType="mysql",
    options =JsonOptions(s"""{
      "url":"jdbc:mysql://databaseMysql-1.rds.amazonaws.com:3306/testdb",
      "dbtable": "athenatest_nycflights13_csv",
      "redshiftTmpDir":"",
      "user":"username",
      "password":"password123"
    }"""),
    transformationContext = "datasource0").getDynamicFrame()
```

def getSparkSession

```
def getSparkSession : SparkSession
```

이 GlueContext와 관련된 SparkSession 객체를 얻습니다. 이 SparkSession 객체를 사용하여 DynamicFrame에서 생성된 DataFrame을 사용할 테이블 및 UDF를 등록합니다.

SparkSession을 반환합니다.

def startTransaction

```
def startTransaction(readOnly: Boolean):String
```

새 트랜잭션을 시작합니다. 내부적으로 Lake Formation startTransaction API를 호출합니다.

- `readOnly` - (부울) 이 트랜잭션이 읽기 전용인지 또는 읽기/쓰기인지를 나타냅니다. 읽기 전용 트랜잭션 ID를 사용하여 수행된 쓰기는 거부됩니다. 읽기 전용 트랜잭션은 커밋할 필요가 없습니다.

트랜잭션 ID를 반환합니다.

def commitTransaction

```
def commitTransaction(transactionId: String, waitForCommit: Boolean): Boolean
```

지정된 트랜잭션을 커밋하려는 시도입니다. 트랜잭션이 커밋을 완료하기 전에 `commitTransaction`이 반환될 수 있습니다. 내부적으로 Lake Formation commitTransaction API를 호출합니다.

- `transactionId` - (문자열) 커밋할 트랜잭션입니다.
- `waitForCommit` - (부울) `commitTransaction`이 즉시 반환되는지 여부를 결정합니다. 기본값은 true입니다. false인 경우 `commitTransaction`은 폴링한 후 트랜잭션이 커밋될 때까지 기다립니다. 대기 시간은 최대 재시도 횟수가 6회인 지수 백오프를 사용하여 1분으로 제한됩니다.

커밋이 수행되었는지 여부를 나타내는 부울을 반환합니다.

def cancelTransaction

```
def cancelTransaction(transactionId: String): Unit
```

지정된 트랜잭션을 취소하려는 시도입니다. 내부적으로 Lake Formation CancelTransaction API를 호출합니다.

- `transactionId` - (문자열) 취소할 트랜잭션입니다.

트랜잭션이 이전에 커밋된 경우 `TransactionCommittedException` 예외를 반환합니다.

def this

```
def this( sc : SparkContext,
          minPartitions : Int,
          targetPartitions : Int )
```

지정된 SparkContext, 최소 파티션 및 대상 파티션을 사용하여 GlueContext 객체를 생성합니다.

- sc - SparkContext
- minPartitions - 파티션의 최소 수입니다.
- targetPartitions - 파티션의 목표 수입니다.

GlueContext을 반환합니다.

def this

```
def this( sc : SparkContext )
```

제공된 SparkContext로 GlueContext 객체를 생성합니다. 최소 파티션을 10으로 대상 파티션을 20으로 설정합니다.

- sc - SparkContext

GlueContext을 반환합니다.

def this

```
def this( sparkContext : JavaSparkContext )
```

제공된 JavaSparkContext로 GlueContext 객체를 생성합니다. 최소 파티션을 10으로 대상 파티션을 20으로 설정합니다.

- sparkContext - JavaSparkContext

GlueContext을 반환합니다.

MappingSpec

패키지: com.amazonaws.services.glue

MappingSpec 케이스 클래스

```
case class MappingSpec( sourcePath: SchemaPath,
                        sourceType: DataType,
                        targetPath: SchemaPath,
                        targetType: DataTyp
                       ) extends Product4[String, String, String, String] {
  override def _1: String = sourcePath.toString
  override def _2: String = ExtendedTypeName.fromDataType(sourceType)
  override def _3: String = targetPath.toString
  override def _4: String = ExtendedTypeName.fromDataType(targetType)
}
```

- `sourcePath` - 소스 필드의 `SchemaPath`입니다.
- `sourceType` - 소스 필드의 `DataType`입니다.
- `targetPath` - 대상 필드의 `SchemaPath`입니다.
- `targetType` - 대상 필드의 `DataType`입니다.

`MappingSpec`은 소스 경로 및 소스 데이터 유형에서 대상 경로 및 대상 데이터 유형까지의 매핑을 지정합니다. 소스 프레임의 소스 경로 값은 대상 경로의 대상 프레임에서 볼 수 있습니다. 소스 데이터 유형은 대상 데이터 유형으로 캐스팅됩니다.

`applyMapping` 인터페이스에서 `Product4`을 처리할 수 있도록 `Product4`에서 확장됩니다.

MappingSpec 객체

```
object MappingSpec
```

`MappingSpec` 객체에는 다음 멤버가 있습니다.

Val orderingByTarget

```
val orderingByTarget: Ordering[MappingSpec]
```

정의 적용

```
def apply( sourcePath : String,
           sourceType : DataType,
```

```
            targetPath : String,
            targetType : DataType
          ) : MappingSpec
```

MappingSpec을 만듭니다.

- sourcePath - 소스 경로의 문자열 표현입니다.
- sourceType - 소스 DataType입니다.
- targetPath - 대상 경로의 문자열 표현입니다.
- targetType - 대상 DataType입니다.

MappingSpec을 반환합니다.

정의 적용

```
def apply( sourcePath : String,
           sourceTypeString : String,
           targetPath : String,
           targetTypeString : String
         ) : MappingSpec
```

MappingSpec을 만듭니다.

- sourcePath - 소스 경로의 문자열 표현입니다.
- sourceType - 원본 데이터 유형의 문자열 표현입니다.
- targetPath - 대상 경로의 문자열 표현입니다.
- targetType - 대상 데이터 유형의 문자열 표현입니다.

MappingSpec을 반환합니다.

정의 적용

```
def apply( product : Product4[String, String, String, String] ) : MappingSpec
```

MappingSpec을 만듭니다.

- product - 소스 경로, 원본 데이터 유형, 대상 경로 및 대상 데이터 유형의 Product4입니다.

MappingSpec을 반환합니다.

AWS Glue Scala ResolveSpec API

주제

- ResolveSpec 객체
- ResolveSpec 케이스 클래스

패키지: com.amazonaws.services.glue

ResolveSpec 객체

ResolveSpec

```
object ResolveSpec
```

def

```
def apply( path : String,
           action : String
         ) : ResolveSpec
```

ResolveSpec을 만듭니다.

- path - 확인해야 할 선택 필드를 나타내는 문자열입니다.
- action - 확인 작업입니다. Project, KeepAsStruct 또는 Cast 중 하나가 작업이 될 수 있습니다.

ResolveSpec을 반환합니다.

def

```
def apply( product : Product2[String, String] ) : ResolveSpec
```

ResolveSpec을 만듭니다.

- product - 소스 경로, 확인 작업의 Product2입니다.

ResolveSpec을 반환합니다.

ResolveSpec 케이스 클래스

```
case class ResolveSpec extends Product2[String, String]  (
           path : SchemaPath,
           action : String )
```

ResolveSpec을 만듭니다.

- path - 확인해야 할 선택 필드의 SchemaPath입니다.
- action - 확인 작업입니다. Project, KeepAsStruct 또는 Cast 중 하나가 작업이 될 수 있습니다.

ResolveSpec def 메서드

```
def _1 : String
```

```
def _2 : String
```

AWS Glue Scala ArrayNode API

패키지: com.amazonaws.services.glue.types

ArrayNode 케이스 클래스

ArrayNode

```
case class ArrayNode extends DynamicNode  (
           value : ArrayBuffer[DynamicNode] )
```

ArrayNode 정의 메서드

```
def add( node : DynamicNode )
```

```
def clone
```

```
def equals( other : Any )
```

```
def get( index : Int ) : Option[DynamicNode]
```

```
def getValue
```

```
def hashCode : Int
```

```
def isEmpty : Boolean
```

```
def nodeType
```

```
def remove( index : Int )
```

```
def this
```

```
def toIterator : Iterator[DynamicNode]
```

```
def toJson : String
```

```
def update( index : Int,
            node : DynamicNode )
```

AWS Glue Scala BinaryNode API

패키지: com.amazonaws.services.glue.types

BinaryNode 케이스 클래스

BinaryNode

```
case class BinaryNode extends ScalarNode(value, TypeCode.BINARY)  (
           value : Array[Byte] )
```

BinaryNode val 필드

- ordering

BinaryNode 정의 메서드

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

AWS Glue Scala BooleanNode API

패키지: com.amazonaws.services.glue.types

BooleanNode 케이스 클래스

BooleanNode

```
case class BooleanNode extends ScalarNode(value, TypeCode.BOOLEAN)  (
           value : Boolean )
```

BooleanNode val 필드

- ordering

BooleanNode 정의 메서드

```
def equals( other : Any )
```

AWS Glue Scala ByteNode API

패키지: com.amazonaws.services.glue.types

ByteNode 케이스 클래스

ByteNode

```
case class ByteNode extends ScalarNode(value, TypeCode.BYTE)  (
           value : Byte )
```

ByteNode val 필드

- `ordering`

ByteNode 정의 메서드

```
def equals( other : Any )
```

AWS Glue Scala DateNode API

패키지: com.amazonaws.services.glue.types

DateNode 케이스 클래스

DateNode

```
case class DateNode extends ScalarNode(value, TypeCode.DATE)  (
           value : Date )
```

DateNode val 필드

- `ordering`

ByteNode def 메서드

```
def equals( other : Any )
```

```
def this( value : Int )
```

AWS Glue Scala DecimalNode API

패키지: com.amazonaws.services.glue.types

DecimalNode 케이스 클래스

DecimalNode

```
case class DecimalNode extends ScalarNode(value, TypeCode.DECIMAL)  (
```

```
            value : BigDecimal )
```

DecimalNode val 필드

- `ordering`

DecimalNode def 메서드

```
def equals( other : Any )
```

```
def this( value : Decimal )
```

AWS Glue Scala DoubleNode API

패키지: com.amazonaws.services.glue.types

DoubleNode 케이스 클래스

DoubleNode

```
case class DoubleNode extends ScalarNode(value, TypeCode.DOUBLE)  (
            value : Double )
```

DoubleNode val 필드

- `ordering`

DoubleNode def 메서드

```
def equals( other : Any )
```

AWS Glue Scala DynamicNode API

주제

- DynamicNode 클래스
- DynamicNode 객체

패키지: com.amazonaws.services.glue.types

DynamicNode 클래스

DynamicNode

```
class DynamicNode extends Serializable with Cloneable
```

DynamicNode def 메서드

```
def getValue : Any
```

플레인 값을 가져오며 현재 레코드에 바인딩됩니다.

```
def nodeType : TypeCode
```

```
def toJson : String
```

디버깅 방법

```
def toRow( schema : Schema,
           options : Map[String, ResolveOption]
         ) : Row
```

```
def typeName : String
```

DynamicNode 객체

DynamicNode

```
object DynamicNode
```

DynamicNode def 메서드

```
def quote( field : String,
           useQuotes : Boolean
         ) : String
```

```
def quote( node : DynamicNode,
```

```
            useQuotes : Boolean
          ) : String
```

EvaluateDataQuality 클래스

AWS Glue 데이터 품질은 AWS Glue의 미리 보기 릴리스이므로 변경될 수 있습니다.

패키지: com.amazonaws.services.glue.dq

```
object EvaluateDataQuality
```

def apply

```
def apply(frame: DynamicFrame,
            ruleset: String,
            publishingOptions: JsonOptions = JsonOptions.empty): DynamicFrame
```

DynamicFrame을 기준으로 데이터 품질 규칙 세트를 평가하고 새 DynamicFrame을 평가 결과와 함께 반환합니다. AWS Glue 데이터 품질에 대한 자세한 내용은 AWS Glue Data Quality 섹션을 참조하세요.

- frame - 데이터 품질을 평가하려는 DynamicFrame입니다.
- ruleset - 문자열 형식의 DQDL(데이터 품질 정의 언어) 규칙 세트입니다. DQDL에 대한 자세한 내용은 데이터 품질 정의 언어(DQDL) 참조 안내서를 참조하세요.
- publishingOptions - 평가 결과 및 지표를 게시하기 위해 다음 옵션을 지정하는 사전입니다.
  - dataQualityEvaluationContext - AWS Glue가 Amazon CloudWatch 지표와 데이터 품질 결과를 게시할 네임스페이스를 지정하는 문자열입니다. 집계된 지표는 CloudWatch에 표시되고 전체 결과는 AWS Glue Studio 인터페이스에 표시됩니다.
    - 필수 항목 여부: 아니요
    - 기본 값: default_context
  - enableDataQualityCloudWatchMetrics - 데이터 품질 평가 결과를 CloudWatch에 게시할지 여부를 지정합니다. dataQualityEvaluationContext 옵션을 사용하여 지표의 네임스페이스를 지정합니다.
    - 필수 항목 여부: 아니요
    - 기본값: False

- `enableDataQualityResultsPublishing` - AWS Glue Studio 인터페이스의 Data Quality(데이터 품질) 탭에 데이터 품질 결과를 표시할지 여부를 지정합니다.
  - 필수 항목 여부: 아니요
  - 기본값: True
- `resultsS3Prefix` - AWS Glue가 데이터 품질 평가 결과를 기록할 수 있는 Amazon S3 위치를 지정합니다.
  - 필수 항목 여부: 아니요
  - 기본값: ""(빈 문자열)

예

다음 예제 코드는`SelectFields` 변환을 수행하기 전에 `DynamicFrame`의 데이터 품질을 평가하는 방법을 보여줍니다. 스크립트는 변환을 시도하기 전에 모든 데이터 품질 규칙이 통과하는지 확인합니다.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.dq.EvaluateDataQuality

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Create DynamicFrame with data
    val Legislators_Area = glueContext.getCatalogSource(database="legislators",
 tableName="areas_json", transformationContext="S3bucket_node1").getDynamicFrame()

    // Define data quality ruleset
    val DQ_Ruleset = """
      Rules = [ColumnExists "id"]
```

```
    """

    // Evaluate data quality
    val DQ_Results = EvaluateDataQuality.apply(frame=Legislators_Area,
 ruleset=DQ_Ruleset, publishingOptions=JsonOptions("""{"dataQualityEvaluationContext":
 "Legislators_Area", "enableDataQualityMetrics": "true",
 "enableDataQualityResultsPublishing": "true"}"""))
    assert(DQ_Results.filter(_.getField("Outcome").contains("Failed")).count == 0,
 "Failing DQ rules for Legislators_Area caused the job to fail.")

    // Script generated for node Select Fields
    val SelectFields_Results = Legislators_Area.selectFields(paths=Seq("id", "name"),
 transformationContext="Legislators_Area")

    Job.commit()
  }
}
```

AWS Glue Scala FloatNode API

패키지: com.amazonaws.services.glue.types

FloatNode 케이스 클래스

FloatNode

```
case class FloatNode extends ScalarNode(value, TypeCode.FLOAT)  (
           value : Float )
```

FloatNode val 필드

- `ordering`

FloatNode def 메서드

```
def equals( other : Any )
```

FillMissingValues 클래스

패키지: com.amazonaws.services.glue.ml

```
object FillMissingValues
```

def apply

```
def apply(frame: DynamicFrame,
          missingValuesColumn: String,
          outputColumn: String = "",
          transformationContext: String = "",
          callSite: CallSite = CallSite("Not provided", ""),
          stageThreshold: Long = 0,
          totalThreshold: Long = 0): DynamicFrame
```

지정된 열에 동적 프레임의 누락된 값을 채우고 새 열에 추정 값과 함께 새 프레임을 반환합니다. 누락된 값이 없는 행의 경우 지정된 열의 값이 새 열에 복제됩니다.

- `frame` - 누락된 값을 채울 DynamicFrame입니다. 필수 사항입니다.
- `missingValuesColumn` - 누락된 값(`null` 값 및 빈 문자열)이 포함된 열입니다. 필수 사항입니다.
- `outputColumn` - 값이 누락된 모든 행에 대한 예상 값을 포함할 새 열의 이름입니다. 선택 사항이며 기본값은 `"_filled"`가 접미사로 사용된 `missingValuesColumn`의 값입니다.
- `transformationContext` - 고유 문자열을 통해 상태 정보를 확인합니다(선택 사항).
- `callSite` - 오류 보고에 필요한 컨텍스트 정보를 제공하는 데 사용합니다(선택 사항).
- `stageThreshold` - 오류가 발생하기 전까지 변환에 따라 생길 수 있는 최대 오류 수입니다(선택 사항, 기본값은 0).
- `totalThreshold` - 오류가 진행되기 전까지 생길 수 있는 최대 전체 오류 수입니다(선택 사항, 기본값은 0).

누락된 값이 있는 행에 대한 추정 값과 다른 행에 대한 현재 값이 포함된 하나의 추가 열이 있는 새 동적 프레임을 반환합니다.

FindMatches 클래스

패키지: com.amazonaws.services.glue.ml

```
object FindMatches
```

def apply

```
def apply(frame: DynamicFrame,
            transformId: String,
            transformationContext: String = "",
```

```
            callSite: CallSite = CallSite("Not provided", ""),
            stageThreshold: Long = 0,
            totalThreshold: Long = 0,
            enforcedMatches: DynamicFrame = null): DynamicFrame,
   computeMatchConfidenceScores: Boolean
```

입력 프레임에서 일치 항목을 찾고 일치 그룹당 고유 ID를 포함하는 새 열이 있는 새 프레임을 반환합니다.

- `frame` - 일치하는 항목을 찾을 DynamicFrame입니다. 필수 사항입니다.
- `transformId` - 입력 프레임에 적용할 FindMatches 변환과 연결된 고유 ID입니다. 필수 사항입니다.
- `transformationContext` - 이 DynamicFrame의 식별자입니다. `transformationContext`는 실행 간 지속되는 작업 북마크 상태에 대한 키로 사용됩니다. 선택 사항입니다.
- `callSite` - 오류 보고에 필요한 컨텍스트 정보를 제공하는 데 사용합니다. Python에서 호출 할 때 자동으로 이 값이 설정됩니다. 선택 사항입니다.
- `stageThreshold` - 예외가 발생하기 전에 이 DynamicFrame 계산에 허용되는 오류 레코드의 최대 개수입니다(이전 DynamicFrame에 존재하는 레코드 제외). 선택 사항입니다. 기본값은 0입니다.
- `totalThreshold` - 예외가 발생하기 전 전체 오류 레코드의 최대 개수입니다(이전 프레임의 레코드 포함). 선택 사항입니다. 기본값은 0입니다.
- `enforcedMatches` - 강제 일치 항목에 대한 프레임입니다. 선택 사항입니다. 기본값은 `null`입니다.
- `computeMatchConfidenceScores` - 각 일치 레코드 그룹의 신뢰도 점수를 컴퓨팅할지 여부를 나타내는 부울 값입니다. 선택 사항입니다. 기본값은 false입니다.

일치하는 레코드의 각 그룹에 고유 식별자가 할당된 새 동적 프레임을 반환합니다.

FindIncrementalMatches 클래스

패키지: com.amazonaws.services.glue.ml

```
object FindIncrementalMatches
```

def apply

```
apply(existingFrame: DynamicFrame,
            incrementalFrame: DynamicFrame,
```

```
            transformId: String,
            transformationContext: String = "",
            callSite: CallSite = CallSite("Not provided", ""),
            stageThreshold: Long = 0,
            totalThreshold: Long = 0,
            enforcedMatches: DynamicFrame = null): DynamicFrame,
  computeMatchConfidenceScores: Boolean
```

기존 및 증분 프레임에서 일치 항목을 찾고 일치 그룹당 고유 ID가 포함된 열이 있는 새 프레임을 반환합니다.

- `existingframe` - 각 그룹에 대해 일치하는 ID가 할당된 기존 프레임입니다. 필수 사항입니다.
- `incrementalframe` - 기존 프레임과 일치하는 항목을 찾는 데 사용되는 증분 프레임입니다. 필수 사항입니다.
- `transformId` - 입력 프레임에 적용할 FindIncrementalMatches 변환과 연결된 고유 ID입니다. 필수 사항입니다.
- `transformationContext` - 이 `DynamicFrame`의 식별자입니다. `transformationContext`는 실행 간 지속되는 작업 북마크 상태에 대한 키로 사용됩니다. 선택 사항입니다.
- `callSite` - 오류 보고에 필요한 컨텍스트 정보를 제공하는 데 사용합니다. Python에서 호출 할 때 자동으로 이 값이 설정됩니다. 선택 사항입니다.
- `stageThreshold` - 예외가 발생하기 전에 이 `DynamicFrame` 계산에 허용되는 오류 레코드의 최대 개수입니다(이전 `DynamicFrame`에 존재하는 레코드 제외). 선택 사항입니다. 기본값은 `0`입니다.
- `totalThreshold` - 예외가 발생하기 전 전체 오류 레코드의 최대 개수입니다(이전 프레임의 레코드 포함). 선택 사항입니다. 기본값은 `0`입니다.
- `enforcedMatches` - 강제 일치 항목에 대한 프레임입니다. 선택 사항입니다. 기본값은 `null`입니다.
- `computeMatchConfidenceScores` - 각 일치 레코드 그룹의 신뢰도 점수를 컴퓨팅할지 여부를 나타내는 부울 값입니다. 선택 사항입니다. 기본값은 false입니다.

일치하는 레코드의 각 그룹에 고유 식별자가 할당된 새 동적 프레임을 반환합니다.

AWS Glue Scala IntegerNode API

패키지: com.amazonaws.services.glue.types

IntegerNode 케이스 클래스

IntegerNode

```
case class IntegerNode extends ScalarNode(value, TypeCode.INT)  (
           value : Int )
```

IntegerNode val 필드

- `ordering`

IntegerNode def 메서드

```
def equals( other : Any )
```

AWS Glue Scala LongNode API

패키지: com.amazonaws.services.glue.types

LongNode 케이스 클래스

LongNode

```
case class LongNode extends ScalarNode(value, TypeCode.LONG)  (
           value : Long )
```

LongNode val 필드

- `ordering`

LongNode def 메서드

```
def equals( other : Any )
```

AWS Glue Scala MapLikeNode API

패키지: com.amazonaws.services.glue.types

MapLikeNode 클래스

MapLikeNode

```
class MapLikeNode extends DynamicNode  (
```

```
            value : mutable.Map[String, DynamicNode] )
```

MapLikeNode def 메서드

```
def clear : Unit
```

```
def get( name : String ) : Option[DynamicNode]
```

```
def getValue
```

```
def has( name : String ) : Boolean
```

```
def isEmpty : Boolean
```

```
def put( name : String,
         node : DynamicNode
       ) : Option[DynamicNode]
```

```
def remove( name : String ) : Option[DynamicNode]
```

```
def toIterator : Iterator[(String, DynamicNode)]
```

```
def toJson : String
```

```
def toJson( useQuotes : Boolean ) : String
```

예제: 이 JSON을 고려할 때

```
{"foo": "bar"}
```

`useQuotes == true`이면 `toJson`이 `{"foo": "bar"}`를 생성합니다. `useQuotes == false`이면 `toJson`이 `{foo: bar}` @return을 생성합니다.

AWS Glue Scala MapNode API

패키지: com.amazonaws.services.glue.types

MapNode 케이스 클래스

MapNode

```
case class MapNode extends MapLikeNode(value)  (
           value : mutable.Map[String, DynamicNode] )
```

MapNode def 메서드

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

AWS Glue Scala NullNode API

주제

- NullNode 클래스
- NullNode 케이스 객체

패키지: com.amazonaws.services.glue.types

NullNode 클래스

NullNode

```
class NullNode
```

NullNode 케이스 객체

NullNode

```
case object NullNode extends NullNode
```

AWS Glue Scala ObjectNode API

주제

- ObjectNode 객체
- ObjectNode 케이스 클래스

패키지: com.amazonaws.services.glue.types

ObjectNode 객체

ObjectNode

```
object ObjectNode
```

ObjectNode def 메서드

```
def apply( frameKeys : Set[String],
           v1 : mutable.Map[String, DynamicNode],
           v2 : mutable.Map[String, DynamicNode],
           resolveWith : String
         ) : ObjectNode
```

ObjectNode 케이스 클래스

ObjectNode

```
case class ObjectNode extends MapLikeNode(value)  (
           val value : mutable.Map[String, DynamicNode] )
```

ObjectNode def 메서드

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

AWS Glue Scala ScalarNode API

주제

- ScalarNode 클래스
- ScalarNode 객체

패키지: com.amazonaws.services.glue.types

ScalarNode 클래스

ScalarNode

```
class ScalarNode extends DynamicNode  (
           value : Any,
           scalarType : TypeCode )
```

ScalarNode def 메서드

```
def compare( other : Any,
             operator : String
           ) : Boolean
```

```
def getValue
```

```
def hashCode : Int
```

```
def nodeType
```

```
def toJson
```

ScalarNode 객체

ScalarNode

```
object ScalarNode
```

ScalarNode def 메서드

```
def apply( v : Any ) : DynamicNode
```

```
def compare( tv : Ordered[T],
             other : T,
             operator : String
           ) : Boolean
```

```
def compareAny( v : Any,
                y : Any,
                o : String )
```

```
def withEscapedSpecialCharacters( jsonToEscape : String ) : String
```

AWS Glue Scala ShortNode API

패키지: com.amazonaws.services.glue.types

ShortNode 케이스 클래스

ShortNode

```
case class ShortNode extends ScalarNode(value, TypeCode.SHORT)  (
           value : Short )
```

ShortNode val 필드

- `ordering`

ShortNode def 메서드

```
def equals( other : Any )
```

AWS Glue Scala StringNode API

패키지: com.amazonaws.services.glue.types

StringNode 케이스 클래스

StringNode

```
case class StringNode extends ScalarNode(value, TypeCode.STRING)  (
           value : String )
```

StringNode val 필드

- `ordering`

StringNode def 메서드

```
def equals( other : Any )
```

```
def this( value : UTF8String )
```

AWS Glue Scala TimestampNode API

패키지: com.amazonaws.services.glue.types

TimestampNode 케이스 클래스

TimestampNode

```
case class TimestampNode extends ScalarNode(value, TypeCode.TIMESTAMP)  (
           value : Timestamp )
```

TimestampNode val 필드

- `ordering`

TimestampNode def 메서드

```
def equals( other : Any )
```

```
def this( value : Long )
```

AWS Glue Scala GlueArgParser API

패키지: com.amazonaws.services.glue.util

GlueArgParser 객체

GlueArgParser

```
object GlueArgParser
```

이것은 AWSGlueDataplanePython 패키지에 있는 utils.getResolvedOptions의 Python 버전과 엄격하게 일치합니다.

GlueArgParser def 메서드

```
def getResolvedOptions( args : Array[String],
                        options : Array[String]
                      ) : Map[String, String]
```

```
def initParser( userOptionsSet : mutable.Set[String] ) : ArgumentParser
```

Example 작업으로 전달된 인수 검색

작업 인수를 검색하려면 getResolvedOptions 메서드를 사용할 수 있습니다. aws_region 작업 인수를 검색하는 다음 예제를 검토합니다.

```
val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME","aws_region").toArray)
Job.init(args("JOB_NAME"), glueContext, args.asJava)
val region = args("aws_region")
println(region)
```

AWS Glue Scala 작업 API

패키지: com.amazonaws.services.glue.util

작업 객체

작업

```
object Job
```

작업 def 메서드

```
def commit
```

```
def init( jobName : String,
          glueContext : GlueContext,
          args : java.util.Map[String, String] = Map[String, String]().asJava
        ) : this.type
```

```
def init( jobName : String,
          glueContext : GlueContext,
          endpoint : String,
          args : java.util.Map[String, String]
        ) : this.type
```

```
def isInitialized
```

```
def reset
```

```
def runId
```

AWS Glue for Spark ETL 스크립트 프로그래밍을 위한 기능 및 최적화

다음 섹션은 어떤 언어든지 AWS Glue for Spark 추출, 전환 및 적재(ETL) 프로그래밍에 일반적으로 적용되는 기술 및 값을 설명합니다.

주제

- AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션
- AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션
- Spark SQL 작업에 대해 AWS Glue Data Catalog 지원
- 작업 북마크 사용
- AWS Glue Studio 외부에서 민감한 데이터 감지 사용

- AWS Glue Visual Job API

AWS Glue for Spark에서 ETL에 대한 연결 유형 및 옵션

AWS Glue for Spark에서 다양한 PySpark와 Scala 메서드 및 변환은 `connectionType` 파라미터를 사용하여 연결 유형을 지정합니다. `connectionOptions` 또는 `options` 파라미터를 사용하여 연결 옵션을 지정합니다.

`connectionType` 파라미터는 다음 표에 표시된 값을 사용할 수 있습니다. 각 유형에 대한 연관된 `connectionOptions`(또는 `options`) 파라미터 값은 다음 섹션에 설명되어 있습니다. 별도의 언급이 없으면 이 파라미터는 연결이 소스 또는 싱크로 사용될 때 적용됩니다.

연결 옵션을 설정하고 사용하는 방법을 보여주는 샘플 코드는 각 연결 유형별 홈페이지를 참조하세요.

connectionType	연결 대상
dynamodb	Amazon DynamoDB 데이터베이스
kinesis	Amazon Kinesis Data Streams
s3	Amazon S3
documentdb	Amazon DocumentDB(MongoDB와 호환) 데이터베이스
OpenSearch	Amazon OpenSearch Service.
redshift	Amazon Redshift 데이터베이스
kafka	Kafka 또는 Amazon Managed Streaming for Apache Kafka
azurecosmos	NoSQL용 Azure Cosmos.
azuresql	Azure SQL.
bigquery	Google BigQuery.
mongodb	MongoDB Atlas를 포함한 MongoDB 데이터베이스.
sqlserver	Microsoft SQL Server 데이터베이스(JDBC 연결 참조)
mysql	MySQL 데이터베이스(JDBC 연결 참조)

connectionType	연결 대상
oracle	Oracle Database(JDBC 연결 참조)
postgresql	PostgreSQL 데이터베이스(JDBC 연결 참조)
saphana	SAP HANA.
snowflake	Snowflake 데이터 레이크
Teradata	Teradata Vantage.
vertica	Vertica.
custom.*	Spark, Athena 또는 JDBC 데이터 원본(사용자 정의 및 AWS Marketplace 연결 유형 값 참조)
marketplace.*	Spark, Athena 또는 JDBC 데이터 원본(사용자 정의 및 AWS Marketplace 연결 유형 값 참조)

DynamoDB 연결

AWS Glue for Spark를 사용하여 AWS Glue에서 DynamoDB의 테이블을 읽고 쓸 수 있습니다. AWS Glue 작업에 연결된 IAM 권한을 사용하여 DynamoDB에 연결합니다. AWS Glue는 다른 AWS 계정의 DynamoDB 테이블에 데이터 쓰기를 지원합니다. 자세한 내용은 the section called “DynamoDB 테이블에 대한 교차 계정 교차 리전 액세스” 단원을 참조하십시오.

AWS Glue DynamoDB ETL 커넥터 외에도 DynamoDB `ExportTableToPointInTime` 요청을 호출하여 사용자가 제공한 Amazon S3 위치에 DynamoDB JSON 형식으로 저장하는 DynamoDB 내보내기 커넥터를 사용하여 DynamoDB에서 읽을 수 있습니다. 그런 다음 AWS Glue는 Amazon S3 내보내기 위치에서 데이터를 읽어서 DynamicFrame 객체를 생성합니다.

DynamoDB 기록기는 AWS Glue 버전 1.0 이상에서 지원됩니다. AWS Glue DynamoDB 내보내기 커넥터는 AWS Glue 버전 2.0 이상 버전에서 사용할 수 있습니다.

DynamoDB에 대한 자세한 내용은 Amazon DynamoDB 설명서를 참조하세요.

> **Note**
>
> DynamoDB ETL 리더는 필터 또는 푸시다운 조건자를 지원하지 않습니다.

DynamoDB 연결 구성

AWS Glue에서 DynamoDB에 연결하려면 AWS Glue 작업과 관련된 IAM 역할에 DynamoDB와 상호 작용할 수 있는 권한을 부여합니다. DynamoDB에서 읽거나 쓰는 데 필요한 권한에 대한 자세한 내용은 IAM 설명서에서 DynamoDB에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.

- DynamoDB 내보내기 커넥터를 사용하는 경우에는 작업에서 DynamoDB 테이블 내보내기를 요청할 수 있도록 IAM을 구성해야 합니다. 또한 내보내기를 위한 Amazon S3 버킷을 식별하고 DynamoDB가 이 버킷에 쓸 수 있고 AWS Glue 작업이 이 버킷에서 읽을 수 있도록 IAM에서 적절한 권한을 제공해야 합니다. 자세한 내용은 DynamoDB에서 테이블 내보내기 요청을 참조하세요.
- AWS Glue 작업에 특정 Amazon VPC 연결 요구 사항이 있는 경우 NETWORK AWS Glue 연결 유형을 사용하여 네트워크 옵션을 제공하세요. DynamoDB에 대한 액세스는 IAM에 의해 권한이 부여되므로 AWS Glue DynamoDB 연결 유형이 필요하지 않습니다.

DynamoDB에서 읽고 쓰기

다음 코드 예제에서는 ETL 커넥터를 통해 DynamoDB 테이블에서 읽고, DynamoDB 테이블에 쓰는 방법을 보여줍니다. 한 테이블에서 읽고 다른 테이블에 쓰는 것을 보여줍니다.

Python

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

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

dyf = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={"dynamodb.input.tableName": test_source,
        "dynamodb.throughput.read.percent": "1.0",
        "dynamodb.splits": "100"
    }
)
```

```
print(dyf.getNumPartitions())

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()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.input.tableName" -> test_source,
        "dynamodb.throughput.read.percent" -> "1.0",
        "dynamodb.splits" -> "100"
      ))
    ).getDynamicFrame()

    print(dynamicFrame.getNumPartitions())

    val dynamoDbSink: DynamoDbDataSink =  glueContext.getSinkWithFormat(
      connectionType = "dynamodb",
```

```
      options = JsonOptions(Map(
        "dynamodb.output.tableName" -> test_sink,
        "dynamodb.throughput.write.percent" -> "1.0"
      ))
    ).asInstanceOf[DynamoDbDataSink]

    dynamoDbSink.writeDynamicFrame(dynamicFrame)

    Job.commit()
  }

}
```

DynamoDB 내보내기 커넥터 사용

내보내기 커넥터는 DynamoDB 테이블 크기가 80GB보다 큰 경우 ETL 커넥터보다 성능이 우수합니다. 또한 내보내기 요청이 AWS Glue 작업의 Spark 프로세스 외부에서 수행된다는 점을 고려할 때, AWS Glue 작업의 Auto Scaling을 사용 설정하여 내보내기 요청이 진행되는 동안 DPU 사용량을 줄일할 수 있습니다. 내보내기 커넥터를 사용하면 Spark 실행기 병렬 처리 또는 DynamoDB 처리량 읽기 비율에 대한 분할 수를 구성할 필요가 없습니다.

> **Note**
>
> DynamoDB에는 `ExportTableToPointInTime` 요청을 호출하기 위한 특정 요구 사항이 있습니다. 자세한 내용은 DynamoDB에서 테이블 내보내기 요청을 참조하세요. 예를 들어, 이 커넥터를 사용하려면 테이블에서 PITR(특정 시점으로 복구)을 사용 설정해야 합니다. 또한 DynamoDB 커넥터는 Amazon S3으로의 DynamoDB 내보내기 작업에 대해 AWS KMS 암호화를 지원합니다. AWS Glue 작업 구성에서 보안 구성을 제공하면 DynamoDB 내보내기에 대한 AWS KMS 암호화가 사용 설정됩니다. KMS 키는 Amazon S3 버킷과 동일한 리전에 있어야 합니다.
>
> DynamoDB 내보내기 및 Amazon S3 스토리지 비용에 대한 추가 요금이 부과됩니다. Amazon S3에서 내보낸 데이터는 작업 실행이 완료된 후에도 유지되므로 추가 DynamoDB 내보내기 없이 다시 사용할 수 있습니다. 이 커넥터 사용 시 요구 사항은 테이블에 대해 PITR(특정 시점으로 복구)을 사용 설정하는 것입니다.
>
> DynamoDB ETL 커넥터 또는 내보내기 커넥터는 DynamoDB 소스에 적용할 필터 또는 푸시다운 조건자를 지원하지 않습니다.

다음 코드 예제에서는 내보내기 커넥터를 통해 파티션 수를 읽고, 파티션 수를 인쇄하는 방법을 보여줍니다.

Python

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

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

dyf = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": test_source,
        "dynamodb.s3.bucket": bucket_name,
        "dynamodb.s3.prefix": bucket_prefix,
        "dynamodb.s3.bucketOwner": account_id_of_bucket,
    }
)
print(dyf.getNumPartitions())

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {
```

```
  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> test_source,
        "dynamodb.s3.bucket" -> bucket_name,
        "dynamodb.s3.prefix" -> bucket_prefix,
        "dynamodb.s3.bucketOwner" -> account_id_of_bucket,
      ))
    ).getDynamicFrame()

    print(dynamicFrame.getNumPartitions())

    Job.commit()
  }

}
```

다음 예제에서는 내보내기 커넥터를 통해 dynamodb 분류가 있는 AWS Glue 데이터 카탈로그 테이블에서 읽기 작업을 수행하고 파티션 수를 인쇄하는 방법을 보여줍니다.

Python

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

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

dynamicFrame = glue_context.create_dynamic_frame.from_catalog(
        database=catalog_database,
        table_name=catalog_table_name,
```

```
        additional_options={
            "dynamodb.export": "ddb",
            "dynamodb.s3.bucket": s3_bucket,
            "dynamodb.s3.prefix": s3_bucket_prefix
        }
    )
print(dynamicFrame.getNumPartitions())

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getCatalogSource(
        database = catalog_database,
        tableName = catalog_table_name,
        additionalOptions = JsonOptions(Map(
            "dynamodb.export" -> "ddb",
            "dynamodb.s3.bucket" -> s3_bucket,
            "dynamodb.s3.prefix" -> s3_bucket_prefix
        ))
    ).getDynamicFrame()
    print(dynamicFrame.getNumPartitions())
)
```

DynamoDB 내보내기 JSON 사용 간소화

AWS Glue DynamoDB 내보내기 커넥터를 사용하여 DynamoDB 내보내기를 수행하면 특정 중첩 구조의 JSON 파일이 생성됩니다. 자세한 내용은 데이터 객체를 참조하세요. AWS Glue는 이러한 구조를 다운스트림 애플리케이션에서 사용하기 쉬운 형태로 중첩 해제를 수행할 수 있는 DynamicFrame 변환을 제공합니다.

두 가지 방법 중 하나로 변환을 호출할 수 있습니다. DynamoDB에서 읽을 메서드를 직접적으로 호출할 때 "true" 값을 사용하여 "dynamodb.simplifyDDBJson" 연결 옵션을 설정할 수 있습니다. AWS Glue 라이브러리에서 독립적으로 사용할 수 있는 메서드로 변환을 직접적으로 호출할 수도 있습니다.

DynamoDB 내보내기로 생성된 다음 스키마를 고려해 보세요.

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

simplifyDDBJson 변환은 이를 다음과 같이 간소화합니다.

```
root
```

```
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

> **Note**
>
> `simplifyDDBJson`은 AWS Glue 3.0 이상 버전에서 사용할 수 있습니다. `unnestDDBJson` 변환은 DynamoDB 내보내기 JSON을 간소화하는 데도 사용할 수 있습니다. 사용자가 `simplifyDDBJson`에서 `unnestDDBJson`으로 전환할 것을 권장합니다.

DynamoDB 작업의 병렬 처리 구성

성능을 향상시키기 위해 DynamoDB 커넥터에 사용할 수 있는 특정 파라미터를 조정할 수 있습니다. 병렬 처리 파라미터를 조정할 때 목표는 프로비저닝된 AWS Glue 작업자의 사용을 극대화하는 것입니다. 더 높은 성능이 필요한 경우 DPU 수를 늘려 작업을 스케일 아웃하는 것이 좋습니다.

ETL 커넥터를 사용할 때 `dynamodb.splits` 파라미터를 사용하여 DynamoDB 읽기 작업의 병렬 처리를 변경할 수 있습니다. 내보내기 커넥터를 사용하여 읽을 때는 Spark 실행기 병렬 처리를 위한 분할 횟수를 구성할 필요가 없습니다. `dynamodb.output.numParallelTasks`를 사용하여 DynamoDB 쓰기 작업의 병렬 처리를 변경할 수 있습니다.

DynamoDB ETL 커넥터를 사용하여 읽기

작업 구성에 설정된 최대 작업자 수와 다음 `numSlots` 계산을 기준으로 `dynamodb.splits`를 계산하는 것이 좋습니다. 자동 스케일링의 경우 실제 사용 가능한 작업자 수는 해당 한도 내에서 변경될 수 있습니다. 최대 작업자 수 설정에 대한 자세한 내용은 the section called “Spark 작업 속성 구성”의 작업자 수(`NumberOfWorkers`)를 참조하세요.

- `numExecutors = NumberOfWorkers - 1`

상황에 따라 한 실행기는 Spark 드라이버용으로 예약되어 있고 다른 실행기는 데이터 처리에 사용됩니다.

- numSlotsPerExecutor =

  AWS Glue 3.0 and later versions

  - WorkerType이 G.1X일 경우 4
  - WorkerType이 G.2X일 경우 8
  - WorkerType이 G.4X일 경우 16
  - WorkerType이 G.8X일 경우 32

  AWS Glue 2.0 and legacy versions

  - WorkerType이 G.1X일 경우 8
  - WorkerType이 G.2X일 경우 16
- numSlots = numSlotsPerExecutor * numExecutors

사용 가능한 슬롯 수인 numSlots에 dynamodb.splits를 설정하는 것이 좋습니다.

DynamoDB에 쓰기

dynamodb.output.numParallelTasks 파라미터는 다음 계산을 사용하여 Spark 태스크당 WCU를 결정하는 데 사용됩니다.

```
permittedWcuPerTask = ( TableWCU * dynamodb.throughput.write.percent ) / 
dynamodb.output.numParallelTasks
```

구성이 DynamoDB에 쓰는 Spark 태스크의 수를 정확하게 나타내는 경우 DynamoDB 작성기가 가장 잘 작동합니다. 경우에 따라 쓰기 성능을 개선하기 위해 기본 계산을 재정의해야 할 수 있습니다. 해당 파라미터를 지정하지 않으면 Spark 태스크당 허용되는 WCU는 다음 공식에 따라 자동으로 계산됩니다.

-
  - numPartitions = dynamicframe.getNumPartitions()
  - numSlots(이 섹션에서 이전에 정의한 대로)
  - numParallelTasks = min(numPartitions, numSlots)
- 예 1. DPU=10, WorkerType=Standard. 입력 DynamicFrame에는 100개의 RDD 파티션이 있습니다.
  - numPartitions = 100
  - numExecutors = (10 - 1) * 2 - 1 = 17

  - `numSlots = 4 * 17 = 68`
  - `numParallelTasks = min(100, 68) = 68`
- 예 2. DPU=10, WorkerType=Standard. 입력 DynamicFrame에는 20개의 RDD 파티션이 있습니다.
  - `numPartitions = 20`
  - `numExecutors = (10 - 1) * 2 - 1 = 17`
  - `numSlots = 4 * 17 = 68`
  - `numParallelTasks = min(20, 68) = 20`

> **Note**
>
> 레거시 AWS Glue 버전의 작업과 Standard 작업자를 사용하는 작업은 슬롯 수를 계산하는 방법이 다릅니다. 이러한 작업의 성능을 조정해야 하는 경우 지원되는 AWS Glue 버전으로 전환하는 것이 좋습니다.

DynamoDB 연결 옵션 참조

Amazon DynamoDB에 대한 연결을 지정합니다.

연결 옵션은 소스 연결 및 싱크 연결에 따라 다릅니다.

ETL 커넥터를 소스로 사용하는 "connectionType": "dynamodb"

AWS Glue DynamoDB ETL 커넥터 사용 시 `"connectionType": "dynamodb"`를 소스로 사용하여 다음 연결 옵션을 사용합니다.

- `"dynamodb.input.tableName"`: (필수 사항) 읽을 DynamoDB 테이블입니다.
- `"dynamodb.throughput.read.percent"`: (선택 사항) 사용할 읽기 용량 유닛(RCU)의 백분율. 기본값은 "0.5"로 설정되어 있습니다. 허용되는 값은 "0.1"에서 "1.5"(포함)입니다.
  - `0.5`는 기본 읽기 속도입니다. 즉, AWS Glue가 테이블 읽기 용량의 절반을 사용하려고 시도합니다. 값을 `0.5`보다 높게 설정할 경우 AWS Glue가 요청 속도를 높이고, 값을 `0.5`보다 낮게 설정할 경우 읽기 요청 속도를 낮춥니다. (실제 읽기 속도는 DynamoDB 테이블의 키 분포가 균일한지 여부와 같은 요소에 따라 달라집니다.)
  - DynamoDB 테이블이 온디맨드 모드인 경우 AWS Glue는 테이블의 읽기 용량을 40,000으로 처리합니다. 큰 테이블을 내보내려면 DynamoDB 테이블을 온디맨드 모드로 전환하는 것이 좋습니다.

- "dynamodb.splits": (선택 사항) 읽는 동안 이 DynamoDB 테이블을 몇 개의 분할로 파티셔닝하는지 정의합니다. 기본값은 "1"로 설정되어 있습니다. 허용되는 값은 "1"에서 "1,000,000"(포함)입니다.

  1은 병렬 처리가 없음을 나타냅니다. 아래 공식을 사용하여 더 나은 성능을 위해 더 큰 값을 지정하는 것이 좋습니다. 값을 적절하게 설정하는 방법에 대한 자세한 내용은 the section called "DynamoDB 병렬 처리" 섹션을 참조하세요.
- "dynamodb.sts.roleArn": (선택 사항) 교차 계정 액세스를 위해 수임할 IAM 역할 ARN입니다. 이 파라미터는 AWS Glue 1.0 이상에서 사용 가능합니다.
- "dynamodb.sts.roleSessionName": (선택 사항) STS 세션 이름입니다. 기본값은 "glue-dynamodb-read-sts-session"으로 설정됩니다. 이 파라미터는 AWS Glue 1.0 이상에서 사용 가능합니다.

AWS Glue DynamoDB 내보내기 커넥터를 소스로 사용하는 "connectionType": "dynamodb"

AWS Glue 버전 2.0 이상에서만 사용할 수 있는 AWS Glue DynamoDB 내보내기 커넥터 사용 시 "connectionType": "dynamodb"를 소스로 사용하는 다음 연결 옵션을 사용합니다.

- "dynamodb.export": (필수) 문자열 값:
  - ddb로 설정한 경우 AWS Glue 작업 중 새 ExportTableToPointInTimeRequest가 호출되는 AWS Glue DynamoDB 내보내기 커넥터가 사용 설정됩니다. dynamodb.s3.bucket 및 dynamodb.s3.prefix에서 전달된 위치로 새 내보내기가 생성됩니다.
  - s3로 설정한 경우 AWS Glue DynamoDB 내보내기 커넥터가 사용 설정되지만, 새 DynamoDB 내보내기 생성을 건너뛰고 대신 dynamodb.s3.bucket 및 dynamodb.s3.prefix를 해당 테이블의 과거 내보내기 Amazon S3 위치로 사용합니다.
- "dynamodb.tableArn": (필수 사항) 읽을 DynamoDB 테이블입니다.
- "dynamodb.unnestDDBJson": (선택 사항) 기본값: false. 유효 값: 부울. true로 설정된 경우 내보내기에 있는 DynamoDB JSON 구조의 비중첩 변환을 수행합니다. "dynamodb.unnestDDBJson"과 "dynamodb.simplifyDDBJson"을 동시에 true로 설정하면 오류가 발생합니다. AWS Glue 3.0 이상 버전에서는 DynamoDB 맵 유형을 간소화할 때 더 나은 동작을 위해 "dynamodb.simplifyDDBJson"을 사용하는 것이 좋습니다. 자세한 내용은 the section called "DynamoDB 내보내기 JSON 사용 간소화" 단원을 참조하십시오.
- "dynamodb.simplifyDDBJson": (선택 사항) 기본값: false. 유효 값: 부울. true로 설정하면 변환을 수행하여 내보내기에 있는 DynamoDB JSON 구조의 스키마를 간소화합니다. "dynamodb.unnestDDBJson" 옵션과 용도는 동일하지만 DynamoDB 맵 유형 또는 DynamoDB 테이블의 중첩된 맵 유형에 대한 지원을 강화합니다. 이 옵션은 AWS Glue 3.0 이상 버전에서 사용할

수 있습니다. "dynamodb.unnestDDBJson"과 "dynamodb.simplifyDDBJson"을 동시에 true로 설정하면 오류가 발생합니다. 자세한 내용은 the section called “DynamoDB 내보내기 JSON 사용 간소화” 단원을 참조하십시오.

- "dynamodb.s3.bucket": (선택 사항) DynamoDB ExportTableToPointInTime 프로세스가 수행될 Amazon S3 버킷 위치를 나타냅니다. 내보내기의 파일 형식은 DynamoDB JSON입니다.
  - "dynamodb.s3.prefix": (선택 사항) DynamoDB ExportTableToPointInTime 로드가 저장될 Amazon S3 버킷 내부의 Amazon S3 접두사 위치를 나타냅니다. dynamodb.s3.prefix와 dynamodb.s3.bucket을 모두 지정하지 않을 경우 이 값은 기본적으로 AWS Glue 작업 구성에 지정된 임시 디렉터리 위치로 설정됩니다. 자세한 내용은 AWS Glue가 사용하는 특수 파라미터를 참조하세요.
  - "dynamodb.s3.bucketOwner": 교차 계정 Amazon S3 액세스를 위해 필요한 버킷 소유자를 나타냅니다.
- "dynamodb.sts.roleArn": (선택 사항) DynamoDB 테이블에 대한 교차 계정 액세스 및/또는 교차 리전 액세스에 대해 수임될 IAM 역할 ARN입니다. 참고: 동일한 IAM 역할 ARN은 ExportTableToPointInTime 요청에 대해 지정된 Amazon S3 위치에 액세스하는 데 사용됩니다.
- "dynamodb.sts.roleSessionName": (선택 사항) STS 세션 이름입니다. 기본값은 "glue-dynamodb-read-sts-session"으로 설정됩니다.
- "dynamodb.exportTime"(선택 사항) 유효 값: ISO-8601 인스턴트를 나타내는 문자열. 내보내기를 수행해야 하는 시점입니다.
- "dynamodb.sts.region": (리전 엔드포인트를 사용하여 리전 간 직접 호출하는 경우 필수임) 읽으려는 DynamoDB 테이블을 호스팅하는 지역입니다.

싱크로 ETL 커넥터를 사용하는 "connectionType": "dynamodb"

싱크로서의 "connectionType": "dynamodb"에는 다음 연결 옵션을 사용합니다.

- "dynamodb.output.tableName": (필수) 쓸 대상 DynamoDB 테이블입니다.
- "dynamodb.throughput.write.percent": (선택 사항) 사용할 쓰기 용량 유닛(WCU)의 백분율입니다. 기본값은 "0.5"로 설정되어 있습니다. 허용되는 값은 "0.1"에서 "1.5"(포함)입니다.
  - 0.5는 기본 쓰기 속도입니다. 즉, AWS Glue가 테이블 쓰기 용량의 절반을 사용하려고 시도합니다. 값을 0.5보다 높게 설정할 경우 AWS Glue가 요청 속도를 높이고, 값을 0.5보다 낮게 설정할 경우 쓰기 요청 속도를 낮춥니다. (실제 쓰기 속도는 DynamoDB 테이블의 키 분포가 균일한지 여부와 같은 요소에 따라 달라집니다.)

  - DynamoDB 테이블이 온디맨드 모드인 경우 AWS Glue는 테이블의 쓰기 용량을 40000으로 처리합니다. 큰 테이블을 가져오려면 DynamoDB 테이블을 온디맨드 모드로 전환하는 것이 좋습니다.
- "dynamodb.output.numParallelTasks": (선택 사항) 동시에 DynamoDB에 쓰는 병렬 태스크 수를 정의합니다. Spark 태스크당 허용되는 WCU를 계산하는 데 사용됩니다. 대부분의 경우 AWS Glue는 해당 값에 대해 적절한 기본값을 계산합니다. 자세한 내용은 the section called “DynamoDB 병렬 처리” 단원을 참조하십시오.
- "dynamodb.output.retry": (선택 사항) DynamoDB에서 ProvisionedThroughputExceededException이 있을 때 수행할 재시도 횟수를 정의합니다. 기본값은 "10"으로 설정됩니다.
- "dynamodb.sts.roleArn": (선택 사항) 교차 계정 액세스를 위해 수임할 IAM 역할 ARN입니다.
- "dynamodb.sts.roleSessionName": (선택 사항) STS 세션 이름입니다. 기본값은 "glue-dynamodb-write-sts-session"으로 설정됩니다.

DynamoDB 테이블에 대한 교차 계정 교차 리전 액세스

AWS Glue ETL 작업은 DynamoDB 테이블에 대한 교차 리전 및 교차 계정 액세스를 지원합니다. AWS Glue ETL 작업은 다른 AWS 계정의 DynamoDB 테이블에서 데이터 읽기와 다른 AWS 계정의 DynamoDB 테이블에 데이터 쓰기를 모두 지원합니다. AWS Glue는 또한 다른 리전의 DynamoDB 테이블에서 읽기와 다른 리전의 DynamoDB 테이블에 쓰기를 모두 지원합니다. 이 섹션에서는 액세스 설정에 대한 지침과 예제 스크립트를 제공합니다.

이 섹션의 절차에서는 IAM 역할을 생성하고 역할에 대한 액세스 권한을 부여하는 IAM 튜토리얼을 참조합니다. 이 튜토리얼에서는 역할 수임에 대해서도 설명하지만 여기서는 대신 작업 스크립트를 사용하여 AWS Glue에서 역할을 수임합니다. 이 튜토리얼에는 일반적인 교차 계정 사례에 대한 정보도 들어 있습니다. 자세한 내용은 IAM User Guide의 Tutorial: Delegate Access Across AWS Accounts Using IAM Roles를 참조하세요.

역할 생성

튜토리얼의 1단계에 따라 계정 A에서 IAM 역할을 생성합니다. 역할의 권한을 정의할 때 역할이 DynamoDB를 읽고 쓸 수 있도록 AmazonDynamoDBReadOnlyAccess 또는 AmazonDynamoDBFullAccess 등의 기존 정책을 연결하도록 선택할 수 있습니다. 다음 예에서는 권한 정책 AmazonDynamoDBFullAccess를 사용하여 DynamoDBCrossAccessRole이라는 역할을 생성하는 방법을 보여줍니다.

역할에 대한 액세스 권한 부여

IAM User Guide의 튜토리얼의 1단계에 따라 계정 B가 새로 생성된 역할로 전환하도록 허용할 수 있습니다. 다음 예에서는 다음 문을 사용하여 새 정책을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "<DynamoDBCrossAccessRole's ARN>"
    }
}
```

그런 다음 이 정책을 DynamoDB에 액세스하는 데 사용할 그룹/역할/사용자에 연결할 수 있습니다.

AWS Glue 작업 스크립트에서 역할 수임

이제 B 계정에 로그인하여 AWS Glue 작업을 생성할 수 있습니다. 작업을 생성하려면 AWS Glue에서 Spark 작업에 대한 작업 속성 구성의 지침을 참조하세요.

작업 스크립트에서 `DynamoDBCrossAccessRole` 역할을 수임하려면 `dynamodb.sts.roleArn` 파라미터를 사용해야 합니다. 이 역할을 수임하면 계정 B의 DynamoDB에 액세스하는 데 필요한 임시 자격 증명을 얻을 수 있습니다. 다음 예제 스크립트를 검토하세요.

리전 간 교차 계정 읽기의 경우(ETL 커넥터):

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

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

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
    "dynamodb.region": "us-east-1",
```

```
    "dynamodb.input.tableName": "test_source",
    "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
    }
)
dyf.show()
job.commit()
```

리전 간 교차 계정 읽기의 경우(ELT 커넥터):

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

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

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": "<test_source ARN>",
        "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
    }
)
dyf.show()
job.commit()
```

리전 간 읽기 및 교차 계정 쓰기의 경우:

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

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)
```

```
dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.region": "us-east-1",
        "dynamodb.input.tableName": "test_source"
    }
)
dyf.show()

glue_context.write_dynamic_frame_from_options(
    frame=dyf,
    connection_type="dynamodb",
    connection_options={
        "dynamodb.region": "us-west-2",
        "dynamodb.output.tableName": "test_sink",
        "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
    }
)

job.commit()
```

Kinesis 연결

Data Catalog 테이블에 저장된 정보를 사용하거나 데이터 스트림에 직접 액세스할 수 있는 정보를 제공하여 Kinesis 연결을 통해 Amazon Kinesis 데이터 스트림에서 읽고 쓸 수 있습니다. Kinesis의 정보를 Spark DataFrame으로 읽은 다음 AWS Glue DynamicFrame으로 변환할 수 있습니다. DynamicFrame을 JSON 형식으로 Kinesis에 쓸 수 있습니다. 데이터 스트림에 직접 액세스하는 경우 이러한 옵션을 사용하여 데이터 스트림에 액세스하는 방법에 대한 정보를 제공합니다.

`getCatalogSource` 또는 `create_data_frame_from_catalog`를 통해 Kinesis 스트리밍 소스의 레코드를 사용하는 경우 작업에 데이터 카탈로그 데이터베이스 및 테이블 이름 정보가 있으며, 해당 정보를 사용하여 Kinesis 스트리밍 소스에서 읽기 위한 몇 가지 기본 파라미터를 얻을 수 있습니다. `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` 또는 `create_data_frame_from_options`를 사용하는 경우 여기에 설명된 연결 옵션을 통해 이러한 기본 파라미터를 지정해야 합니다.

`GlueContext` 클래스에 지정된 메서드에 대해 다음 인수를 사용하여 Kinesis에 대한 연결 옵션을 지정할 수 있습니다.

- Scala

  - `connectionOptions`: `getSource`, `createDataFrameFromOptions`, `getSink`와(과) 함께 사용
  - `additionalOptions`: `getCatalogSource`, `getCatalogSink`와 함께 사용
  - `options`: `getSourceWithFormat`, `getSinkWithFormat`와 함께 사용
- Python
  - `connection_options`: `create_data_frame_from_options`, `write_dynamic_frame_from_options`와 함께 사용
  - `additional_options`: `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`와 함께 사용
  - `options`: `getSource`, `getSink`와 함께 사용

스트리밍 ETL 작업에 대한 참고 및 제한 사항은 the section called “스트리밍 ETL 참고 사항 및 제한 사항” 섹션을 참조하세요.

Kinesis 구성

AWS Glue Spark 작업에서 Kinesis 데이터 스트림을 연결하려면 몇 가지 필수 조건이 필요합니다.

- 읽는 경우 AWS Glue 작업에는 Kinesis 데이터 스트림에 대한 읽기 액세스 수준의 IAM 권한이 있어야 합니다.
- 쓰는 경우 AWS Glue 작업에는 Kinesis 데이터 스트림에 대한 쓰기 액세스 수준의 IAM 권한이 있어야 합니다.

경우에 따라 추가 필수 조건을 구성해야 합니다.

- AWS Glue 작업이 추가 네트워크 연결(일반적으로 다른 데이터 세트에 연결)로 구성되었고 이러한 연결 중 하나에서 Amazon VPC 네트워크 옵션을 제공하는 경우 작업에 Amazon VPC를 통해 통신하도록 지시합니다. 이 경우 Amazon VPC를 통해 통신하도록 Kinesis 데이터 스트림도 구성해야 합니다. Amazon VPC와 Kinesis 데이터 스트림 사이에서 인터페이스 VPC 엔드포인트를 생성하면 됩니다. 자세한 내용은 Using Kinesis Data Streams with Interface VPC Endpoints를 참조하세요.
- 다른 계정에서 Amazon Kinesis Data Streams를 지정할 때 크로스 계정 액세스를 허용하도록 역할과 정책을 설정해야 합니다. 자세한 내용은 예: 다른 계정의 Kinesis 스트림에서 읽기를 참조하세요.

스트리밍 ETL 작업 필수 조건에 대한 자세한 내용은 the section called “스트리밍 ETL 작업” 섹션을 참조하세요.

예제: Kinesis 스트림에서 읽기

예제: Kinesis 스트림에서 읽기

the section called “forEachBatch”과(와) 함께 사용합니다.

Amazon Kinesis 스트리밍 소스의 예:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

예: Kinesis 스트림에 쓰기

예제: Kinesis 스트림에서 읽기

the section called “forEachBatch”과(와) 함께 사용합니다.

Amazon Kinesis 스트리밍 소스의 예:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Kinesis 연결 옵션 참조

Amazon Kinesis Data Streams에 대한 연결 옵션을 지정합니다.

Kinesis 스트리밍 데이터 원본에 대한 다음 연결 옵션을 사용합니다.

- `"streamARN"` (필수) 읽기/쓰기에 사용됩니다. Kinesis 데이터 스트림의 ARN.

- "classification" (읽기 필수) 읽기에 사용됩니다. 레코드의 데이터에서 사용하는 파일 형식. 데이터 카탈로그를 통해 제공되지 않는 한 필수입니다.
- "streamName" – (선택 사항) 읽기에 사용됩니다. 읽을 Kinesis 데이터 스트림의 이름. endpointUrl와(과) 함께 사용됩니다.
- "endpointUrl" – (선택 사항) 읽기에 사용됩니다. 기본값: "https://kinesis.us-east-1.amazonaws.com". Kinesis 스트림의 AWS 엔드포인트. 특정 지역에 연결하는 경우가 아니면 변경하지 않아도 됩니다.
- "partitionKey" – (선택 사항) 쓰기에 사용됩니다. 레코드를 생성할 때 사용되는 Kinesis 파티션 키.
- "delimiter" (선택 사항) 읽기에 사용됩니다. classification이(가) CSV일 때 사용되는 값 구분 기호입니다. 기본값은 ','입니다.
- "startingPosition": (선택 사항) 읽기에 사용됩니다. 데이터를 읽을 Kinesis 데이터 스트림의 시작 위치입니다. 가능한 값은 yyyy-mm-ddTHH:MM:SSZ 패턴에서 UTC 형식의 타임스탬프 문자열이나 "latest", "trim_horizon" 또는 "earliest"입니다(여기서, Z는 UTC 시간대 오프셋(+/-)임, 예: '2023-04-04T08:00:00-04:00'). 기본값은 "latest"입니다. 참고: "startingPosition"에 대한 UTC 형식의 타임스탬프 문자열은 AWS Glue 버전 4.0 이상에서만 지원됩니다.
- "failOnDataLoss": (선택 사항) 활성 샤드가 누락되거나 만료된 경우 작업이 실패합니다. 기본값은 "false"입니다.
- "awsSTSRoleARN": (선택 사항) 읽기/쓰기에 사용됩니다. AWS Security Token Service(AWS STS)을(를) 사용하여 맡을 역할의 Amazon 리소스 이름(ARN). 이 역할에는 Kinesis 데이터 스트림에 대한 레코드 작업을 설명하거나 읽을 수 있는 권한이 있어야 합니다. 다른 계정의 데이터 스트림에 액세스할 때 이 파라미터를 사용해야 합니다. "awsSTSSessionName"과(와) 함께 사용합니다.
- "awsSTSSessionName": (선택 사항) 읽기/쓰기에 사용됩니다. AWS STS을(를) 사용하여 역할을 맡는 세션의 식별자입니다. 다른 계정의 데이터 스트림에 액세스할 때 이 파라미터를 사용해야 합니다. "awsSTSRoleARN"과(와) 함께 사용합니다.
- "awsSTSEndpoint": (선택 사항) 위임 받은 역할을 사용하여 Kinesis에 연결할 때 사용할 AWS STS 엔드포인트. 이를 통해 VPC에서 리전의 AWS STS 엔드포인트를 사용할 수 있지만, 기본 글로벌 엔드포인트로는 불가능합니다.
- "maxFetchTimeInMs": (선택 사항) 읽기에 사용됩니다. 작업 실행기가 Kinesis 데이터 스트림에서 현재 배치에 대한 레코드를 읽는 데 걸리는 최대 시간(밀리초(ms) 단위로 지정)입니다. 이 시간 내에 여러 개의 GetRecords API 호출을 할 수 있습니다. 기본값은 1000입니다.
- "maxFetchRecordsPerShard": (선택 사항) 읽기에 사용됩니다. 마이크로 배치에 따라 Kinesis 데이터 스트림에서 샤드당 가져올 최대 레코드 수입니다. 참고: 스트리밍 작업이 이미 Kinesis의 동일한 get-records 호출에서 추가 레코드를 읽은 경우 클라이언트가 이 제한을 초과할 수 있습니다.

maxFetchRecordsPerShard가 엄격해야 한다면 maxRecordPerRead의 배수여야 합니다. 기본값은 100000입니다.

- "maxRecordPerRead": (선택 사항) 읽기에 사용됩니다. 각 getRecords 작업에서 Kinesis 데이터 스트림에서 가져올 최대 레코드 수. 기본값은 10000입니다.
- "addIdleTimeBetweenReads": (선택 사항) 읽기에 사용됩니다. 연속 두 getRecords 작업 사이에 시간 지연을 추가합니다. 기본값은 "False"입니다. 이 옵션은 Glue 버전 2.0 이상에서만 구성할 수 있습니다.
- "idleTimeBetweenReadsInMs": (선택 사항) 읽기에 사용됩니다. 연속 두 getRecords 작업 사이의 최소 시간 지연으로, ms 단위로 지정됩니다. 기본값은 1000입니다. 이 옵션은 Glue 버전 2.0 이상에서만 구성할 수 있습니다.
- "describeShardInterval": (선택 사항) 읽기에 사용됩니다. 스크립트가 리샤딩을 고려하기 위한 두 ListShards API 호출 사이의 최소 시간 간격. 자세한 내용은 Amazon Kinesis Data Streams Developer Guide의 Strategies for Resharding을 참조하세요. 기본값은 1s입니다.
- "numRetries": (선택 사항) 읽기에 사용됩니다. Kinesis Data Streams API 요청의 최대 재시도 횟수입니다. 기본값은 3입니다.
- "retryIntervalMs": (선택 사항) 읽기에 사용됩니다. Kinesis Data Streams API 호출을 재시도하기 전의 휴지 기간(ms 단위로 지정)입니다. 기본값은 1000입니다.
- "maxRetryIntervalMs": (선택 사항) 읽기에 사용됩니다. Kinesis Data Streams API 호출을 두 번 재시도하는 사이의 최대 휴지 시간(ms 단위로 지정)입니다. 기본값은 10000입니다.
- "avoidEmptyBatches": (선택 사항) 읽기에 사용됩니다. 배치가 시작되기 전에 Kinesis 데이터 스트림에서 읽지 않은 데이터를 확인하여 빈 마이크로 배치 작업 생성을 방지합니다. 기본값은 "False"입니다.
- "schema": (inferSchema가 거짓으로 설정된 경우 필수 사항) 읽기에 사용됩니다. 페이로드를 처리하는 데 사용하는 스키마. 분류가 avro인 경우 제공된 스키마는 Avro 스키마 형식이어야 합니다. 분류가 avro가 아닌 경우 제공된 스키마는 DDL 스키마 형식이어야 합니다.

  다음은 스키마의 예입니다.

  Example in DDL schema format

  ```
  `column1` INT, `column2` STRING , `column3` FLOAT
  ```

  Example in Avro schema format

  ```
  {
    "type":"array",
  ```

```
  "items":
  {
    "type":"record",
    "name":"test",
    "fields":
    [
      {
        "name":"_id",
        "type":"string"
      },
      {
        "name":"index",
        "type":
        [
          "int",
          "string",
          "float"
        ]
      }
    ]
  }
}
```

- "inferSchema": (선택 사항) 읽기에 사용됩니다. 기본값은 'false'입니다. 'true'로 설정하면 스키마가 런타임 시 foreachbatch 내의 페이로드에서 감지됩니다.
- "avroSchema": (더 이상 사용되지 않음) 읽기에 사용됩니다. Avro 형식을 사용할 때 Avro 데이터의 스키마를 지정하는 데 사용되는 파라미터입니다. 이 파라미터는 이제 사용 중단되었습니다. schema 파라미터를 사용합니다.
- "addRecordTimestamp": (선택 사항) 읽기에 사용됩니다. 이 옵션이 'true'로 설정되면 데이터 출력에는 이름이 '__src_timestamp'라는 추가 열이 포함됩니다. 이 열은 스트림에서 해당 레코드를 수신한 시간을 나타냅니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.
- "emitConsumerLagMetrics": (선택 사항) 읽기에 사용됩니다. 옵션을 '참'으로 설정하면 각 배치에 대해 스트림에서 수신한 가장 오래된 레코드와 AWS Glue에 도착한 시간 사이의 지표를 CloudWatch로 내보냅니다. 지표의 이름은 'glue.driver.streaming.maxConsumerLagInMs'입니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.
- "fanoutConsumerARN": (선택 사항) 읽기에 사용됩니다. streamARN에서 지정된 스트림에 대한 Kinesis 스트림 소비자의 ARN. Kinesis 연결에 대해 향상된 팬아웃 모드를 활성화하는 데 사용됩니다. 향상된 팬아웃과 함께 Kinesis 스트림을 사용하는 방법에 대한 자세한 내용은 the section called “Kinesis 스트리밍 작업에서 향상된 팬아웃 사용” 섹션을 참조하세요.

- `"recordMaxBufferedTime"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 1000(ms). 레코드 쓰기를 대기하는 중 레코드가 버퍼링되는 최대 시간.
- `"aggregationEnabled"` – (선택 사항) 쓰기에 사용됩니다. 기본값: true. Kinesis로 보내기 전에 레코드를 집계해야 하는지 여부를 지정합니다.
- `"aggregationMaxSize"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 51200(바이트). 레코드가 이 한도보다 크면 애그리게이터를 우회합니다. 참고 Kinesis는 레코드 크기를 50KB로 제한합니다. 이 값이 50KB를 초과하도록 설정하면 Kinesis에서 크기가 초과된 레코드를 거부합니다.
- `"aggregationMaxCount"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 4294967295. 집계된 레코드에 포함할 최대 항목 수.
- `"producerRateLimit"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 150(%). 단일 생성자(예: 작업)에서 보내는 샤드당 처리량을 백엔드 한도의 백분율로 제한합니다.
- `"collectionMaxCount"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 500. PutRecords 요청에 포함할 최대 항목 수.
- `"collectionMaxSize"` – (선택 사항) 쓰기에 사용됩니다. 기본값: 5242880(바이트). PutRecords 요청으로 전송할 수 있는 데이터 최대량.

Kinesis 스트리밍 작업에서 향상된 팬아웃 사용

향상된 팬아웃 소비자는 일반 소비자보다 더 높은 전용 처리량으로 Kinesis 스트림에서 레코드를 수신할 수 있습니다. 이는 Kinesis 소비자에게 데이터(예: 작업)를 제공하는 데 사용되는 전송 프로토콜을 최적화하는 방식으로 지원됩니다. Kinesis의 향상된 팬아웃에 대한 자세한 내용은 Kinesis 설명서를 참조하세요.

향상된 팬아웃 모드에서는 `maxRecordPerRead` 및 `idleTimeBetweenReadsInMs` 연결 옵션이 더 이상 적용되지 않습니다. 향상된 팬아웃을 사용할 때 해당 파라미터는 구성할 수 없기 때문입니다. 재시도를 위한 구성 옵션은 설명한 대로 수행됩니다.

다음 절차를 사용하여 스트리밍 작업에 대한 향상된 팬아웃을 활성화 및 비활성화합니다. 스트림에서 데이터를 소비하는 각 작업에 대해 스트림 소비자를 등록해야 합니다.

작업에서 향상된 팬아웃 소비를 활성화하려면:

1. Kinesis API를 사용하여 작업에 대한 스트림 소비자를 등록합니다. Kinesis 설명서의 register a consumer with enhanced fan-out using the Kinesis Data Streams API에 대한 지침을 따릅니다. 첫 번째 단계인 RegisterStreamConsumer를 직접적으로 호출합니다. 요청은 ARN, *`consumerARN`*을 반환해야 합니다.

2. 연결 메서드 인수에서 `fanoutConsumerARN` 연결 옵션을 *`consumerARN`*으로 설정합니다.
3. 작업을 다시 시작합니다.

작업에서 향상된 팬아웃 소비를 비활성화하려면:

1. 메서드 직접 호출에서 `fanoutConsumerARN` 연결 옵션을 제거합니다.
2. 작업을 다시 시작합니다.
3. Kinesis 설명서의 deregister a consumer에 대한 지침을 따릅니다. 이 지침은 콘솔에도 적용되지만 Kinesis API를 통해서도 수행할 수 있습니다. Kinesis API를 통한 스트림 소비자 등록 취소에 대한 자세한 내용은 Kinesis 설명서의 DeregisterStreamConsumer를 참조하세요.

Amazon S3 연결

AWS Glue for Spark를 사용하여 Amazon S3에서 파일을 읽고 쓸 수 있습니다. AWS Glue for Spark는 CSV, Avro, JSON, Orc, Parquet을 포함하여 기본적으로 Amazon S3에 저장되는 여러 가지 공통 데이터 형식을 지원합니다. 지원되는 데이터 형식에 대한 자세한 내용은 the section called “데이터 포맷 옵션” 섹션을 참조하세요. 각 데이터 형식은 AWS Glue 기능의 여러 가지 세트를 지원할 수 있습니다. 특정 기능 지원에 대해서는 해당 데이터 형식 페이지를 참조하세요. 또한 Hudi, Iceberg 및 Delta Lake 데이터 레이크 프레임워크에 저장되는 버전 관리되는 파일을 읽고 쓸 수 있습니다. 데이터 레이크 프레임워크에 대한 자세한 내용은 the section called “데이터 레이크 프레임워크” 섹션을 참조하세요.

AWS Glue를 사용하면 쓰는 동안 Amazon S3 객체를 폴더 구조로 파티셔닝한 다음 파티션별로 검색하여 간단한 구성으로 성능을 개선할 수 있습니다. 성능을 개선하기 위해 데이터를 변환할 때 작은 파일을 함께 그룹화하도록 구성을 설정할 수도 있습니다. Amazon S3에서 `bzip2` 및 `gzip` 아카이브를 읽고 쓸 수 있습니다.

주제

- S3 연결 구성
- Amazon S3 연결 옵션 참조
- 데이터 형식에 대한 연결 구문은 더 이상 사용되지 않습니다.
- Amazon S3 스토리지 클래스 제외
- AWS Glue에서 ETL 출력의 파티션 관리
- 입력 파일을 더 큰 그룹에서 읽기
- Amazon S3용 Amazon VPC 엔드포인트

S3 연결 구성

AWS Glue에서 Spark 작업으로 Amazon S3에 연결하려면 몇 가지 필수 조건이 필요합니다.

- AWS Glue 작업에는 관련 Amazon S3 버킷에 대한 IAM 권한이 있어야 합니다.

경우에 따라 추가 필수 조건을 구성해야 합니다.

- 크로스 계정 액세스를 구성할 때 Amazon S3 버킷에서 적절한 액세스 제어.
- 보안상의 이유로 Amazon S3 요청을 Amazon VPC를 통해 라우팅하도록 선택할 수 있습니다. 단, 이러한 접근 방식은 대역폭 및 가용성 문제를 일으킬 수 있습니다. 자세한 내용은 the section called “Amazon S3용 Amazon VPC 엔드포인트” 단원을 참조하십시오.

Amazon S3 연결 옵션 참조

Amazon S3에 대한 연결을 지정합니다.

Amazon S3에서는 테이블이 아닌 파일을 관리하므로 이 문서에 제공된 연결 속성을 지정하는 것 외에도 파일 유형에 대한 추가 구성을 지정해야 합니다. 데이터 형식 옵션을 통해 이 정보를 지정합니다. 형식 옵션에 대한 자세한 내용은 the section called “데이터 포맷 옵션” 섹션을 참조하세요. AWS Glue 데이터 카탈로그와 통합하여 이 정보를 지정할 수도 있습니다.

연결 옵션과 형식 옵션 간의 차이점에 대한 예로, the section called “create_dynamic_frame_from_options” 메서드에서 `connection_type`, `connection_options`, `format` 및 `format_options`를 사용하는 방법을 고려합니다. 이 섹션에서는 `connection_options`에 제공된 파라미터에 대해 자세히 논의합니다.

`"connectionType": "s3"`에는 다음 연결 옵션을 사용합니다.

- `"paths"`: (필수 사항) 읽을 소스 Amazon S3 경로 목록입니다.
- `"exclusions"`: (선택사항) 배제할 Unix 식 glob 패턴의 JSON 목록이 포함된 문자열. 예를 들어 `"[\"**.pdf\"]"`는 모든 PDF 파일을 제외합니다. AWS Glue에서 지원하는 glob 구문에 대한 자세한 내용은 포함 및 제외 패턴을 참조하십시오.
- `"compressionType"`: 또는 `"compression"`: (선택 사항) 데이터 압축 방식을 지정합니다. Amazon S3 소스에는 `"compressionType"`을 사용하고 Amazon S3 대상에는 `"compression"`을 사용합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip2"`입니다). 특정 형식에 대해 추가 압축 형식이 지원될 수 있습니다. 자세한 기능 지원은 데이터 형식 페이지를 참조하세요.

- "groupFiles": (선택 사항) 입력에 50,000개 이상의 파일이 포함된 경우에는 기본적으로 파일 그룹화가 설정됩니다. 파일이 50,000개 미만일 때 그룹화를 설정하려면 이 파라미터를 "inPartition"으로 설정합니다. 파일이 50,000개 이상일 때 그룹화를 비활성화하려면 이 파라미터를 "none"으로 설정합니다.
- "groupSize": (선택사항) 대상 그룹 크기(바이트). 입력 데이터 크기와 클러스터 크기에 따라 기본값을 계산합니다. 입력 파일이 50,000개 미만일 때는 "groupFiles"을 "inPartition"으로 설정해야 적용됩니다.
- "recurse": (선택사항) true로 설정할 경우 지정된 경로의 모든 하위 디렉터리에 있는 파일을 반복적으로 읽습니다.
- "maxBand": (선택 사항, 고급) 이 옵션은 s3 목록이 일정할 것으로 예상하는 시간(밀리초)을 제어합니다. 마지막 maxBand밀리초에 속하는 수정 타임스탬프가 있는 파일은 특히 JobBookmarks를 사용하여 Amazon S3 최종 일관성을 고려할 때 추적됩니다. 대부분의 사용자는 이 옵션을 설정할 필요가 없습니다. 기본값은 900,000밀리초 또는 15분입니다.
- "maxFilesInBand": (선택 사항, 고급) 이 옵션은 마지막 maxBand초부터 저장할 최대 파일 수를 지정합니다. 이 수를 초과할 경우 추가 파일은 건너뛰고 다음 작업 실행에서만 처리됩니다. 대부분의 사용자는 이 옵션을 설정할 필요가 없습니다.
- "isFailFast": (선택 사항) 이 옵션은 AWS Glue ETL 작업에서 리더 구문 분석 예외가 발생하는지 여부를 결정합니다. true로 설정하면 Spark 태스크의 4번 재시도가 데이터를 제대로 구문 분석하지 못하면 작업이 빠르게 실패합니다.
- "catalogPartitionPredicate": (선택 사항) 읽기에 사용됩니다. SQL WHERE 절의 콘텐츠입니다. 파티션 수가 매우 많은 데이터 카탈로그 테이블에서 읽을 때 사용됩니다. 데이터 카탈로그 인덱스에서 일치하는 파티션을 검색합니다. the section called “create_dynamic_frame_from_catalog” 메서드 및 기타 유사한 메서드의 옵션인 push_down_predicate와 함께 사용됩니다. 자세한 내용은 the section called “카탈로그 파티션 조건자” 단원을 참조하십시오.
- "partitionKeys": (선택 사항) 쓰기에 사용됩니다. 열 레이블 문자열의 배열입니다. AWS Glue는 이 구성에 지정된 대로 데이터를 파티셔닝합니다. 자세한 내용은 the section called “파티션 작성” 단원을 참조하십시오.
- "excludeStorageClasses": (선택 사항) 읽기에 사용됩니다. Amazon S3 스토리지 클래스를 지정하는 문자열 배열입니다. AWS Glue는 이 구성을 기반으로 Amazon S3 객체를 제외합니다. 자세한 내용은 the section called “Amazon S3 스토리지 클래스 제외” 단원을 참조하십시오.

데이터 형식에 대한 연결 구문은 더 이상 사용되지 않습니다.

특정 연결 유형 구문을 사용하여 특정 데이터 형식에 액세스할 수 있습니다. 이 구문은 더 이상 사용되지 않습니다. 대신 the section called “데이터 포맷 옵션”에 제공된 s3 연결 유형과 형식 옵션을 사용하여 형식을 지정하는 것이 좋습니다.

"connectionType": "Orc"

Amazon S3에 저장된 파일에 Apache ORC(Hive Optimized Row Columnar) 파일 포맷으로 연결을 지정합니다.

`"connectionType": "orc"`에는 다음 연결 옵션을 사용합니다.

- `paths`: (필수 사항) 읽을 소스 Amazon S3 경로 목록입니다.
- (기타 옵션 이름/값 페어): 형식 지정 옵션을 포함한 모든 추가 옵션이 SparkSQL `DataSource`에 직접 전달됩니다.

"connectionType": "parquet"

Amazon S3에 Apache Parquet 파일 포맷으로 저장된 파일에 대한 연결을 지정합니다.

`"connectionType": "parquet"`에는 다음 연결 옵션을 사용합니다.

- `paths`: (필수 사항) 읽을 소스 Amazon S3 경로 목록입니다.
- (기타 옵션 이름/값 페어): 형식 지정 옵션을 포함한 모든 추가 옵션이 SparkSQL `DataSource`에 직접 전달됩니다.

Amazon S3 스토리지 클래스 제외

Amazon Simple Storage Service(Amazon S3)에서 파일 또는 파티션을 읽는 AWS Glue ETL 작업을 실행하는 경우 일부 Amazon S3 스토리지 클래스 유형을 제외할 수 있습니다.

Amazon S3에서 제공되는 스토리지 클래스는 다음과 같습니다.

- `STANDARD` - 자주 액세스하는 데이터의 범용 스토리지일 때
- `INTELLIGENT_TIERING` - 액세스 패턴이 불확실하거나 바뀌는 데이터일 때
- `STANDARD_IA` 및 `ONEZONE_IA` - 저장 기간이 길지만 액세스 횟수가 비교적 적은 데이터일 때
- `GLACIER`, `DEEP_ARCHIVE` 및 `REDUCED_REDUNDANCY` - 장기 아카이브 및 디지털 보존이 필요할 때

자세한 내용은 Amazon S3 Developer Guide의 Amazon S3 Storage Classes를 참조하세요.

이번 단원의 예제는 GLACIER 및 DEEP_ARCHIVE 스토리지 클래스를 제외하는 방법을 나타낸 것입니다. 두 클래스는 파일을 나열하는 데 사용되지만 복원하지 않을 경우 파일을 읽어오지는 못합니다. 자세한 내용은 Amazon S3 Developer Guide의 Restoring Archived Objects를 참조하세요.

스토리지 클래스 제외를 사용하면 해당 스토리지 클래스 계층에서 파티션이 포함된 테이블에 대해 AWS Glue 작업을 실행할 수 있습니다. 제외를 사용하지 않으면 해당 계층에서 데이터를 읽어오는 작업은 다음 오류와 함께 중단됩니다. AmazonS3Exception: The operation is not valid for the object's storage class.

AWS Glue에서 Amazon S3 스토리지 클래스를 필터링할 수 있는 여러 가지 방법이 있습니다.

주제

- 동적 프레임 생성 시 Amazon S3 스토리지 클래스 제외
- Data Catalog 테이블에서 Amazon S3 스토리지 클래스 제외

동적 프레임 생성 시 Amazon S3 스토리지 클래스 제외

동적 프레임을 생성할 때 Amazon S3 스토리지 클래스를 제외하려면 additionalOptions에서 excludeStorageClasses를 사용합니다. AWS Glue는 자동으로 고유한 Amazon S3 Lister 구현을 사용하여 지정된 스토리지 클래스에 해당하는 파일을 나열하고 제외합니다.

다음 Python 및 Scala 예제는 동적 프레임을 생성할 때 GLACIER 및 DEEP_ARCHIVE 스토리지 클래스를 제외하는 방법을 나타낸 것입니다.

Python 예제:

```
glueContext.create_dynamic_frame.from_catalog(
    database = "my_database",
    tableName = "my_table_name",
    redshift_tmp_dir = "",
    transformation_ctx = "my_transformation_context",
    additional_options = {
        "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)
```

Scala 예제:

```
val* *df = glueContext.getCatalogSource(
    nameSpace, tableName, "", "my_transformation_context",
    additionalOptions = JsonOptions(
        Map("excludeStorageClasses" -> List("GLACIER", "DEEP_ARCHIVE"))
    )
).getDynamicFrame()
```

Data Catalog 테이블에서 Amazon S3 스토리지 클래스 제외

AWS Glue ETL 작업에서 사용할 스토리지 클래스 제외를 AWS Glue Data Catalog의 테이블 파라미터로 지정할 수 있습니다. 이러한 파라미터는 AWS Command Line Interface(AWS CLI) 를 사용하거나 API에서 프로그래밍 방식으로 `CreateTable` 작업에 추가할 수 있습니다. 자세한 내용은 테이블 구조 단원과 CreateTable 단원을 참조하십시오.

그 밖에 AWS Glue 콘솔에서 제외된 스토리지 클래스를 지정하는 방법도 있습니다.

Amazon S3 스토리지 클래스를 제외하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 테이블을 선택합니다.
3. 목록에서 테이블 이름과 테이블 편집을 차례대로 선택합니다.
4. 테이블 속성에서 **`excludeStorageClasses`**를 키로, 그리고 **`[\"GLACIER\", \"DEEP_ARCHIVE\"]`**를 값으로 추가합니다.
5. Apply(적용)를 선택합니다.

AWS Glue에서 ETL 출력의 파티션 관리

파티셔닝은 데이터 세트를 조직하기 위한 중요한 기술로써 효율적으로 데이터 세트를 쿼리할 수 있습니다. 하나 이상의 열에 대한 구별되는 값을 기반으로 계층적 디렉터리 구조에서 데이터를 정리합니다.

예를 들어, 날짜를 기준으로 Amazon Simple Storage Service(Amazon S3)에서 애플리케이션 로그를 분할하고 연, 월, 일을 기준으로 쪼갤 수 있습니다. 하루만큼의 데이터와 대응하는 파일은 `s3://my_bucket/logs/year=2018/month=01/day=23/`와 같은 접두사 아래 위치합니다. 이제 AWS Glue는 Amazon Athena, Amazon Redshift Spectru과 같은 시스템과 더불어 Amazon S3의 기본 데이터를 모두 읽지 않고도 이러한 파티션을 사용하여 파티션 값을 기준으로 데이터를 필터링할 수 있습니다.

크롤러는 파일 유형 및 스키마를 추론할 뿐만 아니라 크롤러가 AWS Glue Data Catalog를 채울 경우 데이터 집합의 파티션 구조를 자동적으로 확인합니다. 결과로 얻은 파티션 열을 AWS Glue ETL 작업 또는 Amazon Athena와 같은 쿼리 엔진에서 쿼리할 수 있습니다.

테이블을 크롤링한 후 크롤러가 생성한 파티션을 볼 수 있습니다. AWS Glue 콘솔의 왼쪽 탐색 창에서 테이블(Tables)을 선택합니다. 크롤러에서 생성한 테이블을 선택한 다음 [파티션 보기(View Partitions)]를 선택합니다.

`key=val` 스타일에서 Apache Hive 스타일로 분할된 경로의 경우, 크롤러는 열 이름에 자동으로 키 이름을 채웁니다. 그렇지 않은 경우에는 `partition_0`, `partition_1` 같은 기본 이름을 사용합니다. 콘솔에서 기본 이름을 변경할 수 있습니다. 이를 위해 테이블로 이동합니다. 인덱스 탭 아래에 인덱스가 있는지 확인합니다. 인덱스가 있는 경우 계속 진행하려면 해당 인덱스를 삭제해야 합니다(이후 새 열 이름을 사용하여 다시 생성할 수 있음). 그런 다음 스키마 편집을 선택하고 여기에서 파티션 열의 이름을 수정합니다.

ETL 스크립트에서 파티션 열에 필터링합니다. 파티션 정보는 Data Catalog에 저장되므로 `from_catalog` API 호출을 사용하여 `DynamicFrame`에 파티션 열을 포함합니다. 예를 들어 `create_dynamic_frame.from_options` 대신 `create_dynamic_frame.from_catalog`을 사용하세요.

파티셔닝은 데이터 스캔을 줄이는 최적화 기법입니다. 이 기법이 적절한 시기를 식별하는 프로세스에 대한 자세한 내용은 AWS 권장 가이드의 Apache Spark용 AWS Glue 작업 성능 조정 모범 사례 가이드에 있는 데이터 스캔량 감소를 참조하세요.

푸시다운 조건자를 사용하여 예비 필터링

많은 경우, 조건자를 푸시다운하여 데이터 세트 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링할 수 있습니다. 데이터 집합 전체를 읽는 대신 DynamicFrame에서 필터링한 다음 Data Catalog에서 파티션 메타데이터에 직접 필터링을 적용할 수 있습니다. 그런 다음 실제로 필요한 것을 DynamicFrame에서 나열하고 읽습니다.

예를 들어, Python에서는 다음을 작성할 수 있습니다.

```
glue_context.create_dynamic_frame.from_catalog(
    database = "my_S3_data_set",
    table_name = "catalog_data_table",
    push_down_predicate = my_partition_predicate)
```

조건자 표현식을 만족하는 Data Catalog의 파티션을 로딩만 하는 DynamicFrame을 생성합니다. 로딩하는 데이터의 서브셋이 얼마나 작은지에 따라 진행 시간을 많이 줄일 수 있습니다.

조건자 표현식은 Spark SQL가 지원한 부울 확장일 수 있습니다. Spark SQL 쿼리의 WHERE에 넣는 어떤 것도 만족할 수 있습니다. 예를 들어 조건자 표현식 pushDownPredicate = "(year=='2017' and month=='04')"는 year가 2017이고 month가 04인 Data Catalog의 파티션만 로드합니다. 자세한 내용은 Apache Spark SQL 설명서, 특히 Scala SQL 함수 참조를 참조하십시오.

카탈로그 파티션 조건자를 사용한 서버 측 필터링

push_down_predicate 옵션은 카탈로그의 모든 파티션을 나열한 후 해당 파티션에 대한 Amazon S3의 파일을 나열하기 전에 적용됩니다. 테이블에 대한 파티션이 많은 경우 카탈로그 파티션 목록에 여전히 추가 시간 오버헤드가 발생할 수 있습니다. 이 오버헤드를 해결하기 위해 AWS Glue Data Catalog의 파티션 인덱스를 사용하는 catalogPartitionPredicate 옵션과 함께 서버 측 파티션 정리를 사용할 수 있습니다. 따라서 한 테이블에 수백만 개의 파티션이 있는 경우 파티션 필터링이 훨씬 빨라집니다. 카탈로그 파티션 인덱스에서 아직 지원되지 않는 조건자 구문이 catalogPartitionPredicate에 필요한 경우 additional_options에서 push_down_predicate와 catalogPartitionPredicate를 함께 사용할 수 있습니다.

Python:

```
dynamic_frame = glueContext.create_dynamic_frame.from_catalog(
    database=dbname,
    table_name=tablename,
    transformation_ctx="datasource0",
    push_down_predicate="day>=10 and customer_id like '10%'",
    additional_options={"catalogPartitionPredicate":"year='2021' and month='06'"}
)
```

Scala:

```
val dynamicFrame = glueContext.getCatalogSource(
    database = dbname,
    tableName = tablename,
    transformationContext = "datasource0",
    pushDownPredicate="day>=10 and customer_id like '10%'",
    additionalOptions = JsonOptions("""{
        "catalogPartitionPredicate": "year='2021' and month='06'"}""")
    ).getDynamicFrame()
```

> **Note**
>
> `push_down_predicate`와 `catalogPartitionPredicate`는 다른 구문을 사용합니다. 전자는 Spark SQL 표준 구문을 사용하고 후자는 JSQL 구문 분석기를 사용합니다.

파티션 작성

기본적으로 DynamicFrame은 작성될 때 파티션이 되지 않습니다. 출력 파일 모두는 지정된 출력 경로의 상위 수준에 작성됩니다. 최근까지 파티션에 DynamicFrame을 작성하는 유일한 방법은 작성 전에 DynamicFrame을 Spark SQL DataFrame으로 전환하는 방법이었습니다.

그러나 DynamicFrame은 싱크를 생성할 때 `partitionKeys` 옵션을 사용하여 키의 시퀀스를 사용한 본래 파티셔닝을 지원합니다. 예를 들어, 다음 Python 코드는 Parquet 포맷의 Amazon S3로 데이터 집합을 유형 필드로 분할된 디렉터리에 작성합니다. 그런 다음, Amazon Athena와 같은 다른 시스템을 사용하여 이런 파티션을 진행할 수 있습니다.

```
glue_context.write_dynamic_frame.from_options(
    frame = projectedEvents,
    connection_type = "s3",
    connection_options = {"path": "$outpath", "partitionKeys": ["type"]},
    format = "parquet")
```

입력 파일을 더 큰 그룹에서 읽기

테이블에 속성을 설정하여 AWS Glue ETL 작업을 활성화하여 속성을 Amazon S3 데이터 스토어에서 읽을 때 파일을 모읍니다. 이 속성은 각 ETL 태스크이 입력 파일 그룹을 단일 인메모리 파티션으로 읽을 수 있도록 만듭니다. 이렇게 하는 것은 Amazon S3 데이터 스토어에 많은 수의 작은 파일을 있을 때 유용합니다. 특정 속성을 설정하면 AWS Glue에 지시를 내려 Amazon S3 데이터 파티션 내 파일을 모으고 읽을 그룹의 크기를 설정합니다. `create_dynamic_frame.from_options` 메서드로 Amazon S3 데이터 스토어에서 읽을 때 이러한 옵션을 설정할 수도 있습니다.

테이블 파일을 모으려면 테이블 구조의 파라미터 필드에서 키 값 페어를 설정합니다. JSON 표기법을 사용하여 테이블 파라미터 필드 값을 설정합니다. 테이블 속성을 편집하는 방법에 대한 더 자세한 정보는 테이블 세부 정보 보기 및 편집를 참조하십시오.

이 방법을 사용하여 Amazon S3 데이터 스토어로 Data Catalog의 테이블을 모을 수 있도록 합니다.

groupFiles

groupFiles를 `inPartition`으로 설정하면 Amazon S3 데이터 파티션 내 파일 그룹화가 가능합니다. 입력 파일이 50,000개 이상일 경우 AWS Glue가 그룹화를 자동으로 활성화합니다.

```
'groupFiles': 'inPartition'
```

groupSize

groupSize를 바이트 단위로 그룹의 대상 크기로 설정합니다. groupSize 속성은 조건부입니다. 제공되지 않으면 총 ETL 작업 수 및 인메모리 파티션을 줄이면서 AWS Glue는 크기를 계산하여 클러스터의 모든 CPU 코어를 사용합니다.

예를 들어, 다음은 그룹 크기를 1MB로 설정합니다.

```
'groupSize': '1048576'
```

계산 결과로 `groupsize`를 설정해야 합니다. 예: 1024 * 1024 = 1048576.

recurse

recurse를 `True`로 설정하여 `paths`를 경로 어레이로 지정할 때 모든 하위 디렉터리의 파일을 재귀적으로 읽을 수 있습니다. 다음 예와 같이 `paths`(이가) Amazon S3의 객체 키 배열이거나 입력 형식이 parquet/orc인 경우 재귀를 설정하지 않아도 됩니다.

```
'recurse':True
```

`create_dynamic_frame.from_options` 메서드를 사용하여 Amazon S3에서 직접 읽는 경우 이러한 연결 옵션을 추가합니다. 예를 들어, 다음과 같이 파일을 1MB 그룹으로 그룹화하려는 시도합니다.

```
df = glueContext.create_dynamic_frame.from_options("s3", {'paths': ["s3://s3path/"],
 'recurse':True, 'groupFiles': 'inPartition', 'groupSize': '1048576'}, format="json")
```

> **Note**
>
> `groupFiles`는 csv, ion, grokLog, json, xml 등의 데이터 포맷에서 생성된 DynamicFrames에서 지원됩니다. avro, parquet, orc에서는 이 옵션이 지원되지 않습니다.

Amazon S3용 Amazon VPC 엔드포인트

보안상의 이유로 많은 AWS 고객은 Amazon Virtual Private Cloud 환경(Amazon VPC)에서 애플리케이션을 실행합니다. Amazon VPC를 통해 Amazon EC2 인스턴스를 퍼블릭 인터넷을 비롯한 다른 네트워크와 논리적으로 분리된 Virtual Private Cloud로 시작할 수 있습니다. Amazon VPC를 사용하면 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 및 보안 설정을 통제할 수 있습니다.

> **Note**
>
> 2013년 12월 4일 이후 AWS 계정을 생성했다면 각 AWS 리전에 기본 VPC가 갖춰져 있습니다. 별도의 구성 단계 없이 기본 VPC를 사용하여 즉시 시작할 수 있습니다.
> 자세한 내용은 Amazon VPC 사용 설명서의 기본 VPC 및 서브넷을 참조하세요.

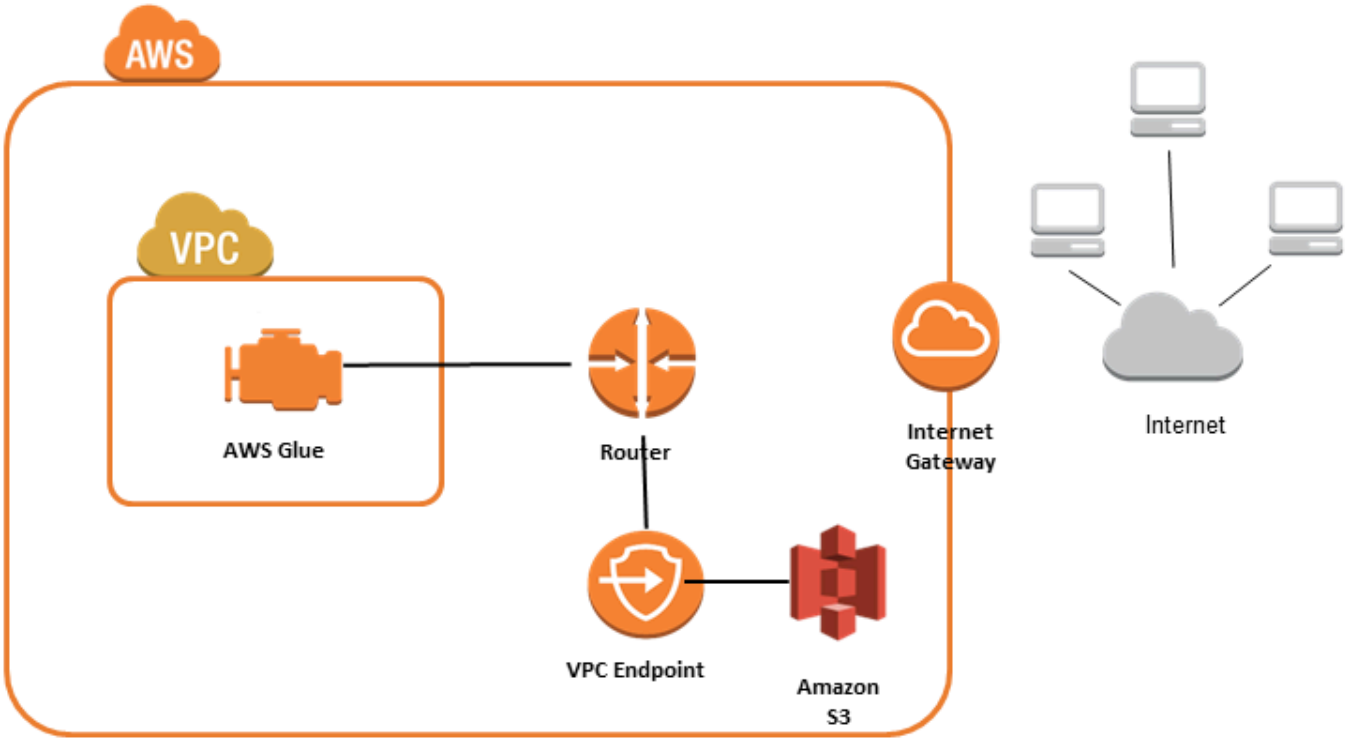
많은 고객이 퍼블릭 인터넷을 통해 데이터를 주고받는 것에 대한 합법적인 개인 정보 보호 및 보안 문제를 우려합니다. 가상 사설 네트워크(VPN)를 사용하여 모든 Amazon S3 네트워크 트래픽을 자체 기업 네트워크 인프라를 통해 라우팅함으로써 이러한 문제를 해결할 수 있습니다. 그러나 이러한 접근 방식은 대역폭 및 가용성 문제를 일으킬 수 있습니다.

Amazon S3에 대한 VPC 엔드포인트는 이러한 문제를 완화할 수 있습니다. Amazon S3에 대한 VPC 엔드포인트를 사용하여 AWS Glue가 퍼블릭 인터넷에 노출되지 않고도 프라이빗 IP 주소를 사용하여 Amazon S3에 액세스할 수 있습니다. AWS Glue에 퍼블릭 IP 주소를 지정할 필요가 없으며 VPC에서 인터넷 게이트웨이, NAT 디바이스 또는 가상 프라이빗 게이트웨이가 필요 없습니다. 엔드포인트 정책을 사용하여 Amazon S3에 대한 액세스를 제어합니다. VPC와 AWS 서비스 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

Amazon S3에 대한 VPC 엔드포인트를 생성하면 리전 내의 Amazon S3 엔드포인트(예: s3.us-west-2.amazonaws.com)에 대한 요청이 Amazon 네트워크 내의 프라이빗 Amazon S3 엔드포인트로 라우팅됩니다. VPC의 Amazon EC2 인스턴스에서 실행되는 애플리케이션을 수정할 필요가 없습니다. 엔드포인트 이름은 동일하게 남아있지만 Amazon S3에 대한 경로는 전적으로 Amazon 네트워크에 유지되며 퍼블릭 인터넷에 액세스하지 않습니다.

VPC 엔드포인트에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하세요.

다음 다이어그램은 AWS Glue가 Amazon S3에 액세스하기 위해 VPC 엔드포인트를 사용하는 방법을 보여줍니다.



Amazon S3에 대한 액세스를 설정하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 엔드포인트를 선택합니다.
3. [엔드포인트 생성(Create Endpoint)]을 선택하고 단계에 따라 게이트웨이 유형의 Amazon S3 VPC 엔드포인트를 생성합니다.

Amazon DocumentDB 연결

AWS Glue for Spark를 사용하여 Amazon DocumentDB 데이터베이스의 테이블에서 읽고 쓸 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 보안 인증 정보를 사용하여 Amazon DocumentDB에 연결할 수 있습니다.

Amazon DocumentDB에 대한 자세한 내용은 Amazon DocumentDB 설명서를 참조하세요.

> **Note**
>
> Amazon DocumentDB Elastic Clusters는 현재 AWS Glue 커넥터를 사용하는 경우 지원되지 않습니다. Elastic Clusters에 대한 자세한 내용은 Using Amazon DocumentDB elastic clusters를 참조하세요.

Amazon DocumentDB 컬렉션 읽기 및 쓰기

> **Note**
>
> Amazon DocumentDB에 연결하는 ETL 작업을 생성할 때 `Connections` 작업 속성에 대해 Amazon DocumentDB가 실행 중인 Virtual Private Cloud(VPC)를 지정하는 연결 객체를 지정해야 합니다. 연결 객체의 경우 연결 유형은 `JDBC`여야 하며 `JDBC URL`은 `mongo://<DocumentDB_host>:27017`이어야 합니다.

> **Note**
>
> 이 코드 샘플은 AWS Glue 3.0용으로 개발되었습니다. AWS Glue 4.0으로 마이그레이션하려면 the section called “MongoDB” 섹션을 참조하세요. `uri` 파라미터가 변경되었습니다.

> **Note**
>
> Amazon DocumentDB를 사용하는 경우 작성된 문서에서 `_id`를 지정하는 경우와 같은 특정 상황에서는 `retryWrites`를 false로 설정해야 합니다. 자세한 내용은 Amazon DocumentDB 설명서에서 Functional Differences with MongoDB를 참조하세요.

다음 Python 스크립트는 Amazon DocumentDB를 읽고 쓰기 위한 연결 유형 및 연결 옵션을 사용하는 방법을 보여줍니다.

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

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'], args)

output_path = "s3://some_bucket/output/" + str(time.time()) + "/"
documentdb_uri = "mongodb://<mongo-instanced-ip-address>:27017"
documentdb_write_uri = "mongodb://<mongo-instanced-ip-address>:27017"

read_docdb_options = {
    "uri": documentdb_uri,
    "database": "test",
    "collection": "coll",
    "username": "username",
    "password": "1234567890",
    "ssl": "true",
    "ssl.domain_match": "false",
    "partitioner": "MongoSamplePartitioner",
    "partitionerOptions.partitionSizeMB": "10",
    "partitionerOptions.partitionKey": "_id"
}

write_documentdb_options = {
    "retryWrites": "false",
    "uri": documentdb_write_uri,
    "database": "test",
    "collection": "coll",
```

```
    "username": "username",
    "password": "pwd"
}

# Get DynamicFrame from  DocumentDB
dynamic_frame2 =
 glueContext.create_dynamic_frame.from_options(connection_type="documentdb",

 connection_options=read_docdb_options)

# Write DynamicFrame to MongoDB and DocumentDB
glueContext.write_dynamic_frame.from_options(dynamic_frame2,
 connection_type="documentdb",

 connection_options=write_documentdb_options)

job.commit()
```

다음 Scala 스크립트는 Amazon DocumentDB를 읽고 쓰기 위한 연결 유형 및 연결 옵션을 사용하는 방법을 보여줍니다.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  val DOC_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  val DOC_WRITE_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  lazy val documentDBJsonOption = jsonOptions(DOC_URI)
  lazy val writeDocumentDBJsonOption = jsonOptions(DOC_WRITE_URI)
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Get DynamicFrame from DocumentDB
```

```
    val resultFrame2: DynamicFrame = glueContext.getSource("documentdb",
 documentDBJsonOption).getDynamicFrame()

    // Write DynamicFrame to DocumentDB
    glueContext.getSink("documentdb", writeJsonOption).writeDynamicFrame(resultFrame2)

    Job.commit()
  }

  private def jsonOptions(uri: String): JsonOptions = {
    new JsonOptions(
      s"""{"uri": "${uri}",
         |"database":"test",
         |"collection":"coll",
         |"username": "username",
         |"password": "pwd",
         |"ssl":"true",
         |"ssl.domain_match":"false",
         |"partitioner": "MongoSamplePartitioner",
         |"partitionerOptions.partitionSizeMB": "10",
         |"partitionerOptions.partitionKey": "_id"}""".stripMargin)
  }
}
```

Amazon DocumentDB 연결 옵션 참조

Amazon DocumentDB에 대한 연결을 지정합니다(MongoDB와 호환).

연결 옵션은 소스 연결 및 싱크 연결에 따라 다릅니다.

소스로서의 "connectionType": "Documentdb"

소스로서의 `"connectionType": "documentdb"`에는 다음 연결 옵션을 사용합니다.

- `"uri"`: (필수 사항) 읽을 소스 Amazon DocumentDB 호스트로, `mongodb://<host>:<port>` 포맷입니다.
- `"database"`: (필수) 읽을 소스 Amazon DocumentDB 데이터베이스입니다.
- `"collection"`: (필수) 읽을 소스 Amazon DocumentDB 컬렉션입니다.
- `"username"`: (필수) Amazon DocumentDB 사용자 이름입니다.
- `"password"`: (필수) Amazon DocumentDB 암호입니다.

- "ssl": (SSL을 사용하는 경우 필수) 연결에서 SSL을 사용하는 경우 "true" 값과 함께 이 옵션을 포함해야 합니다.
- "ssl.domain_match": (SSL을 사용하는 경우 필수) 연결에서 SSL을 사용하는 경우 "false" 값과 함께 이 옵션을 포함해야 합니다.
- "batchSize": (선택 사항) 내부 배치의 커서 내에서 사용되는 배치당 반환할 문서 수입니다.
- "partitioner": (선택 사항) Amazon DocumentDB에서 입력 데이터를 읽는 파티셔너의 클래스 이름입니다. 커넥터는 다음 파티셔너를 제공합니다.
  - MongoDefaultPartitioner(기본값)(AWS Glue 4.0에서는 지원되지 않음)
  - MongoSamplePartitioner(AWS Glue 4.0에서는 지원되지 않음)
  - MongoShardedPartitioner
  - MongoSplitVectorPartitioner
  - MongoPaginateByCountPartitioner
  - MongoPaginateBySizePartitioner(AWS Glue 4.0에서는 지원되지 않음)
- "partitionerOptions": (선택 사항) 지정된 파티셔너에 대한 옵션입니다. 각 파티셔너에 대해 다음 옵션이 지원됩니다.
  - MongoSamplePartitioner: partitionKey, partitionSizeMB, samplesPerPartition
  - MongoShardedPartitioner: shardkey
  - MongoSplitVectorPartitioner: partitionKey, partitionSizeMB
  - MongoPaginateByCountPartitioner: partitionKey, numberOfPartitions
  - MongoPaginateBySizePartitioner: partitionKey, partitionSizeMB

  이러한 옵션에 대한 자세한 내용은 MongoDB 설명서의 파티셔너 구성을 참조하십시오.

싱크로서의 "connectionType": "Documentdb"

싱크로서의 "connectionType": "documentdb"에는 다음 연결 옵션을 사용합니다.

- "uri": (필수 사항) 쓸 대상 Amazon DocumentDB 호스트로, mongodb://<host>:<port> 포맷입니다.
- "database": (필수) 쓸 대상 Amazon DocumentDB 데이터베이스입니다.
- "collection": (필수) 쓸 대상 소스 Amazon DocumentDB 컬렉션입니다.
- "username": (필수) Amazon DocumentDB 사용자 이름입니다.
- "password": (필수) Amazon DocumentDB 암호입니다.

- `"extendedBsonTypes"`: (선택 사항) `true`이면 데이터를 Amazon DocumentDB에 쓸 때 확장 BSON 유형을 허용합니다. 기본값은 `true`입니다.
- `"replaceDocument"`: (선택 사항) `true`이면 `_id` 필드가 포함된 데이터 세트를 저장할 때 전체 문서를 대체합니다. `false`이면 데이터 세트의 필드와 일치하는 문서의 필드만 업데이트됩니다. 기본값은 `true`입니다.
- `"maxBatchSize"`: (선택 사항) 데이터를 저장할 때 대량 작업의 최대 배치 크기입니다. 기본값은 512입니다.
- `"retryWrites"`: (선택 사항): AWS Glue에서 네트워크 오류가 발생하는 경우 특정 쓰기 작업을 한 번 자동으로 재시도합니다.

OpenSearch Service 연결

AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 OpenSearch Service의 테이블에서 읽고 쓸 수 있습니다. OpenSearch 쿼리를 사용하여 OpenSearch Service에서 읽을 내용을 정의할 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 HTTP 기본 인증 보안 인증 정보를 사용하여 OpenSearch Service에 연결합니다. 이 기능은 OpenSearch Service 서버리스와 호환되지 않습니다.

Amazon OpenSearch Service에 대한 자세한 정보는 Amazon OpenSearch Service 설명서를 참조하십시오.

OpenSearch Service 연결 구성

AWS Glue에서 OpenSearch Service에 연결하려면 OpenSearch Service 보안 인증 정보를 생성하여 암호에 저장한 다음 해당 AWS Secrets Manager 암호를 OpenSearch Service AWS Glue 연결에 연결해야 합니다.

사전 조건:

- Amazon OpenSearch Service 설명서의 지침에 따라 읽으려는 도메인 엔드포인트, *`aosEndpoint`* 및 포트, *`aosPort`*를 식별하거나 리소스를 생성하십시오. 도메인 생성에 관해 자세한 내용을 알아보려면 Amazon OpenSearch Service 설명서의 Amazon OpenSearch Service 도메인 생성 및 관리를 참조하십시오.

  Amazon OpenSearch Service 도메인 엔드포인트는 기본 형식이 https://search-*`domainName`*-*`unstructuredIdContent`*.*`region`*.es.amazonaws.com입니다. 도메인 엔드포인트에 관해 자세한 내용을 알아보려면 Amazon OpenSearch Service 설명서의 Amazon OpenSearch Service 도메인 생성 및 관리를 참조하십시오.

도메인의 HTTP 기본 보안 인증 정보, *aosUser*와 *aosPassword*를 확인하거나 생성하십시오.

OpenSearch Service에 대한 연결을 구성하는 방법:

1. AWS Secrets Manager에서 OpenSearch 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
   - 키/값 페어를 선택하면 값 *aosUser*이 포함된 키 USERNAME에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *aosPassword*이 포함된 키 PASSWORD에 대한 페어를 생성합니다.
2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때는 OpenSearch Service를 선택합니다.
   - 도메인 엔드포인트를 선택할 때는 *aosEndpoint*를 제공하십시오.
   - 포트를 선택할 때는 *aosPort*를 제공하십시오.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

AWS Glue OpenSearch Service 연결을 생성한 후에는 AWS Glue 작업을 실행하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

OpenSearch Service 인덱스에서 읽기

사전 조건:

- 읽으려는 OpenSearch Service 인덱스, *aosIndex*.
- 인증 및 네트워크 위치 정보를 제공하도록 구성된 AWS Glue OpenSearch Service 연결입니다. 이 정보를 얻으려면 앞 절차인 OpenSearch Service에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

이 예제는 Amazon OpenSearch Service에서 인덱스을 읽습니다. pushdown 파라미터를 제공해야 합니다.

예시:

```
opensearch_read = glueContext.create_dynamic_frame.from_options(
    connection_type="opensearch",
    connection_options={
        "connectionName": "connectionName",
        "opensearch.resource": "aosIndex",
        "pushdown": "true",
    }
)
```

DynamicFrame에 반환되는 결과를 필터링하는 쿼리 문자열을 제공할 수도 있습니다. opensearch.query를 구성해야 합니다.

opensearch.query는 URL 쿼리 매개 변수 문자열 *queryString* 또는 쿼리 DSL JSON 객체 *queryObject*를 사용할 수 있습니다. 쿼리 DSL에 대한 자세한 내용은 OpenSearch 설명서의 쿼리 DSL을 참조하십시오. URL 쿼리 매개 변수 문자열을 제공하려면 정규화된 URL에서처럼 쿼리 앞에 ?q=를 추가하십시오. 쿼리 DSL 객체를 제공하려면 JSON 객체를 제공하기 전에 문자열을 이스케이프 처리해야 합니다.

예시:

```
            queryObject = "{ "query": { "multi_match": { "query": "Sample", "fields":
 [ "sample" ] } } }"
            queryString = "?q=queryString"

            opensearch_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="opensearch",
    connection_options={
        "connectionName": "connectionName",
        "opensearch.resource": "aosIndex",
        "opensearch.query": queryString,
        "pushdown": "true",
    }
)
```

특정 구문을 벗어나 쿼리를 작성하는 방법에 대한 자세한 내용은 OpenSearch 설명서의 쿼리 문자열 구문을 참조하십시오.

배열 형식 데이터가 포함된 OpenSearch 컬렉션에서 읽을 때는 `opensearch.read.field.as.array.include` 파라미터를 사용하여 메서드 직접 호출에서 어떤 필드가 배열 형식인지 지정해야 합니다.

예를 들어 다음 문서를 읽을 때 `genre` 및 `actor` 배열 필드가 나타납니다.

```
{
    "_index": "movies",
    "_id": "2",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 1,
    "found": true,
    "_source": {
        "director": "Frankenheimer, John",
        "genre": [
            "Drama",
            "Mystery",
            "Thriller",
            "Crime"
        ],
        "year": 1962,
        "actor": [
            "Lansbury, Angela",
            "Sinatra, Frank",
            "Leigh, Janet",
            "Harvey, Laurence",
            "Silva, Henry",
            "Frees, Paul",
            "Gregory, James",
            "Bissell, Whit",
            "McGiver, John",
            "Parrish, Leslie",
            "Edwards, James",
            "Flowers, Bess",
            "Dhiegh, Khigh",
            "Payne, Julie",
            "Kleeb, Helen",
            "Gray, Joe",
            "Nalder, Reggie",
            "Stevens, Bert",
            "Masters, Michael",
            "Lowell, Tom"
```

```
        ],
        "title": "The Manchurian Candidate"
    }
}
```

이 경우 메서드 직접 호출에 해당 필드 이름을 포함하면 됩니다. 예시:

```
"opensearch.read.field.as.array.include": "genre,actor"
```

배열 필드가 문서 구조 안에 중첩되어 있는 경우 점 표기법("genre,actor,foo.bar.baz")을 사용하여 참조하세요. 이는 포함된 문서(bar)를 포함하는 포함된 문서(foo)를 통해 소스 문서에 포함된 배열(baz)을 지정합니다.

OpenSearch Service 테이블에 쓰기

이 예제에서는 기존 DynamicFrame, *dynamicFrame*의 정보를 OpenSearch Service에 기록합니다. 인덱스에 이미 정보가 있는 경우 AWS Glue는 DynamicFrame의 데이터를 추가합니다. pushdown 파라미터를 제공해야 합니다.

사전 조건:

- 쓰려는 OpenSearch Service 테이블. 테이블에 대한 식별 정보가 필요합니다. 이 *tableName*이라고 부르겠습니다.
- 인증 및 네트워크 위치 정보를 제공하도록 구성된 AWS Glue OpenSearch Service 연결입니다. 이 정보를 얻으려면 앞 절차인 OpenSearch Service에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예시:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="opensearch",
    connection_options={
      "connectionName": "connectionName",
      "opensearch.resource": "aosIndex",
    },
)
```

OpenSearch Service 연결 옵션 참조

- `connectionName` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법에 인증 및 네트워크 위치 정보를 제공하도록 구성된 AWS Glue OpenSearch Service 연결의 이름입니다.
- `opensearch.resource` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 유효한 값: OpenSearch 인덱스 이름. 연결 방법이 상호 작용하는 인덱스의 이름입니다.
- `opensearch.query` — 읽기에 사용됩니다. 유효한 값: 문자열이 JSON으로 이스케이프되거나 이 문자열이 ?로 시작되는 경우 URL의 검색 부분. 읽을 때 검색해야 하는 항목을 필터링하는 OpenSearch 쿼리입니다. 이 매개변수 사용에 대한 자세한 내용은 이전 섹션 the section called "OpenSearch Service에서 읽기"를 참조하십시오.
- `pushdown` - 다음과 같은 경우 필수입니다. 읽기에 사용됩니다. 유효 값: 부울. 데이터베이스가 관련 문서만 반환하도록 Spark가 읽기 쿼리를 OpenSearch에 전달하도록 지시합니다.
- `opensearch.read.field.as.array.include` - 배열 유형 데이터를 읽을 때 필요합니다. 읽기에 사용됩니다. 유효 값: 쉼표로 구분된 필드 이름 목록. OpenSearch 문서에서 배열로 읽을 필드를 지정합니다. 이 매개변수 사용에 대한 자세한 내용은 이전 섹션 the section called "OpenSearch Service에서 읽기"를 참조하십시오.

Redshift 연결

AWS Glue for Spark를 사용하여 Amazon Redshift 데이터베이스의 테이블에서 읽고 쓸 수 있습니다. Amazon Redshift 데이터베이스에 연결할 때 AWS Glue는 Amazon Redshift SQL `COPY` 및 `UNLOAD` 명령을 사용하여 Amazon S3를 통해 데이터를 이동함으로써 처리량을 극대화합니다. AWS Glue 4.0 이상에서는 Apache Spark용 Amazon Redshift 통합을 사용하여 이전 버전을 통해 연결할 때 사용할 수 있었던 기능 외에도 Amazon Redshift에서만 지원하는 기능 및 최적화를 사용하여 읽고 쓸 수 있습니다.

Amazon Redshift 사용자가 AWS Glue를 사용하여 서버리스 데이터 통합 및 ETL을 위해 보다 쉽게 AWS로 마이그레이션하는 방법에 대해 알아보세요.

Redshift 연결 구성

AWS Glue에서 Amazon Redshift 클러스터를 사용하려면 다음과 같은 몇 가지 필수 조건이 필요합니다.

- 데이터베이스에서 읽고 쓸 때 임시 스토리지로 사용하는 Amazon S3 디렉터리.
- Amazon Redshift 클러스터, AWS Glue 작업 및 Amazon S3 디렉터리 간 통신을 지원하는 Amazon VPC.

- AWS Glue 작업 및 Amazon Redshift 클러스터에 대한 적절한 IAM 권한.

IAM 역할 구성

Amazon Redshift 클러스터의 역할 설정

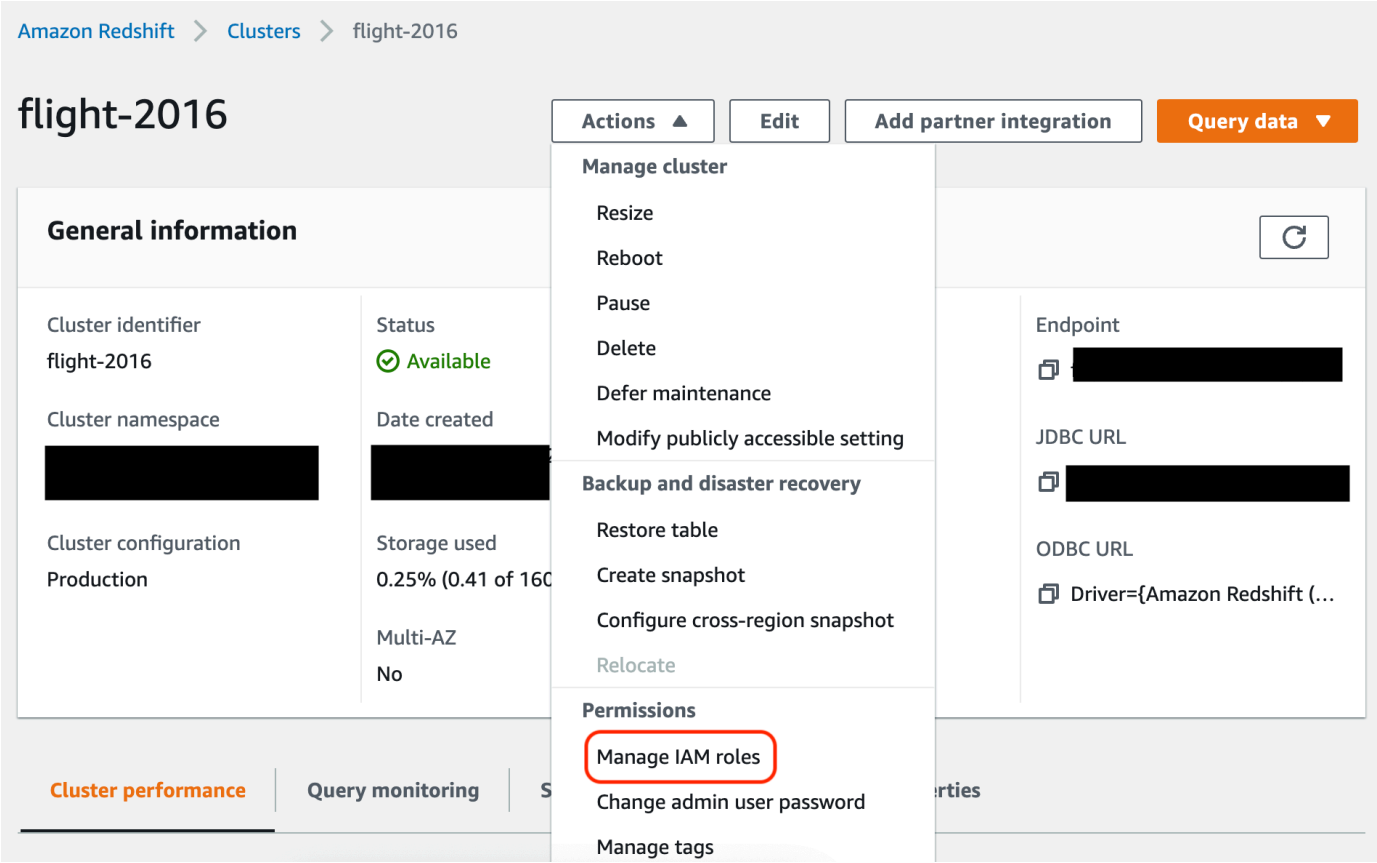
AWS Glue 작업과 통합하려면 Amazon Redshift 클러스터는 Amazon S3에서 읽고 쓸 수 있어야 합니다. 이를 허용하기 위해 연결하려는 Amazon Redshift 클러스터에 IAM 역할을 연결할 수 있습니다. 역할에는 Amazon S3 임시 디렉터리에서의 읽기 및 쓰기를 허용하는 정책이 있어야 합니다. 역할에는 `AssumeRole`을 위해 `redshift.amazonaws.com` 서비스를 허용하는 신뢰 관계가 있어야 합니다.

IAM 역할을 Amazon Redshift에 연결하려면

1. 필수 조건: 파일의 임시 저장에 사용되는 Amazon S3 버킷 또는 디렉터리.
2. Amazon Redshift 클러스터에 필요한 Amazon S3 권한을 식별합니다. Amazon Redshift 클러스터 간에 데이터를 이동하는 경우 AWS Glue 작업이 Amazon Redshift에 대해 COPY 및 UNLOAD 문을 실행합니다. 작업에서 Amazon Redshift의 테이블을 수정하는 경우 AWS Glue는 CREATE LIBRARY 문도 실행합니다. Amazon Redshift에서 이러한 명령문을 실행하는 데 필요한 특정 Amazon S3 권한에 대한 자세한 내용은 Amazon Redshift 설명서, Amazon Redshift: 다른 AWS 리소스에 대한 액세스 권한을 참조하세요.
3. IAM 콘솔에서 필요한 권한이 포함된 IAM 정책을 생성합니다. IAM 정책 생성에 대한 자세한 내용은 IAM 정책 생성을 참조하세요.
4. IAM 콘솔에서 Amazon Redshift가 역할을 수임할 수 있는 역할 및 신뢰 관계를 생성합니다. IAM 설명서의 지침, AWS 서비스에 대한 역할 생성(콘솔)을 수행합니다.
   - AWS 서비스 사용 사례를 선택하라는 메시지가 표시되면 'Redshift - 사용자 지정 가능'을 선택합니다.
   - 정책을 연결하라는 메시지가 표시되면 이전에 정의한 정책을 선택합니다.

   > **Note**
   >
   > Amazon Redshift의 역할을 구성하는 방법에 대한 자세한 내용은 Amazon Redshift 설명서에서 Amazon Redshift가 사용자를 대신하여 다른 AWS 서비스에 액세스할 수 있도록 권한 부여를 참조하세요.
5. Amazon Redshift 콘솔에서 Amazon Redshift 클러스터에 역할을 연결합니다. Amazon Redshift 설명서의 지침을 수행합니다.

Amazon Redshift 콘솔에서 강조 표시된 옵션을 선택하여 다음 설정을 구성합니다.



> **Note**
>
> 기본적으로 AWS Glue는 작업을 실행하기 위해 사용자가 지정한 역할을 사용하여 생성되는 Amazon Redshift의 임시 보안 인증을 전달합니다. 이 보안 인증은 사용하지 않는 것이 좋습니다. 보안을 위해 이러한 보안 인증은 1시간 후에 만료됩니다.

AWS Glue 작업의 역할 설정

AWS Glue 작업에는 Amazon S3 버킷에 액세스하기 위한 역할이 필요합니다. Amazon Redshift 클러스터에는 IAM 권한이 필요하지 않습니다. 액세스는 데이터베이스 보안 인증 및 Amazon VPC의 연결을 통해 제어됩니다.

Amazon VPC 설정

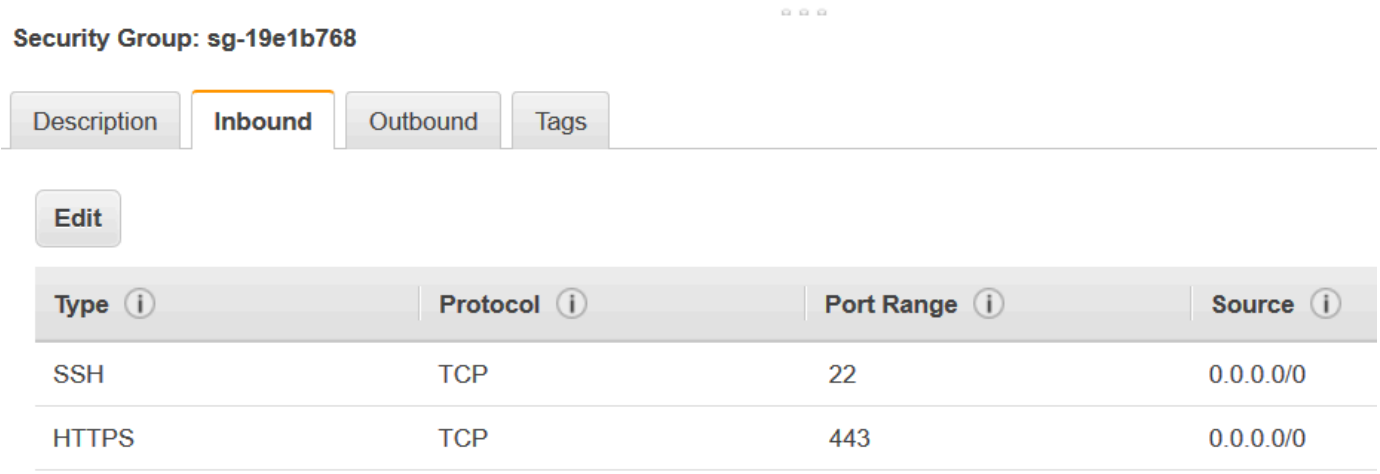
Amazon Redshift 데이터 스토어에 대한 액세스를 설정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/redshiftv2/에서 Amazon Redshift 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. AWS Glue로부터 액세스하고자 하는 클러스터 이름을 선택합니다.
4. Cluster Properties(클러스터 속성) 섹션에서 VPC security groups(VPC 보안 그룹)에서 보안 그룹을 선택하여 사용할 AWS Glue를 허용합니다. 미래 참조를 위해 선택하는 보안 그룹의 이름을 기록합니다. 보안 그룹을 선택하면 Amazon EC2 콘솔 [보안 그룹(Security Groups)] 목록을 엽니다.
5. 보안 그룹을 선택하여 [Inbound(인바운드)] 탭으로 수정하고 탐색합니다.
6. 자기 참조 규칙을 추가하여 AWS Glue 구성 요소를 허용하여 통신합니다. 특히, [Type(유형)] `All TCP` 의 규칙을 추가하고 확인합니다. [Protocol(프로토콜)]는 `TCP`이고 [Port Range(포트 범위)]는 모든 포트를 포함하고 포트의 [Source(원본)]은 [Group ID(그룹 ID)]이라는 동일한 보안 그룹입니다.

   인바운드 규칙은 다음과 비슷하게 보입니다.

유형	프로토콜	포트 범위	소스
모든 TCP	TCP	0~65535	database-security-group

예:



7. 아웃바운드 트래픽 규칙도 추가합니다. 예를 들어 아웃바운드 트래픽을 모든 포트로 엽니다.

유형	프로토콜	포트 범위	대상
모든 트래픽	ALL	ALL	0.0.0.0/0

또는 자기 참조 규칙을 생성합니다. 여기서 Type(유형) 은 `All TCP`이고 Protocol(프로토콜)은 `TCP`이고 Port Range(포트 범위)는 모든 포트를 포함하고 포트의 Destination(대상)은 Group ID(그룹 ID)와 동일한 보안 그룹 이름입니다. Amazon S3 VPC 엔드포인트를 사용할 경우 Amazon S3 액세스에 대한 HTTPS 규칙도 추가합니다. *`s3-prefix-list-id`*는 VPC에서 Amazon S3 VPC 엔드포인트로 트래픽을 허용하기 위해 보안 그룹에 필요합니다.

예:

유형	프로토콜	포트 범위	대상
모든 TCP	TCP	0~65535	*`security-group`*
HTTPS	TCP	443	*`s3-prefix-list-id`*

AWS Glue 설정

Amazon VPC 연결 정보를 제공하는 AWS Glue 데이터 카탈로그 연결을 생성해야 합니다.

콘솔에서 AWS Glue에 대한 Amazon Redshift Amazon VPC 연결을 구성하려면

1. the section called “AWS Glue 연결 추가”의 단계에 따라 데이터 카탈로그 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *`connectionName`*을 유지합니다.
   - 연결 유형을 선택할 때는 Amazon Redshift를 선택합니다.
   - Redshift 클러스터를 선택할 때는 이름을 기준으로 클러스터를 선택합니다.
   - 클러스터의 Amazon Redshift 사용자에 대한 기본 연결 정보를 제공합니다.
   - Amazon VPC 설정이 자동으로 구성됩니다.

> **Note**
>
> AWS SDK를 통해 Amazon Redshift 연결을 생성할 때는 Amazon VPC에 대한 `PhysicalConnectionRequirements`를 수동으로 제공해야 합니다.

2. AWS Glue 작업 구성에서 추가 네트워크 연결로 *`connectionName`*을 제공합니다.

예제: Amazon Redshift 테이블에서 읽기

Amazon Redshift 클러스터와 Amazon Redshift 서버리스 환경에서 읽을 수 있습니다.

필수 조건: 읽으려는 Amazon Redshift 테이블. 임시 디렉터리로 Amazon S3 URI(*`temp-s3-dir`*) 및 IAM 역할(*`role-account-id`* 계정에 있는 *`rs-role-name`*)을 보유한 후에 이전 the section called "Redshift 구성" 섹션의 단계를 수행합니다.

Using the Data Catalog

추가 필수 조건: 읽으려는 Amazon Redshift 테이블의 데이터 카탈로그 데이터베이스 및 테이블. 데이터 카탈로그에 대한 자세한 내용은 데이터 검색 및 카탈로그 작성 섹션을 참조하세요. Amazon Redshift 테이블에 대한 항목을 생성한 후 *`redshift-dc-database-name`* 및 *`redshift-table-name`*으로 연결을 식별합니다.

구성: 함수 옵션에서 `database` 및 `table_name` 파라미터로 데이터 카탈로그 테이블을 식별합니다. Amazon S3 임시 디렉터리를 `redshift_tmp_dir`로 식별합니다. 또한 `additional_options` 파라미터의 `aws_iam_role` 키를 사용하여 *`rs-role-name`*도 제공합니다.

```
glueContext.create_dynamic_frame.from_catalog(
   database = "redshift-dc-database-name",
   table_name = "redshift-table-name",
   redshift_tmp_dir = args["temp-s3-dir"],
   additional_options = {"aws_iam_role": "arn:aws:iam::role-account-id:role/rs-role-name"})
```

Connecting directly

추가 필수 조건: *Amazon Redshift ### ##(redshift-table-name*)이 필요합니다. 해당 테이블을 저장하는 Amazon Redshift 클러스터에 대한 JDBC 연결 정보가 필요합니다. *host*, *port*, *redshift-database-name*, *username*, *password*에서 연결 정보를 제공합니다.

Amazon Redshift 클러스터를 사용할 때 Amazon Redshift 콘솔에서 연결 정보를 검색할 수 있습니다. Amazon Redshift Serverless를 사용하는 경우 Amazon Redshift 설명서에서 Amazon Redshift Serverless에 연결을 참조하세요.

구성: 함수 옵션에서 `url`, `dbtable`, `user` 및 `password`로 연결 파라미터를 식별합니다. Amazon S3 임시 디렉터리를 `redshift_tmp_dir`로 식별합니다. `from_options`를 사용할 때 `aws_iam_role`을 사용하여 IAM 역할을 지정할 수 있습니다. 구문은 데이터 카탈로그를 통한 연결과 비슷하지만 `connection_options` 맵에 파라미터를 입력합니다.

암호를 AWS Glue 스크립트로 하드코딩하는 것은 좋지 않습니다. SDK for Python(Boto3)을 사용하여 AWS Secrets Manager에 암호를 저장하고 스크립트에서 검색하는 방법을 고려합니다.

```
my_conn_options = {
    "url": "jdbc:redshift://host:port/redshift-database-name",
    "dbtable": "redshift-table-name",
    "user": "username",
    "password": "password",
    "redshiftTmpDir": args["temp-s3-dir"],
    "aws_iam_role": "arn:aws:iam::account id:role/rs-role-name"
}

df = glueContext.create_dynamic_frame.from_options("redshift", my_conn_options)
```

예제: Amazon Redshift 테이블에 쓰기

Amazon Redshift 클러스터와 Amazon Redshift 서버리스 환경에 쓸 수 있습니다.

필수 조건: Amazon Redshift 클러스터. 그리고 임시 디렉터리로 Amazon S3 URI(*temp-s3-dir*) 및 IAM 역할(*role-account-id* 계정에 있는 *rs-role-name*)을 보유한 후에 이전 the section called “Redshift 구성” 섹션의 단계를 수행합니다. 또한 데이터베이스에 해당 콘텐츠를 쓰려는 `DynamicFrame`도 필요합니다.

Using the Data Catalog

추가 필수 조건: 쓰려는 Amazon Redshift 클러스터 및 테이블의 데이터 카탈로그 데이터베이스. 데이터 카탈로그에 대한 자세한 내용은 데이터 검색 및 카탈로그 작성 섹션을 참조하세요. *redshift-dc-database-name*으로 연결을, *redshift-table-name*으로 대상 테이블을 식별합니다.

구성: 함수 옵션에서 database 파라미터로 데이터 카탈로그 데이터베이스를 식별하고 table_name에서 테이블을 제공합니다. Amazon S3 임시 디렉터리를 redshift_tmp_dir로 식별합니다. 또한 additional_options 파라미터의 aws_iam_role 키를 사용하여 *rs-role-name*도 제공합니다.

```
glueContext.write_dynamic_frame.from_catalog(
    frame = input dynamic frame,
    database = "redshift-dc-database-name",
    table_name = "redshift-table-name",
    redshift_tmp_dir = args["temp-s3-dir"],
    additional_options = {"aws_iam_role": "arn:aws:iam::account-id:role/rs-role-
name"})
```

Connecting through a AWS Glue connection

write_dynamic_frame.from_options 메서드를 사용하여 Amazon Redshift에 직접 연결할 수 있습니다. 하지만 스크립트에 연결 세부 정보를 직접 삽입하는 대신 from_jdbc_conf 메서드를 사용하여 데이터 카탈로그 연결에 저장된 연결 세부 정보를 참조할 수 있습니다. 데이터베이스의 데이터 카탈로그 테이블을 크롤링하거나 생성하지 않고도 이 작업을 수행할 수 있습니다. 데이터 카탈로그 연결에 대한 자세한 내용은 데이터에 연결 섹션을 참조하세요.

추가 필수 조건: 읽으려는 Amazon Redshift 테이블, 사용자 데이터베이스에 대한 데이터 카탈로그 연결.

구성: *dc-connection-name*으로 데이터 카탈로그 연결을 식별합니다. *redshift-table-name* 및 *redshift-database-name*으로 Amazon Redshift 데이터베이스 및 테이블을 식별합니다. catalog_connection으로 데이터 카탈로그 연결 정보를 제공하고 dbtable 및 database로 Amazon Redshift 정보를 제공합니다. 구문은 데이터 카탈로그를 통한 연결과 비슷하지만 connection_options 맵에 파라미터를 입력합니다.

```
my_conn_options = {
```

```
    "dbtable": "redshift-table-name",
    "database": "redshift-database-name",
    "aws_iam_role": "arn:aws:iam::role-account-id:role/rs-role-name"
}

glueContext.write_dynamic_frame.from_jdbc_conf(
    frame = input dynamic frame,
    catalog_connection = "dc-connection-name",
    connection_options = my_conn_options,
    redshift_tmp_dir = args["temp-s3-dir"])
```

Amazon Redshift 연결 옵션 참조

모든 AWS Glue JDBC 연결에서 `url`, `user` 및 `password`와 같은 정보를 설정하는 데 사용되는 기본 연결 옵션은 모든 JDBC 유형에서 일관됩니다. 표준 JDBC 클러스터 파라미터에 대한 자세한 내용은 the section called “JDBC 연결 파라미터” 섹션을 참조하세요.

Amazon Redshift 연결 유형에는 다음과 같은 몇 가지 추가 연결 옵션이 있습니다.

- `"redshiftTmpDir"`: (필수) 데이터베이스 외부에서 복사할 때 임시 데이터를 스테이징할 수 있는 Amazon S3 경로.
- `"aws_iam_role"`: (선택 사항) IAM 역할의 ARN. AWS Glue 작업에서는 이 역할을 Amazon Redshift 클러스터로 전달하여 작업의 지침을 완료하는 데 필요한 권한을 클러스터에 부여합니다.

AWS Glue 4.0 이상에서 사용할 수 있는 추가 연결 옵션

AWS Glue 연결 옵션을 통해 새로운 Amazon Redshift 커넥터의 옵션을 전달할 수도 있습니다. 지원되는 커넥터 옵션의 전체 목록은 Apache Spark용 Amazon Redshift 통합의 Spark SQL 파라미터 섹션을 참조하세요.

편의를 위해 다음과 같은 새로운 특정 옵션을 다시 설명합니다.

명칭	필수	기본값	설명
autopushdown	아니요	TRUE	SQL 작업에 대한 Spark 논리 계획을 캡처하고 분석하여 조건

명칭	필수	기본값	설명
			자 및 쿼리 푸시다운을 적용합니다. 작업은 SQL 쿼리로 변환된 다음 Amazon Redshift에서 실행되어 성능이 향상됩니다.
autopushdown.s3_result_cache	아니요	FALSE	동일한 쿼리가 동일한 Spark 세션에서 다시 실행될 필요가 없도록 SQL 쿼리를 캐시하여 Amazon S3 경로 매핑 데이터를 메모리에서 언로드합니다. `autopushdown` 이 활성화된 경우에만 지원됩니다.
unload_s3_format	아니요	PARQUET	PARQUET - 쿼리 결과를 Parquet 형식으로 언로드합니다. TEXT - 쿼리 결과를 파이프로 구분된 텍스트 형식으로 언로드합니다.
sse_kms_key	아니요	N/A	AWS의 기본 암호화 대신 `UNLOAD` 작업 중에 암호화에 사용할 AWS SSE-KMS 키입니다.

명칭	필수	기본값	설명
extracopyoptions	아니요	N/A	데이터를 로드할 때 Amazon Redshift COPY 명령에 추가할 추가 옵션 목록(예: TRUNCATECOLUMNS 또는 MAXERROR n) 입니다. 다른 옵션은 COPY: 선택적 파라미터를 참조하세요. 이러한 옵션은 COPY 명령 끝에 추가되므로 명령 끝에 의미가 있는 옵션만 사용할 수 있습니다. 여기서는 가능한 대부분의 사용 사례를 다루어야 합니다.
csvnullstring(실험용)	아니요	NULL	CSV tempformat 을 사용할 때 null에 쓸 문자열 값입니다. 실제 데이터에 나타나지 않는 값이어야 합니다.

이러한 새 파라미터는 다음과 같은 방법으로 사용할 수 있습니다.

성능 개선을 위한 새로운 옵션

새 커넥터에는 몇 가지 새로운 성능 개선 옵션이 도입되었습니다.

- autopushdown: 기본적으로 활성화됩니다.
- autopushdown.s3_result_cache: 기본적으로 비활성화됩니다.
- unload_s3_format: 기본적으로 PARQUET입니다.

이러한 옵션을 사용하는 방법에 대한 자세한 내용은 Apache Spark용 Amazon Redshift 통합을 참조하세요. 캐시된 결과에 오래된 정보가 포함될 수 있으므로 읽기 작업과 쓰기 작업이 혼합된 경우 autopushdown.s3_result_cache를 켜지 않는 것이 좋습니다. 성능을 개선하고 스토리지 비용을 절감하기 위해 UNLOAD 명령의 unload_s3_format 옵션은 기본적으로 PARQUET로 설정됩니다. UNLOAD 명령 기본 동작을 사용하려면 옵션을 TEXT로 재설정합니다.

읽기를 위한 새 암호화 옵션

Amazon Redshift 테이블에서 데이터를 읽을 때 AWS Glue가 사용하는 임시 폴더의 데이터는 기본값으로 SSE-S3 암호화를 사용하여 암호화됩니다. AWS Key Management Service(AWS KMS)의 고객 관리 키를 사용하여 데이터를 암호화하려면 AWS Glue 3.0 버전의 레거시 설정 옵션 ("extraunloadoptions" # s"ENCRYPTED KMS_KEY_ID '$kmsKey'") 대신 ksmKey가 AWS KMS의 키 ID인 ("sse_kms_key" # kmsKey)를 설정할 수 있습니다.

```
datasource0 = glueContext.create_dynamic_frame.from_catalog(
  database = "database-name",
  table_name = "table-name",
  redshift_tmp_dir = args["TempDir"],
  additional_options = {"sse_kms_key":"<KMS_KEY_ID>"},
  transformation_ctx = "datasource0"
)
```

IAM 기반 JDBC URL 지원

새 커넥터는 IAM 기반 JDBC URL을 지원하므로 사용자 및 암호 또는 보안 암호를 전달할 필요가 없습니다. IAM 기반 JDBC URL을 사용하는 커넥터에서는 작업 런타임 역할을 사용하여 Amazon Redshift 데이터 소스에 액세스합니다.

1단계: 다음과 같은 최소 필수 정책을 AWS Glue 작업 런타임 역할에 연결합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentials",
            "Resource": [
                "arn:aws:redshift:<region>:<account>:dbgroup:<cluster name>/*",
                "arn:aws:redshift:*:<account>:dbuser:*/*",
                "arn:aws:redshift:<region>:<account>:dbname:<cluster name>/<database
 name>"
```

```
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "redshift:DescribeClusters",
            "Resource": "*"
        }
    ]
}
```

2단계: IAM 기반 JDBC URL을 다음과 같이 사용합니다. 연결하려는 Amazon Redshift 사용자 이름을 사용하여 새 옵션 DbUser를 지정합니다.

```
conn_options = {
     // IAM-based JDBC URL
    "url": "jdbc:redshift:iam://<cluster name>:<region>/<database name>",
    "dbtable": dbtable,
    "redshiftTmpDir": redshiftTmpDir,
    "aws_iam_role": aws_iam_role,
    "DbUser": "<Redshift User name>" // required for IAM-based JDBC URL
    }

redshift_write = glueContext.write_dynamic_frame.from_options(
    frame=dyf,
    connection_type="redshift",
    connection_options=conn_options
)

redshift_read = glueContext.create_dynamic_frame.from_options(
    connection_type="redshift",
    connection_options=conn_options
)
```

> **Note**
>
> DynamicFrame은 현재 GlueContext.create_dynamic_frame.from_options 워크플로에 DbUser가 있는 IAM 기반 JDBC URL만 지원합니다.

AWS Glue 3.0 버전에서 4.0 버전으로 마이그레이션

AWS Glue 4.0에서 ETL 작업은 다양한 옵션 및 구성을 통해 새로운 Amazon Redshift Spark 커넥터 및 새로운 JDBC 드라이버에 액세스할 수 있습니다. 새로운 Amazon Redshift 커넥터 및 드라이버는 성능을 염두에 두고 작성되었으며 데이터의 트랜잭션 일관성을 유지합니다. 이러한 제품은 Amazon Redshift 설명서에 문서화되어 있습니다. 자세한 내용은 다음을 참조하세요.

- Apache Spark용 Amazon Redshift 통합
- Amazon Redshift JDBC 드라이버, 버전 2.1

테이블/열 이름 및 식별자 제한

새로운 Amazon Redshift Spark 커넥터 및 드라이버에서는 Redshift 테이블 이름에 대한 요구 사항이 더 제한적입니다. 자세한 내용은 Amazon Redshift 테이블 이름을 정의하기 위한 이름 및 식별자를 참조하세요. 규칙을 따르지 않는 테이블 이름 및 특정 문자(예: 공백)에서는 작업 북마크 워크플로가 작동하지 않을 수 있습니다.

이름 및 식별자 규칙을 따르지 않는 이름을 가진 레거시 테이블이 있고 북마크에 문제가 있는 경우(이전 Amazon Redshift 테이블 데이터를 재처리하는 작업) 테이블 이름을 변경하는 것이 좋습니다. 자세한 내용은 ALTER TABLE 예를 참조하세요.

DataFrame의 기본 tempformat 변경

AWS Glue 3.0 버전 Spark 커넥터에서는 Amazon Redshift Redshift에 쓰는 동안 기본적으로 `tempformat`이 CSV로 지정됩니다. 일관성을 유지하기 위해 AWS Glue 3.0 버전에서는 `DynamicFrame`의 기본 `tempformat`로 `CSV`를 계속 사용합니다. 이전에 Amazon Redshift Spark 커넥터에서 Spark Dataframe API를 직접 사용한 적이 있다면 `DataframeReader` 및 `Writer` 옵션에서 `tempformat`을 CSV로 명시적으로 설정할 수 있습니다. 그렇지 않으면 새 Spark 커넥터에서 `tempformat`이 기본적으로 `AVRO`로 설정됩니다.

동작 변경: Amazon Redshift 데이터 형식 REAL을 DOUBLE 대신 Spark 데이터 형식 FLOAT에 매핑합니다.

AWS Glue 3.0 버전에서는 Amazon Redshift `REAL`이 Spark `DOUBLE` 유형으로 변환됩니다. Amazon Redshift `REAL` 유형과 Spark `FLOAT` 유형 간에 상호 변환되도록 새로운 Amazon Redshift Spark 커넥터의 동작을 업데이트했습니다. 레거시 사용 사례에서 Amazon Redshift`REAL` 유형을 Spark`DOUBLE` 유형에 계속 매핑하려는 경우 다음 해결 방법을 사용할 수 있습니다.

- `DynamicFrame`의 경우 `DynamicFrame.ApplyMapping`을 사용하여 `Float` 유형을 `Double` 유형에 매핑합니다. `Dataframe`의 경우 `cast`를 사용해야 합니다.

코드 예제:

```
dyf_cast = dyf.apply_mapping([('a', 'long', 'a', 'long'), ('b', 'float', 'b',
 'double')])
```

VARBYTE 데이터 형식 처리

AWS Glue 3.0 및 Amazon Redshift 데이터 형식으로 작업할 때, AWS Glue 3.0은 Amazon Redshift VARBYTE를 Spark STRING 형식으로 변환합니다. 그러나 최신 Amazon Redshift Spark 커넥터는 VARBYTE 데이터 형식을 지원하지 않습니다. 이 제한을 해결하려면 VARBYTE 열을 지원되는 데이터 형식으로 변환하는 Redshift 뷰를 생성합니다. 그런 다음 새 커넥터를 사용하여 원래 테이블 대신 이 뷰에서 데이터를 로드하면 호환성을 보장하는 동시에 VARBYTE 데이터에 대한 액세스를 유지할 수 있습니다.

Redshift 쿼리의 예:

```
CREATE VIEW view_name AS SELECT FROM_VARBYTE(varbyte_column, 'hex') FROM table_name
```

Kafka 연결

Data Catalog 테이블에 저장된 정보를 사용하거나 데이터 스트림에 직접 액세스할 수 있는 정보를 제공하여 Kafka 연결을 통해 Kafka Data Streams에서 읽고 쓸 수 있습니다. 연결에서는 Kafka 클러스터 또는 Amazon Managed Streaming for Apache Kafka 클러스터를 지원합니다. Kafka의 정보를 Spark DataFrame으로 읽은 다음 AWS Glue DynamicFrame으로 변환할 수 있습니다. DynamicFrame을 JSON 형식으로 Kafka에 쓸 수 있습니다. 데이터 스트림에 직접 액세스하는 경우 이러한 옵션을 사용하여 데이터 스트림에 액세스하는 방법에 대한 정보를 제공합니다.

getCatalogSource 또는 create_data_frame_from_catalog를 사용하여 Kafka 스트리밍 소스에서 레코드를 소비하거나 getCatalogSink 또는 write_dynamic_frame_from_catalog를 사용하여 Kafka에 레코드를 쓰는 경우 작업에는 데이터 카탈로그 데이터베이스와 테이블 이름 정보가 있으며 이를 사용하여 Kafka 스트리밍 소스에서 읽기 위한 몇 가지 기본 파라미터를 얻을 수 있습니다. getSource, getCatalogSink, getSourceWithFormat, getSinkWithFormat, createDataFrameFromOptions, create_data_frame_from_options 또는 write_dynamic_frame_from_catalog를 사용하는 경우 여기에 설명된 연결 옵션을 통해 이러한 기본 파라미터를 지정해야 합니다.

GlueContext 클래스에 지정된 메서드에 대해 다음 인수를 사용하여 Kafka에 대한 연결 옵션을 지정할 수 있습니다.

- Scala

  - connectionOptions: getSource, createDataFrameFromOptions, getSink와(과) 함께 사용
  - additionalOptions: getCatalogSource, getCatalogSink와 함께 사용
  - options: getSourceWithFormat, getSinkWithFormat와 함께 사용
- Python
  - connection_options: create_data_frame_from_options, write_dynamic_frame_from_options와 함께 사용
  - additional_options: create_data_frame_from_catalog, write_dynamic_frame_from_catalog와 함께 사용
  - options: getSource, getSink와 함께 사용

스트리밍 ETL 작업에 대한 참고 및 제한 사항은 the section called “스트리밍 ETL 참고 사항 및 제한 사항” 섹션을 참조하세요.

Kafka 구성

인터넷을 통해 사용할 수 있는 Kafka 스트림에 연결하기 위한 AWS의 필수 조건은 없습니다.

AWS Glue Kafka 연결을 생성하여 연결 보안 인증을 관리할 수 있습니다. 자세한 내용은 the section called “Kafka 데이터 스트림에 대한 연결 생성” 단원을 참조하십시오. AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공하고 메서드 직접 호출에서 connectionName 파라미터로 *ConnectionName*을 제공합니다.

경우에 따라 추가 필수 조건을 구성해야 합니다.

- IAM 인증과 함께 Amazon Managed Streaming for Apache Kafka를 사용하는 경우 적절한 IAM 구성이 필요합니다.
- Amazon VPC 내에서 Amazon Managed Streaming for Apache Kafka를 사용하는 경우 적절한 Amazon VPC 구성이 필요합니다. Amazon VPC 연결 정보를 제공하는 AWS Glue 연결을 생성해야 합니다. AWS Glue 연결을 추가 네트워크 연결로 포함하려면 작업 구성이 필요합니다.

스트리밍 ETL 작업 필수 조건에 대한 자세한 내용은 the section called “스트리밍 ETL 작업” 섹션을 참조하세요.

예제: Kafka 스트림에서 읽기

the section called “forEachBatch”과(와) 함께 사용합니다.

Kafka 스트리밍 소스의 예:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

예: Kafka 스트림에 쓰기

Kafka에 쓰기의 예제:

`getSink` 메서드를 사용한 예제:

```
data_frame_datasource0 =
glueContext.getSink(
 connectionType="kafka",
 connectionOptions={
  JsonOptions("""{
   "connectionName": "ConfluentKafka",
   "classification": "json",
   "topic": "kafka-auth-topic",
   "typeOfData": "kafka"}
  """)},
 transformationContext="dataframe_ApacheKafka_node1711729173428")
 .getDataFrame()
```

`write_dynamic_frame.from_options` 메서드를 사용한 예제:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.write_dynamic_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Kafka 연결 옵션 참조

읽을 때는 `"connectionType": "kafka"`를 사용한 다음 연결 옵션을 사용합니다.

- `"bootstrap.servers"` (필수) 부트스트랩 서버 URL의 목록입니다(예: `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`). 이 옵션은 API 호출에 지정하거나 데이터 카탈로그의 테이블 메타데이터에 정의해야 합니다.
- `"security.protocol"` (필수) 브로커와 통신하는 데 사용되는 프로토콜입니다. 가능한 값은 `"SSL"` 또는 `"PLAINTEXT"`입니다.
- `"topicName"`: (필수) 쉼표로 구분된 구독할 주제의 목록입니다. `"topicName"`, `"assign"` 또는 `"subscribePattern"` 중에서 하나만 지정해야 합니다.
- `"assign"`: (필수) 사용할 특정 `TopicPartitions`을 지정하는 JSON 문자열입니다. `"topicName"`, `"assign"` 또는 `"subscribePattern"` 중에서 하나만 지정해야 합니다.

  예: `'{"topicA":[0,1],"topicB":[2,4]}'`
- `"subscribePattern"`: (필수) 구독할 주제 목록을 식별하는 Java 정규식 문자열입니다. `"topicName"`, `"assign"` 또는 `"subscribePattern"` 중에서 하나만 지정해야 합니다.

  예: `'topic.*'`
- `"classification"`(필수) 레코드의 데이터에서 사용하는 파일 형식입니다. 데이터 카탈로그를 통해 제공되지 않는 한 필수입니다.
- `"delimiter"`(선택 사항) `classification`이 CSV일 때 사용되는 값 구분 기호입니다. 기본값은 `','`입니다.
- `"startingOffsets"`: (선택 사항) 데이터를 읽을 Kafka 주제의 시작 위치입니다. 가능한 값은 `"earliest"` 또는 `"latest"`입니다. 기본값은 `"latest"`입니다.
- `"startingTimestamp"`: (선택 사항, AWS Glue 버전 4.0 이상에서만 지원됨) 데이터를 읽을 Kafka 주제에 있는 레코드의 타임스탬프입니다. 가능한 값은 `yyyy-mm-ddTHH:MM:SSZ` 패턴에서 UTC 형식의 타임스탬프 문자열입니다(여기서, `Z`는 UTC 시간대 오프셋(+/-)임, 예: '2023-04-04T08:00:00-04:00').

  참고: AWS Glue 스트리밍 스크립트의 연결 옵션 목록에는 'startingOffsets' 또는 'startingTimestamp' 중 하나만 존재할 수 있습니다. 이 두 속성을 모두 포함하면 작업에 실패합니다.
- `"endingOffsets"`: (선택 사항) 일괄 쿼리가 종료되는 엔드포인트입니다. 가능한 값은 `"latest"` 또는 각 `TopicPartition`의 끝 오프셋을 지정하는 JSON 문자열입니다.

  JSON 문자열의 경우 포맷은 `{"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}`입니다. 오프셋으로 값 `-1`은 `"latest"`를 나타냅니다.

- "pollTimeoutMs": (선택 사항) Spark 작업 실행기의 Kafka에서 데이터를 폴링하는 시간 제한(밀리초)입니다. 기본값은 600000입니다.
- "numRetries": (선택 사항) Kafka 오프셋 가져오기에 실패하기 전에 재시도할 횟수입니다. 기본값은 3입니다.
- "retryIntervalMs": (선택 사항) Kafka 오프셋을 가져오기를 다시 시도하기 전에 대기하는 시간(밀리초)입니다. 기본값은 10입니다.
- "maxOffsetsPerTrigger": (선택 사항) 트리거 간격당 처리되는 최대 오프셋 수에 대한 속도 제한입니다. 지정된 총 오프셋 수는 서로 다른 볼륨의 topicPartitions에 비례하여 분할됩니다. 기본값은 null입니다. 즉, 소비자가 알려진 최신 오프셋까지 모든 오프셋을 읽습니다.
- "minPartitions": (선택 사항) Kafka에서 읽을 원하는 최소 파티션 수입니다. 기본값은 null이며 이는 Spark 파티션의 수가 Kafka 파티션의 수와 동일함을 의미합니다.
- "includeHeaders": (선택 사항) Kafka 헤더를 포함할지 여부입니다. 옵션이 "true"로 설정되면 데이터 출력에는 유형이 Array[Struct(key: String, value: String)]인 "glue_streaming_kafka_headers"라는 추가 열이 포함됩니다. 기본값은 "false"입니다. 이 옵션은 AWS Glue 버전 3.0 이상에서만 사용할 수 있습니다.
- "schema": (InferSchema가 false로 설정된 경우 필수) 페이로드를 처리하는 데 사용할 스키마입니다. 분류가 avro인 경우 제공된 스키마는 Avro 스키마 형식이어야 합니다. 분류가 avro가 아닌 경우 제공된 스키마는 DDL 스키마 형식이어야 합니다.

  다음은 스키마의 예입니다.

  Example in DDL schema format

  ```
  'column1' INT, 'column2' STRING , 'column3' FLOAT
  ```

  Example in Avro schema format

  ```
  {
  "type":"array",
  "items":
  {
  "type":"record",
  "name":"test",
  "fields":
  [
    {
      "name":"_id",
      "type":"string"
  ```

```
  },
  {
    "name":"index",
    "type":
    [
      "int",
      "string",
      "float"
    ]
  }
]
}
}
```

- "inferSchema": (선택 사항) 기본값은 'false'입니다. 'true'로 설정하면 스키마가 런타임 시 foreachbatch 내의 페이로드에서 감지됩니다.
- "avroSchema": (더 이상 사용되지 않음) Avro 형식을 사용할 때 Avro 데이터의 스키마를 지정하는 데 사용되는 파라미터입니다. 이 파라미터는 이제 사용 중단되었습니다. schema 파라미터를 사용합니다.
- "addRecordTimestamp": (선택 사항) 이 옵션이 'true'로 설정되면 데이터 출력에는 이름이 '__src_timestamp'라는 추가 열이 포함됩니다. 이 열은 주제에서 해당 레코드를 수신한 시간을 나타냅니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.
- "emitConsumerLagMetrics": (선택 사항) 이 옵션을 'true'로 설정하면 각 배치에 대해 주제에서 수신한 가장 오래된 레코드와 AWS Glue에 도착한 시간 사이의 지표를 CloudWatch로 내보냅니다. 지표의 이름은 'glue.driver.streaming.maxConsumerLagInMs'입니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.

쓸 때는 "connectionType": "kafka"를 사용한 다음 연결 옵션을 사용합니다.

- "connectionName"(필수) Kafka 클러스터에 연결하는 데 사용되는 AWS Glue 연결의 이름(Kafka 소스와 유사).
- "topic"(필수) 주제 열이 존재하는 경우 주제 구성 옵션이 설정되지 않은 한 Kafka에 지정된 행을 쓸 때 해당 값이 주제로 사용됩니다. 즉, topic 구성 옵션이 주제 열을 재정의합니다.
- "partition"(선택 사항) 유효한 파티션 번호가 지정되면 레코드를 보낼 때 해당 partition이 사용됩니다.

  파티션을 지정하지 않았지만 key가 있는 경우 키의 해시를 사용하여 파티션이 선택됩니다.

key와 partition이 모두 존재하지 않으면 파티션에 최소 batch.size 바이트가 생성될 때 변경 사항을 고정 파티셔닝하여 파티션이 선택됩니다.

- "key"(선택 사항) partition이 null인 경우 파티셔닝에 사용됩니다.
- "classification"(선택 사항) 레코드의 데이터에서 사용하는 파일 형식입니다. JSON, CSV, Avro만 지원합니다.

  Avro 형식을 사용하면 직렬화할 사용자 지정 avroSchema를 제공할 수 있지만 역직렬화를 위해서는 소스에서도 이를 제공해야 한다는 점에 유의하세요. 그렇지 않으면 기본적으로 직렬화를 위해 Apache AvroSchema를 사용합니다.

또한 필요에 따라 Kafka 프로듀서 구성 파라미터를 업데이트하여 Kafka 싱크를 미세 튜닝할 수 있습니다. 연결 옵션에는 허용 목록이 없으며 모든 키-값 쌍은 싱크에 그대로 유지됩니다.

그러나 적용되지 않는 일부 거부 옵션의 목록이 있습니다. 자세한 내용은 Kafka 특정 구성을 참조하세요.

Azure Cosmos DB 연결

AWS Glue for Spark를 이용하여 Azure Cosmos DB의 기존 컨테이너에서 읽고 쓸 수 있습니다. 이때 AWS Glue 4.0 이상에서 NoSQL API를 이용합니다. SQL 쿼리를 사용하여 Azure Cosmos DB에서 읽을 내용을 정의할 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 Azure Cosmos DB 키를 사용하여 Azure Cosmos DB에 연결합니다.

NoSQL용 Azure Cosmos DB에 대한 자세한 내용은 Azure 설명서를 참조하십시오.

Azure Cosmos DB 연결 구성

AWS Glue에서 Azure Cosmos DB에 연결하려면 Azure Cosmos DB 키를 만들어 AWS Secrets Manager 암호에 저장한 다음 해당 암호를 Azure Cosmos DB AWS Glue 연결에 연결해야 합니다.

사전 조건:

- Azure에서는 AWS Glue cosmosKey에서 사용할 Azure Cosmos DB 키를 식별하거나 생성해야 합니다. 자세한 내용은 Azure 설명서의 Azure Cosmos DB의 데이터에 대한 보안 액세스를 참조하십시오.

Azure Cosmos DB에 대한 연결 구성 방법:

1. AWS Secrets Manager에서 Azure Cosmos DB 키를 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

   - 키/값 페어를 선택하면 값 *cosmosKey*가 포함된 키 `spark.cosmos.accountKey`에 대한 페어를 생성합니다.

2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때는 Azure Cosmos DB를 선택합니다.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

AWS Glue Azure Cosmos DB 연결을 생성한 후에는 AWS Glue 작업을 실행하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

NoSQL 컨테이너용 Azure Cosmos DB에서 읽기

사전 조건:

- 읽으려는 NoSQL 컨테이너용 Azure Cosmos 데이터베이스. 컨테이너의 식별 정보가 필요합니다.

  NoSQL용 Azure Cosmos 컨테이너는 해당 데이터베이스 및 컨테이너로 식별됩니다. NoSQL API용 Azure Cosmos에 연결할 때 데이터베이스, *cosmosDBName*, 및 컨테이너너, *cosmosContainerName*, 이름을 제공해야 합니다.
- 인증 및 네트워크 위치 정보를 제공하도록 구성된 AWS Glue Azure Cosmos DB 연결입니다. 이 정보를 얻으려면 앞 절차인 Azure Cosmos DB에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
azurecosmos_read = glueContext.create_dynamic_frame.from_options(
```

```
    connection_type="azurecosmos",
    connection_options={
    "connectionName": connectionName,
    "spark.cosmos.database": cosmosDBName,
    "spark.cosmos.container": cosmosContainerName,
    }
)
```

SELECT SQL 쿼리에 반환되는 결과를 필터링하는 쿼리를 제공할 수도 있습니다. query을 구성해야 합니다.

예:

```
azurecosmos_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="azurecosmos",
    connection_options={
        "connectionName": "connectionName",
        "spark.cosmos.database": cosmosDBName,
        "spark.cosmos.container": cosmosContainerName,
        "spark.cosmos.read.customQuery": "query"
    }
)
```

NoSQL 컨테이너용 Azure Cosmos DB에 쓰기

이 예제에서는 기존 DynamicFrame, *dynamicFrame*의 정보를 Azure Cosmos DB에 씁니다. 컨테이너에 이미 정보가 있는 경우 AWS Glue는 DynamicFrame의 데이터를 추가합니다. 컨테이너의 정보가 작성한 정보와 다른 스키마를 사용하는 경우 오류가 발생합니다.

사전 조건:

- 쓰려는 Azure Cosmos DB 테이블. 컨테이너의 식별 정보가 필요합니다. 연결 방법을 호출하기 전에 컨테이너를 만들어야 합니다.

  NoSQL용 Azure Cosmos 컨테이너는 해당 데이터베이스 및 컨테이너로 식별됩니다. NoSQL API용 Azure Cosmos에 연결할 때 데이터베이스, *cosmosDBName*, 및 컨테이너너, *cosmosContainerName*, 이름을 제공해야 합니다.
- 인증 및 네트워크 위치 정보를 제공하도록 구성된 AWS Glue Azure Cosmos DB 연결입니다. 이 정보를 얻으려면 앞 절차인 Azure Cosmos DB에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
azurecosmos_write = glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="azurecosmos",
    connection_options={
    "connectionName": connectionName,
    "spark.cosmos.database": cosmosDBName,
    "spark.cosmos.container": cosmosContainerName
)
```

Azure Cosmos DB 연결 옵션 참조

- `connectionName` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법에 인증 및 네트워크 위치 정보를 제공하도록 구성된 AWS Glue Azure Cosmos DB 연결의 이름입니다.
- `spark.cosmos.database` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 유효한 값: 데이터베이스 이름. NoSQL용 Azure Cosmos DB 데이터베이스 이름입니다.
- `spark.cosmos.container` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 유효한 값: 컨테이너 이름. NoSQL용 Azure Cosmos DB 컨테이너 이름입니다.
- `spark.cosmos.read.customQuery` — 읽기에 사용됩니다. 유효한 값: SQL 쿼리를 선택합니다. 읽을 문서를 선택하기 위한 사용자 지정 쿼리.

Azure SQL 연결

AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 Azure SQL Managed Instances의 테이블에서 읽고 쓸 수 있습니다. SQL 쿼리를 사용하여 Azure SQL에서 읽을 내용을 정의할 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 사용자 이름 및 암호 보안 인증 정보를 사용하여 Azure SQL에 연결할 수 있습니다.

Azure SQL에 대한 자세한 내용은 Azure SQL 설명서를 참조하십시오.

Azure SQL 연결 구성

AWS Glue에서 Azure SQL에 연결하려면 Azure SQL 보안 인증 정보를 만들어 AWS Secrets Manager 암호에 저장한 다음 해당 암호를 Azure SQL AWS Glue 연결에 연결해야 합니다.

Azure SQL에 대한 연결을 구성하는 방법:

1. AWS Secrets Manager에서 Azure SQL 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets

Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

- 키/값 페어를 선택하면 값 *azuresqlUsername*이 포함된 키 user에 대한 페어를 생성합니다.
- 키/값 페어를 선택하면 값 *azuresqlPassword*가 포함된 키 password에 대한 페어를 생성합니다.

2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때 Azure SQL를 선택합니다.
   - Azure SQL URL을 제공할 때는 JDBC 엔드포인트 URL을 제공하십시오.

     목록은 `jdbc:sqlserver://`*databaseServerName*`:`*databasePort*`;databaseName=`*azuresqlDBnam* 형식이어야 합니다.

     AWS Glue에는 다음과 같은 URL 속성이 필요합니다.

     - `databaseName` - 연결할 Azure SQL의 기본 데이터베이스입니다.

     Azure SQL 관리형 인스턴스용 JDBC URL에 대한 자세한 내용은 Microsoft 설명서를 참조하십시오.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

AWS Glue Azure SQL 연결을 생성한 후에는 AWS Glue 작업을 실행하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

Azure SQL 테이블에서 읽는 중

사전 조건:

- 읽으려는 Azure SQL 테이블. 테이블, *databaseName*, *tableIdentifier*에 대한 식별 정보가 필요합니다.

  Azure SQL 테이블은 데이터베이스, 스키마 및 테이블 이름으로 식별됩니다. Azure SQL에 연결할 때 데이터베이스 이름과 테이블 이름을 제공해야 합니다. 스키마가 기본값인 "public"이 아닌 경우에

도 스키마를 제공해야 합니다. 데이터베이스는 *connectionName*의 URL 속성을, dbtable을 통해 스키마 및 테이블 이름을 제공 받습니다.

- 인증 정보를 제공하도록 구성된 AWS Glue Azure SQL 연결입니다. 인증 정보를 구성하려면 앞 절차인 Azure SQL에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
azuresql_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

SELECT SQL 쿼리에 반환되는 결과를 필터링하는 쿼리를 제공할 수도 있습니다. query을 구성해야 합니다.

예:

```
azuresql_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Azure SQL 테이블에 쓰기

이 예제에서는 기존 DynamicFrame, *dynamicFrame*의 정보를 Azure SQL에 씁니다. 테이블에 이미 정보가 있는 경우 AWS Glue는 DynamicFrame의 데이터를 추가합니다.

사전 조건:

- 쓰려는 Azure SQL 테이블. 테이블, *databaseName*, *tableIdentifier*에 대한 식별 정보가 필요합니다.

  Azure SQL 테이블은 데이터베이스, 스키마 및 테이블 이름으로 식별됩니다. Azure SQL에 연결할 때 데이터베이스 이름과 테이블 이름을 제공해야 합니다. 스키마가 기본값인 "public"이 아닌 경우에

도 스키마를 제공해야 합니다. 데이터베이스는 *connectionName*의 URL 속성을, dbtable을 통해 스키마 및 테이블 이름을 제공 받습니다.

- Azure SQL 인증 정보. 인증 정보를 구성하려면 앞 절차인 Azure SQL에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
azuresql_write = glueContext.write_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

Azure SQL 연결 옵션 참조

- connectionName — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법에 인증 정보를 제공하도록 구성된 AWS Glue Azure SQL 연결의 이름입니다.
- databaseName - 읽기/쓰기에 사용됩니다. 유효한 값: Azure SQL 데이터베이스 이름. 연결할 Azure SQL 데이터베이스의 이름입니다.
- dbtable - 쓰기 시 필수, query가 제공되지 않는 한 읽기 전용. 읽기 및 쓰기에 사용됩니다. 유효한 값: Azure SQL 테이블의 이름 또는 마침표로 구분된 스키마/테이블 이름 조합. 연결할 테이블을 식별하는 테이블과 스키마를 지정하는 데 사용됩니다. 기본 스키마는 "public"입니다. 테이블이 기본 스키마가 아닌 스키마를 사용하는 경우 이 정보를 양식 *schemaName*.*tableName*에 입력하십시오.
- query — 읽기에 사용됩니다. Azure SQL에서 읽을 때 검색해야 하는 내용을 정의하는 Transact-SQL SELECT 쿼리입니다. 자세한 내용은 Microsoft 설명서를 참조하십시오.

BigQuery 연결

AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 Google BigQuery의 테이블에서 읽고 쓸 수 있습니다. Google SQL 쿼리를 사용하여 BigQuery에서 읽을 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 보안 인증 정보를 사용하여 BigQuery에 연결합니다.

Google BigQuery에 대한 자세한 내용은 Google Cloud BigQuery 웹사이트를 참조하세요.

BigQuery 연결 구성

AWS Glue에서 Google BigQuery에 연결하려면 AWS Secrets Manager 보안 암호에서 Google Cloud Platform 보안 인증 정보를 생성하고 저장한 다음 해당 보안 암호를 Google BigQuery AWS Glue 연결에 연결해야 합니다.

BigQuery에 대한 연결 구성하기:

1. Google Cloud Platform에서 관련 리소스를 생성하고 식별합니다.
   - 연결하려는 BigQuery 테이블이 포함된 GCP 프로젝트를 생성하거나 식별합니다.
   - BigQuery API를 활성화합니다. 자세한 내용은 BigQuery Storage Read API를 사용하여 테이블 데이터 읽기를 참조하세요.
2. Google Cloud Platform에서 서비스 계정 보안 인증 정보를 생성하고 내보냅니다.

   BigQuery 보안 인증 마법사를 사용하여 보안 인증 정보 생성하기 단계를 신속하게 처리할 수 있습니다.

   GCP에서 서비스 계정을 생성하려면 서비스 계정 생성하기에서 제공되는 튜토리얼을 따르세요.
   - 프로젝트를 선택할 때 BigQuery 테이블이 포함된 프로젝트를 선택합니다.
   - 서비스 계정의 GCP IAM 역할을 선택할 때 BigQuery 작업을 실행하여 BigQuery 테이블을 읽고, 쓰고, 생성할 수 있는 적절한 권한을 부여하는 역할을 추가하거나 생성하세요.

   서비스 계정의 보안 인증 정보를 생성하려면 서비스 계정 키 생성하기에서 제공되는 튜토리얼을 따르세요.
   - 키 유형을 선택할 때 JSON을 선택합니다.

   이제 서비스 계정의 보안 인증 정보가 포함된 JSON 파일을 다운로드했어야 합니다. 예를 들면 다음과 같아야 합니다.

```
{
  "type": "service_account",
  "project_id": "*****",
  "private_key_id": "*****",
  "private_key": "*****",
  "client_email": "*****",
  "client_id": "*****",
```

```
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "*****",
  "universe_domain": "googleapis.com"
}
```

3. base64는 다운로드한 보안 인증 파일을 인코딩합니다. AWS CloudShell 세션 또는 유사한 사항인 경우 명령줄에서 `cat credentialsFile.json | base64 -w 0`을(를) 실행하여 이 작업을 수행할 수 있습니다. 이 명령의 출력인 *credentialString*을 유지합니다.
4. AWS Secrets Manager에서 Google Cloud Platform 보안 인증 정보를 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
   - 키/값 페어를 선택하면 값 *credentialString*이 포함된 키 `credentials`에 대한 페어를 생성합니다.
5. AWS Glue 데이터 카탈로그에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때 Google BigQuery를 선택합니다.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.
6. AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
7. AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

BigQuery 테이블에서 읽기

사전 조건:

- 읽으려는 BigQuery 테이블. 양식 `[dataset].[table]`에 있는 BigQuery 테이블 및 데이터 세트 이름이 필요합니다. 이 *tableName*이라고 부르겠습니다.
- BigQuery 테이블의 청구 프로젝트. 프로젝트 이름인 *parentProject*가 필요합니다. 결제 상위 프로젝트가 없는 경우 테이블이 포함된 프로젝트를 사용합니다.
- BigQuery 인증 정보. AWS Glue로 연결 보안 인증 관리하기 단계를 완료하여 인증 정보를 구성합니다. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
bigquery_read = glueContext.create_dynamic_frame.from_options(
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "sourceType": "table",
        "table": "tableName",
    }
```

DynamicFrame에 반환되는 결과를 필터링하는 쿼리를 제공할 수도 있습니다. `query`, `sourceType`, `viewsEnabled` 및 `materializationDataset`을(를) 구성해야 합니다.

예:

추가 사전 조건:

BigQuery가 쿼리에 대한 구체화된 뷰를 작성할 수 있는 BigQuery 데이터 세트인 *`materializationDataset`*를 생서하거나 식별해야 합니다.

*`materializationDataset`*에서 테이블을 생성하려면 서비스 계정에 적절한 GCP IAM 권한을 부여해야 합니다.

```
glueContext.create_dynamic_frame.from_options(
            connection_type="bigquery",
            connection_options={
                "connectionName": "connectionName",
                "materializationDataset": materializationDataset,
                "parentProject": "parentProject",
                "viewsEnabled": "true",
                "sourceType": "query",
                "query": "select * from bqtest.test"
            }
        )
```

BigQuery 테이블에 쓰기

이 예시는 BigQuery 서비스에 직접 작성합니다. BigQuery는 '간접' 쓰기 메서드도 지원합니다. 간접 쓰기 구성에 대한 자세한 내용은 the section called “Google BigQuery를 통한 간접 쓰기 사용”을(를) 참조하세요.

사전 조건:

- 쓰려는 BigQuery 테이블. 양식 `[dataset].[table]`에 있는 BigQuery 테이블 및 데이터 세트 이름이 필요합니다. 자동으로 생성되는 새 테이블 이름을 제공할 수도 있습니다. 이 *`tableName`*이라고 부르겠습니다.
- BigQuery 테이블의 청구 프로젝트. 프로젝트 이름인 *`parentProject`*가 필요합니다. 결제 상위 프로젝트가 없는 경우 테이블이 포함된 프로젝트를 사용합니다.
- BigQuery 인증 정보. AWS Glue로 연결 보안 인증 관리하기 단계를 완료하여 인증 정보를 구성합니다. AWS Glue 연결의 이름인 *`connectionName`*이 필요합니다.

예:

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "writeMethod": "direct",
        "table": "tableName",
    }
)
```

BigQuery연결 옵션 참조

- `project`— 기본값: Google Cloud 서비스 계정 기본값. 읽기 및 쓰기에 사용됩니다. 테이블과 연결된 Google Cloud 프로젝트의 이름.
- `table` -- (필수) 읽기 및 쓰기에 사용됩니다. `[[project:]dataset.]` 형식에 따른 BigQuery 테이블의 이름.
- `dataset`— `table` 옵션을 통해 정의되지 않은 경우 필수입니다. 읽기 및 쓰기에 사용됩니다. BigQuery 테이블을 포함하는 데이터 세트의 이름.
- `parentProject`— 기본값: Google Cloud 서비스 계정 기본값. 읽기 및 쓰기에 사용됩니다. 청구에 사용되는 `project`와(과) 연결된 Google Cloud 프로젝트의 이름.
- `sourceType` — 읽기에 사용됩니다. 읽을 때 필요합니다. 유효한 값:`table`, `query`은(는) 테이블로 읽을 것인지 쿼리로 읽을 것인지 AWS Glue에 알립니다.

- `materializationDataset` — 읽기에 사용됩니다. 유효한 값: 문자열. 보기의 구체화를 저장하는 데 사용되는 BigQuery 데이터 세트의 이름.
- `viewsEnabled` — 읽기에 사용됩니다. 기본값: false. 유효한 값: 참, 거짓. BigQuery에서 보기를 사용할지 여부를 구성합니다.
- `query` — 읽기에 사용됩니다. `viewsEnabled`이(가) 참일 때 사용됩니다. GoogleSQL DQL 쿼리.
- `temporaryGcsBucket`— 쓰기에 사용됩니다. `writeMethod`이(가) 기본값(`indirect`)으로 설정된 경우에 필요합니다. BigQuery에 쓰는 동안 중간 형식의 데이터를 저장하는 데 사용되는 Google Cloud Storage 버킷의 이름.
- `writeMethod` - 기본값: `indirect`. 유효한 값: `direct`, `indirect`. 쓰기에 사용됩니다. 데이터를 쓰는 데 사용되는 메서드를 지정합니다.
  - `direct`(으)로 설정하면 커넥터가 BigQuery Storage 쓰기 API를 사용하여 데이터를 작성합니다.
  - `indirect`(으)로 설정하면 커넥터가 Google Cloud Storage에 쓴 다음 로드 작업을 사용하여 BigQuery로 전송합니다. Google Cloud 서비스 계정에는 적절한 GCS 권한이 필요합니다.

Google BigQuery를 통한 간접 쓰기 사용

이 예시에서는 Google Cloud Storage에 데이터를 쓰고 Google BigQuery에 데이터를 복사하는 간접 쓰기를 사용합니다.

사전 조건:

임시 Google Cloud Storage 버킷인 *`temporaryBucket`*이 필요합니다.

AWS Glue GCP 서비스 계정의 GCP IAM 역할에는 *`temporaryBucket`*에 액세스하는 적절한 GCS 권한이 필요합니다.

추가 구성:

BigQuery를 사용하여 간접 쓰기 구성하기:

1. the section called “BigQuery 구성”을(를) 평가하고, GCP 보안 인증 JSON 파일을 찾거나 다시 다운로드합니다. 작업에 사용되는 Google BigQuery AWS Glue 연결의 AWS Secrets Manager 보안 암호인 *`secretName`*을(를) 식별합니다.
2. 보안 인증 JSON 파일을 적절하게 안전한 Amazon S3 위치에 업로드합니다. 향후 단계를 위해 파일 경로인 *`s3secretpath`*를 유지합니다.

3. *secretName*을(를) 편집하고 `spark.hadoop.google.cloud.auth.service.account.json.keyfile` 키를 추가합니다. 값을 *s3secretpath*에 설정합니다.
4. AWSGlue 작업 Amazon S3 IAM에 *s3secretpath*에 액세스할 수 있는 권한을 부여합니다.

이제 쓰기 메서드에 임시 GCS 버킷 위치를 제공할 수 있습니다. `indirect`은(는) 이전에는 기본값이었기 때문에 `writeMethod`을(를) 제공하지 않아도 됩니다.

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "temporaryGcsBucket": "temporaryBucket",
        "table": "tableName",
    }
)
```

JDBC 연결

일반적으로 특정 관계형 데이터베이스 유형은 JDBC 표준을 통한 연결을 지원합니다. JDBC에 대한 자세한 내용은 Java JDBC API 설명서를 참조하세요. AWS Glue는 기본적으로 JDBC 커넥터를 통해 특정 데이터베이스 연결을 지원합니다. JDBC 라이브러리는 AWS Glue Spark 작업에서 제공됩니다. AWS Glue 라이브러리를 사용하여 이러한 데이터베이스 유형에 연결할 때 표준 옵션 세트에 액세스할 수 있습니다.

JDBC connectionType 값은 다음과 같습니다.

- `"connectionType": "sqlserver"`: Microsoft SQL Server 데이터베이스에 연결을 지정합니다.
- `"connectionType": "mysql"`: MySQL 데이터베이스에 대한 연결을 지정합니다.
- `"connectionType": "oracle"`: Oracle Database에 연결을 지정합니다.
- `"connectionType": "postgresql"`: PostgreSQL 데이터베이스에 대한 연결을 지정합니다.
- `"connectionType": "redshift"`: Amazon Redshift 데이터베이스에 대한 연결을 지정합니다. 자세한 내용은 the section called “Redshift 연결” 섹션을 참조하세요.

다음 표에는 AWS Glue에서 지원하는 JDBC 드라이버 버전 목록이 있습니다.

Product	Glue 5.0용 JDBC 드라이버 버전	Glue 4.0용 JDBC 드라이버 버전	Glue 3.0용 JDBC 드라이버 버전	Glue 0.9, 1.0, 2.0용 JDBC 드라이버 버전
Microsoft SQL Server	10.2.0	9.4.0	7.x	6.x
MySQL	8.0.33	8.0.23	8.0.23	5.1
Oracle Database	23.3.0.23.09	21.7	21.1	11.2
PostgreSQL	42.7.3	42.3.6	42.2.18	42.1.x
Amazon Redshift *	redshift-jdbc42-2.1.0.29	redshift-jdbc42-2.1.0.16	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017

* Amazon Redshift 연결 유형의 경우 형식 지정 옵션을 비롯하여 JDBC 연결에 대한 연결 옵션에 포함된 다른 모든 옵션 이름 및 값 페어는 기본 SparkSQL DataSource에 직접 전달됩니다. AWS Glue 4.0 이상 버전의 Spark 작업을 포함하는 AWS Glue에서 Amazon Redshift용 AWS Glue 네이티브 커넥터는 Apache Spark용 Amazon Redshift 통합을 사용합니다. 자세한 내용은 Apache Spark용 Amazon Redshift 통합을 참조하세요. 이전 버전인 경우 Amazon Redshift data source for Spark를 참조하세요.

JDBC를 사용하여 Amazon RDS 데이터 스토어에 연결하도록 Amazon VPC를 구성하려면 the section called “Amazon RDS 데이터 스토어에 연결하도록 Amazon VPC 설정” 섹션을 참조하세요.

> **Note**
>
> AWS Glue 작업은 실행 중에 하나의 서브넷에만 연결됩니다. 이로 인해 동일한 작업을 통해 여러 데이터 소스에 연결하는 기능이 영향을 받을 수 있습니다. 이 동작은 JDBC 소스에만 국한되지 않습니다.

주제

- JDBC 연결 옵션 참조
- sampleQuery 사용
- 사용자 지정 JDBC 드라이버 사용

- JDBC 테이블을 병렬로 읽기
- AWS Glue에서 Amazon RDS 데이터 스토어에 대해 JDBC를 연결하도록 Amazon VPC 설정

JDBC 연결 옵션 참조

이미 JDBC AWS Glue 연결을 정의한 경우 URL, 사용자 및 암호와 같은 JDBC 연결에 정의된 구성 속성을 재사용할 수 있으므로 코드에서 연결 옵션으로 지정할 필요가 없습니다. 이 기능은 AWS Glue 3.0 이상 버전에서 사용할 수 있습니다. 이렇게 하려면 다음 연결 속성을 사용합니다.

- `"useConnectionProperties"`: 연결에서 구성을 사용하려는 경우 이 값을 'true'로 설정합니다.
- `"connectionName"`: 구성을 검색할 연결 이름을 입력합니다. 연결은 작업과 동일한 영역에 정의되어 있어야 합니다.

이 연결 옵션을 JDBC 연결에 사용하십시오.

- `"url"`: (필수) 데이터베이스의 JDBC URL.
- `"dbtable"`: (필수) 읽을 데이터베이스 테이블입니다. 데이터베이스 내의 스키마를 지원하는 JDBC 데이터 스토어의 경우 `schema.table-name`에 대해 지정합니다. 스키마가 제공되지 않으면 기본 "퍼블릭" 스키마가 사용됩니다.
- `"user"`: (필수 사항) 연결할 때 사용할 사용자 이름입니다.
- `"password"`: (필수) 연결할 때 사용할 암호.
- (선택 사항) 다음 옵션을 사용하면 사용자 정의 JDBC 드라이버를 제공할 수 있습니다. AWS Glue에서 기본적으로 지원하지 않는 드라이버를 사용해야 한다면 이 옵션을 사용합니다.

  ETL 작업은 원본과 대상이 같은 데이터베이스 제품이어도 데이터 원본과 대상에 여러 JDBC 드라이버 버전을 사용할 수 있습니다. 따라서 버전이 다른 원본 데이터베이스와 대상 데이터베이스 간에 데이터를 마이그레이션할 수 있습니다. 이 옵션을 사용하려면 먼저 Amazon S3에 JDBC 드라이버의 JAR 파일을 업로드해야 합니다.

  - `"customJdbcDriverS3Path"`: 사용자 정의 JDBC 드라이버의 Amazon S3 경로입니다.
  - `"customJdbcDriverClassName"`: JDBC 드라이버의 클래스 이름입니다.
- `"bulkSize"`: (선택 사항) JDBC 대상에 신속하게 대량 로드하기 위해 병렬 삽입을 구성하는 데 사용됩니다. 데이터를 쓰거나 삽입할 때 사용할 병렬 처리 수준의 정수 값을 지정합니다. 이 옵션은 Arch User Repository(AUR)와 같은 데이터베이스에 대한 쓰기 성능을 향상시키는 데 유용합니다.
- `"hashfield"`(선택 사항) JDBC 테이블에서 병렬로 읽을 때 데이터를 파티션으로 구분하는 데 사용할 JDBC 테이블의 열 이름을 지정하기 위해 사용하는 문자열입니다. 'hashfield' 또는

'hashexpression'을 제공합니다. 자세한 내용은 the section called “JDBC를 병렬로 읽기” 섹션을 참조하세요.

- "hashexpression"(선택 사항) 정수를 반환하는 SQL select 절입니다. JDBC 테이블에서 병렬로 읽을 때 JDBC 테이블의 데이터를 파티션으로 구분하는 데 사용됩니다. 'hashfield' 또는 'hashexpression'을 제공합니다. 자세한 내용은 the section called “JDBC를 병렬로 읽기” 섹션을 참조하세요.
- "hashpartitions"(선택 사항) 양의 정수입니다. JDBC 테이블에서 병렬로 읽을 때 JDBC 테이블의 병렬 읽기 수를 지정하는 데 사용됩니다. 기본값: 7. 자세한 내용은 the section called “JDBC를 병렬로 읽기” 섹션을 참조하세요.
- "sampleQuery": (선택 사항) 사용자 지정 SQL 쿼리 문입니다. 테이블의 정보 하위 세트를 지정하여 테이블 콘텐츠의 샘플을 검색하는 데 사용됩니다. 데이터를 고려하지 않고 구성하면 DynamicFrame 메서드보다 효율성이 떨어져 시간 초과나 메모리 부족 오류가 발생할 수 있습니다. 자세한 내용은 the section called “sampleQuery 사용” 섹션을 참조하세요.
- "enablePartitioningForSampleQuery": (선택 사항) 부울입니다. 기본값: false. sampleQuery를 지정할 때 JDBC 테이블에서 병렬 읽기를 활성화하는 데 사용됩니다. true로 설정된 경우 AWS Glue에서 분할 조건을 추가하려면 **sampleQuery**가 'where' 또는 'and'로 끝나야 합니다. 자세한 내용은 the section called “sampleQuery 사용” 섹션을 참조하세요.
- "sampleSize": (선택 사항) 양의 정수입니다. 샘플 쿼리에서 반환되는 행의 수를 제한합니다. enablePartitioningForSampleQuery가 true인 경우에만 작동합니다. 분할이 활성화되지 않은 경우 대신 sampleQuery에서 "limit x"를 직접 추가하여 크기를 제한해야 합니다. 자세한 내용은 the section called “sampleQuery 사용” 섹션을 참조하세요.

sampleQuery 사용

이 섹션에서는 sampleQuery, enablePartitioningForSampleQuery 및 sampleSize 사용 방법을 설명합니다.

sampleQuery는 데이터 세트의 몇 개 행을 샘플링할 때 효율적입니다. 기본적으로 쿼리는 단일 실행기에 의해 실행됩니다. 데이터를 고려하지 않고 구성하면 DynamicFrame 메서드보다 효율성이 떨어져 시간 초과나 메모리 부족 오류가 발생할 수 있습니다. ETL 파이프라인의 일부로 기본 데이터베이스에서 SQL을 실행하는 방법은 일반적으로 성능 목적으로만 필요합니다. 데이터 세트의 몇 개 행을 미리 보려는 경우 the section called “show” 사용을 고려합니다. SQL을 사용하여 데이터 세트를 변환하려는 경우 DataFrame 양식의 데이터에 대해 SparkSQL 변환을 정의하는 데 the section called “toDF” 사용을 고려합니다.

쿼리가 다양한 테이블을 조작할 수 있지만 dbtable이 여전히 필요합니다.

sampleQuery를 사용하여 테이블 샘플 검색

기본 sampleQuery 동작을 사용하여 데이터 샘플을 검색할 때 AWS Glue는 뛰어난 처리량을 기대하지 않으므로 단일 실행기에서 쿼리를 실행합니다. 제공하는 데이터를 제한하고 성능 문제를 일으키지 않으려면 SQL에 `LIMIT` 절을 제공하는 것이 좋습니다.

Example 분할 없이 SampleQuery 사용

다음 코드 예제에서는 분할 없이 `sampleQuery`를 사용하는 방법을 보여줍니다.

```
//A full sql query statement.
val query = "select name from $tableName where age > 0 limit 1"
val connectionOptions = JsonOptions(Map(
    "url" -> url,
    "dbtable" -> tableName,
    "user" -> user,
    "password" -> password,
    "sampleQuery" -> query ))
val dyf = glueContext.getSource("mysql", connectionOptions)
          .getDynamicFrame()
```

대규모 데이터 세트에서 sampleQuery 사용

대규모 데이터 세트를 읽는 경우 JDBC 분할을 활성화하여 테이블을 병렬로 쿼리해야 할 수 있습니다. 자세한 내용은 the section called “JDBC를 병렬로 읽기” 섹션을 참조하세요. JDBC 분할과 함께 `sampleQuery`를 사용하려면 `enablePartitioningForSampleQuery`를 true로 설정합니다. 이 기능을 활성화하려면 `sampleQuery`를 일부 변경해야 합니다.

`sampleQuery`와 함께 JDBC 분할을 사용하는 경우 AWS Glue에서 분할 조건을 추가하려면 쿼리가 'where' 또는 'and'로 끝나야 합니다.

JDBC 테이블에서 병렬로 읽을 때 sampleQuery의 결과를 제한하려면 `LIMIT` 절을 지정하는 대신, `"sampleSize"` 파라미터를 설정합니다.

Example JDBC 분할과 함께 sampleQuery 사용

다음 코드 예제는 JDBC 분할과 함께 `sampleQuery`를 사용하는 방법을 보여줍니다.

```
//note that the query should end with "where" or "and" if use with JDBC partitioning.
val query = "select name from $tableName where age > 0 and"

//Enable JDBC partitioning by setting hashfield.
//to use sampleQuery with partitioning, set enablePartitioningForSampleQuery.
```

```
//use sampleSize to limit the size of returned data.
val connectionOptions = JsonOptions(Map(
    "url" -> url,
    "dbtable" -> tableName,
    "user" -> user,
    "password" -> password,
    "hashfield" -> primaryKey,
    "sampleQuery" -> query,
    "enablePartitioningForSampleQuery" -> true,
    "sampleSize" -> "1" ))
val dyf = glueContext.getSource("mysql", connectionOptions)
          .getDynamicFrame()
```

규칙 및 제한:

샘플 쿼리는 작업 북마크와 함께 사용할 수 없습니다. 두 가지 모두에 대한 구성이 제공되면 북마크 상태는 무시됩니다.

사용자 지정 JDBC 드라이버 사용

다음 코드 예제에서는 사용자 지정 JDBC 드라이버로 JDBC 데이터베이스에서 읽고 쓰는 방법을 보여줍니다. 한 버전의 데이터베이스 제품에서 읽고 같은 제품의 나중 버전에 쓰는 작업을 볼 수 있습니다.

Python

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext, SparkConf
from awsglue.context import GlueContext
from awsglue.job import Job
import time
from pyspark.sql.types import StructType, StructField, IntegerType, StringType

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

# Construct JDBC connection options
connection_mysql5_options = {
    "url": "jdbc:mysql://<jdbc-host-name>:3306/db",
    "dbtable": "test",
    "user": "admin",
```

```
    "password": "pwd"}

connection_mysql8_options = {
    "url": "jdbc:mysql://<jdbc-host-name>:3306/db",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd",
    "customJdbcDriverS3Path": "s3://amzn-s3-demo-bucket/mysql-connector-
java-8.0.17.jar",
    "customJdbcDriverClassName": "com.mysql.cj.jdbc.Driver"}

connection_oracle11_options = {
    "url": "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd"}

connection_oracle18_options = {
    "url": "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd",
    "customJdbcDriverS3Path": "s3://amzn-s3-demo-bucket/ojdbc10.jar",
    "customJdbcDriverClassName": "oracle.jdbc.OracleDriver"}

# Read from JDBC databases with custom driver
df_mysql8 = glueContext.create_dynamic_frame.from_options(connection_type="mysql",

 connection_options=connection_mysql8_options)

# Read DynamicFrame from MySQL 5 and write to MySQL 8
df_mysql5 = glueContext.create_dynamic_frame.from_options(connection_type="mysql",

 connection_options=connection_mysql5_options)
glueContext.write_from_options(frame_or_dfc=df_mysql5, connection_type="mysql",
                               connection_options=connection_mysql8_options)

# Read DynamicFrame from Oracle 11 and write to Oracle 18
df_oracle11 =
 glueContext.create_dynamic_frame.from_options(connection_type="oracle",

 connection_options=connection_oracle11_options)
glueContext.write_from_options(frame_or_dfc=df_oracle11, connection_type="oracle",
                               connection_options=connection_oracle18_options)
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {
  val MYSQL_5_URI: String = "jdbc:mysql://<jdbc-host-name>:3306/db"
  val MYSQL_8_URI: String = "jdbc:mysql://<jdbc-host-name>:3306/db"
  val ORACLE_11_URI: String = "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL"
  val ORACLE_18_URI: String = "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL"

  // Construct JDBC connection options
  lazy val mysql5JsonOption = jsonOptions(MYSQL_5_URI)
  lazy val mysql8JsonOption = customJDBCDriverJsonOptions(MYSQL_8_URI, "s3://amzn-
s3-demo-bucket/mysql-connector-java-8.0.17.jar", "com.mysql.cj.jdbc.Driver")
  lazy val oracle11JsonOption = jsonOptions(ORACLE_11_URI)
  lazy val oracle18JsonOption = customJDBCDriverJsonOptions(ORACLE_18_URI, "s3://
amzn-s3-demo-bucket/ojdbc10.jar", "oracle.jdbc.OracleDriver")

  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Read from JDBC database with custom driver
    val df_mysql8: DynamicFrame = glueContext.getSource("mysql",
 mysql8JsonOption).getDynamicFrame()

    // Read DynamicFrame from MySQL 5 and write to MySQL 8
    val df_mysql5: DynamicFrame = glueContext.getSource("mysql",
 mysql5JsonOption).getDynamicFrame()
    glueContext.getSink("mysql", mysql8JsonOption).writeDynamicFrame(df_mysql5)
```

```
    // Read DynamicFrame from Oracle 11 and write to Oracle 18
    val df_oracle11: DynamicFrame = glueContext.getSource("oracle",
 oracle11JsonOption).getDynamicFrame()
    glueContext.getSink("oracle", oracle18JsonOption).writeDynamicFrame(df_oracle11)

    Job.commit()
  }

  private def jsonOptions(url: String): JsonOptions = {
    new JsonOptions(
      s"""{"url": "${url}",
         |"dbtable":"test",
         |"user": "admin",
         |"password": "pwd"}""".stripMargin)
  }

  private def customJDBCDriverJsonOptions(url: String, customJdbcDriverS3Path:
 String, customJdbcDriverClassName: String): JsonOptions = {
    new JsonOptions(
      s"""{"url": "${url}",
         |"dbtable":"test",
         |"user": "admin",
         |"password": "pwd",
         |"customJdbcDriverS3Path": "${customJdbcDriverS3Path}",
         |"customJdbcDriverClassName" :
 "${customJdbcDriverClassName}"}""".stripMargin)
  }
}
```

JDBC 테이블을 병렬로 읽기

JDBC 테이블의 속성을 AWS Glue에서 분할된 데이터를 병렬로 읽도록 설정할 수 있습니다. 특정 속성을 설정할 때 AWS Glue에게 데이터의 논리적 파티션에 대해 병렬 SQL 쿼리를 실행하도록 지시합니다. 해시 필드 또는 해시 표현식을 설정하여 파티셔닝을 제어할 수 있습니다. 데이터에 액세스할 때 사용되는 병렬 읽기의 수도 제어할 수 있습니다.

JDBC 테이블에서 병렬로 읽는 것은 성능을 향상시킬 수 있는 최적화 기법입니다. 이 기법이 적절한 시기를 식별하는 프로세스에 대한 자세한 내용은 AWS 권장 가이드의 Apache Spark용 AWS Glue 작업 성능 조정 모범 사례 가이드에 있는 데이터 스캔량 감소를 참조하세요.

병렬 읽기를 활성화하려면 테이블 구조의 파라미터 필드에서 키 값 페어를 설정합니다. JSON 표기법을 사용하여 테이블 파라미터 필드 값을 설정합니다. 테이블 속성을 편집하는 방법에 대한 더 자세한 정보는 테이블 세부 정보 보기 및 편집를 참조하십시오. ETL(추출, 변환 및 로드) 방법 `create_dynamic_frame_from_options` 및 `create_dynamic_frame_from_catalog`을 호출할 때도 병렬 읽기를 활성화할 수 있습니다 이러한 방법에서 옵션을 지정하는 방법은 from_options 및 from_catalog 단원을 참조하십시오.

이 방법을 JDBC 테이블, 즉 베이스 데이터가 JDBC 데이터 스토어인 대부분의 테이블에 사용할 수 있습니다. Amazon Redshift 및 Amazon S3 테이블을 읽을 때 이러한 속성은 무시됩니다.

hashfield

`hashfield`을 사용할 JDBC 테이블의 열 이름으로 설정하여 데이터를 파티션으로 분할합니다. 파티션 사이로 데이터를 분배할 값이 이 열에 고르게 분포하는 것이 가장 좋습니다. 이 열은 데이터 형식이 될 수 있습니다. AWS Glue는 이 열로 분할된 데이터를 읽기 위해 병렬로 실행되는 비-중첩 쿼리를 생성합니다. 예를 들어, 데이터가 월별로 고르게 분포할 경우 `month` 열을 사용하여 각 월의 데이터를 병렬로 읽을 수 있습니다.

```
'hashfield': 'month'
```

AWS Glue은 쿼리를 생성하여 필드 값을 파티션 번호로 해싱하고 전체 파티션의 질의를 병렬로 실행합니다. 사용자의 쿼리를 사용하여 테이블 읽기를 분할하려면 `hashfield` 대신 `hashexpression`를 제공합니다.

hashexpression

`hashexpression`을 자연수를 반환하는 SQL 표현식(JDBC 데이터베이스 엔진 문법 준수)으로 설정합니다. 간단한 표현식은 테이블의 숫자 열 이름입니다. AWS Glue는 `WHERE` 절의 `hashexpression`를 사용하여 JDBC 데이터를 병렬로 읽어 데이터를 분할할 SQL 쿼리를 생성합니다.

예를 들어, 숫자 열 `customerID`를 사용하여 고객 번호로 분할된 데이터를 읽습니다.

```
'hashexpression': 'customerID'
```

AWS Glue이 파티셔닝을 제어하게 하려면 `hashexpression` 대신 `hashfield`를 제공합니다.

hashpartitions

`hashpartitions`을 JDBC 테이블 병렬 읽기 번호로 설정합니다. 이 속성을 설정하지 않을 경우 기본값은 7입니다.

예를 들어, 병렬 읽기 번호를 5로 설정하면 AWS Glue는 쿼리가 5개(또는 이하)인 데이터를 읽습니다.

```
'hashpartitions': '5'
```

AWS Glue에서 Amazon RDS 데이터 스토어에 대해 JDBC를 연결하도록 Amazon VPC 설정

JDBC를 사용하여 Amazon RDS의 데이터베이스에 연결하는 경우 추가 설정을 수행해야 합니다. AWS Glue 구성 요소가 Amazon RDS와 통신할 수 있도록 하려면 Amazon VPC에서 Amazon RDS 데이터 스토어에 대한 액세스를 설정해야 합니다. AWS Glue가 구성 요소 간에 통신하려면 보안 그룹이 모든 TCP 포트에 자기 참조 인바운드 규칙을 지정해야 합니다. 자체 참조 규칙을 생성하면 소스를 VPC의 동일한 보안 그룹으로 제한할 수 있습니다. 자체 참조 규칙을 사용한다고 해서 VPC가 모든 네트워크에 개방되는 것은 아닙니다. VPC의 기본 보안 그룹은 ALL Traffic의 자기 참조 인바운드 액세스 규칙이 있습니다.

AWS Glue 및 Amazon RDS 데이터 스토어 간에 액세스를 설정하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔에서 Amazon RDS 데이터베이스에 대한 액세스를 제어하는 데 사용되는 보안 그룹을 식별합니다.

   왼쪽 탐색 창에서 데이터베이스를 선택한 다음 기본 창의 목록에서 연결하려는 인스턴스를 선택합니다.

   데이터베이스 세부 정보 페이지의 연결 및 보안 탭에서 VPC 보안 그룹을 찾습니다.
3. 네트워크 아키텍처를 기반으로 AWS Glue 서비스에 대한 액세스를 허용하도록 수정하는 것이 가장 적합한 관련 보안 그룹을 식별합니다. 나중에 참조할 수 있도록 *database-security-group*이라는 이름을 저장하세요. 적절한 보안 그룹이 없는 경우 Amazon RDS 설명서의 보안 그룹을 생성하여 VPC에서 DB 인스턴스에 대한 액세스 권한 제공 지침을 따르세요.

4. AWS Management Console에 로그인하고 https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
5. Amazon VPC 콘솔에서 *database-security-group*을 업데이트하는 방법을 확인합니다.

   왼쪽 탐색 창에서 보안 그룹을 선택한 다음 기본 창 목록에서 *database-security-group*을 선택합니다.
6. *database-security-group*, *database-sg-id*의 보안 그룹 ID를 식별합니다. 나중에 참조할 수 있도록 저장합니다.

   보안 그룹 세부 정보 페이지에서 보안 그룹 ID를 찾으세요.
7. *database-security-group*의 인바운드 규칙을 변경하고 AWS Glue 구성 요소가 통신할 수 있도록 자체 참조 규칙을 추가합니다. 구체적으로 유형이 `All TCP`, 프로토콜이 `TCP`, 포트 범위가 모든 포트를 포함하고 소스가 *database-sg-id*인 규칙을 추가하거나 있는지 확인합니다. 소스에 입력한 보안 그룹이 편집 중인 보안 그룹과 동일한지 확인합니다.

   보안 그룹 세부 정보 페이지에서 인바운드 규칙 편집을 선택합니다.

   인바운드 규칙은 다음과 비슷하게 보입니다.

유형	프로토콜	포트 범위	소스
모든 TCP	TCP	0~65535	*database-sg-id*

8. 아웃바운드 트래픽에 대한 규칙을 추가합니다.

   보안 그룹 세부 정보 페이지에서 아웃바운드 규칙 편집을 선택합니다.

   보안 그룹에서 모든 아웃바운드 트래픽을 허용하는 경우에는 별도의 규칙이 필요하지 않습니다. 예:

유형	프로토콜	포트 범위	대상
모든 트래픽	ALL	ALL	0.0.0.0/0

   네트워크 아키텍처가 아웃바운드 트래픽을 제한하도록 설계된 경우 다음과 같은 아웃바운드 규칙을 생성하세요.

유형은 All TCP, 프로토콜은 TCP, 포트 범위는 모든 포트를 포함하고 대상은 *database-sg-id*인 자체 참조 규칙을 생성합니다. 대상에 입력한 보안 그룹이 편집 중인 보안 그룹과 동일한지 확인합니다.

Amazon S3 VPC 엔드포인트를 사용하는 경우에는 HTTPS 규칙을 추가하여 VPC에서 Amazon S3로의 트래픽을 허용하세요. 유형은 HTTPS, 프로토콜은 TCP, 포트 범위는 443이고 대상이 Amazon S3 게이트웨이 엔드포인트인 *s3-prefix-list-id*에 대한 관리형 접두사 목록의 ID인 규칙을 생성합니다. 접두사 목록 및 Amazon S3 게이트웨이 엔드포인트에 대한 자세한 내용은 Amazon VPC 설명서에서 Amazon S3에 대한 게이트웨이 엔드포인트를 참조하세요.

예:

유형	프로토콜	포트 범위	대상
모든 TCP	TCP	0~65535	*database-sg-id*
HTTPS	TCP	443	*s3-prefix-list-id*

MongoDB 연결

AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 MongoDB와 MongoDB Atlas의 테이블에서 읽고 쓸 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 사용자 이름 및 암호 보안 인증 정보를 사용하여 MongoDB에 연결할 수 있습니다.

MongoDB 대한 자세한 내용은 MongoDB 설명서를 참조하십시오.

MongoDB 연결 구성

AWS Glue에서 MongoDB에 연결하려면 MongoDB 보안 인증 정보와 *mongodbUser*, *mongodbPass*가 필요합니다.

AWS Glue에서 MongoDB에 연결하려면 몇 가지 전제 조건이 필요할 수 있습니다.

- MongoDB 인스턴스가 Amazon VPC에 있는 경우, 퍼블릭 인터넷을 통과하는 트래픽 없이 AWS Glue 작업이 MongoDB 인스턴스와 통신할 수 있도록 Amazon VPC를 구성하십시오.

  Amazon VPC에서 AWS Glue가 작업을 실행하는 동안 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 MongoDB 인스턴스와 이 위치 간의 네트워크 트래픽을 허용하도록 Amazon

VPC를 구성해야 합니다. 네트워크 레이아웃에 따라 보안 그룹 규칙, 네트워크 ACL, NAT 게이트웨이 및 피어링 연결을 변경해야 할 수도 있습니다.

그런 다음 MongoDB와 함께 사용할 수 있도록 AWS Glue를 구성할 수 있습니다.

MongoDB 연결 구성 방법:

1. 대안으로 AWS Secrets Manager에서 MongoDB 보안 인증을 사용하여 보안 암호를 생성할 수 있습니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

   - 키/값 페어를 선택하면 값 *mongodbUser*이 포함된 키 username에 대한 페어를 생성합니다.

     키/값 페어를 선택하면 값 *mongodbPass*가 포함된 키 password에 대한 페어를 생성합니다.

2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때에는 MongoDB 또는 MongoDB Atlas를 선택합니다.
   - MongoDB URL 또는 MongoDB Atlas URL을 선택할 때에는 MongoDB 인스턴스의 호스트 이름을 제공하십시오.

     MongoDB URL은 mongodb://*mongoHost*:*mongoPort*/*mongoDBname* 형식으로 제공됩니다.

     MongoDB Atlas URL은 mongodb+srv://*mongoHost*:*mongoPort*/*mongoDBname* 형식으로 제공됩니다.

     연결을 위한 기본 데이터베이스를 제공하는 *mongoDBName*은 선택 사항입니다.
   - Secrets Manager 암호를 생성하기로 선택한 경우 AWS Secrets Manager 보안 인증 정보 유형을 선택합니다.

     그런 다음 AWS 암호에 *secretName*을 입력합니다.
   - 사용자 이름과 비밀번호를 제공하기로 선택한 경우 *mongodbUser*와 *mongodbPass*를 제공하십시오.

3. 다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.

   - Amazon VPC에서 AWS에 호스팅된 MongoDB 인스턴스의 경우

- MongoDB 보안 보안 인증 정보를 정의하는 AWS Glue 연결에 Amazon VPC 연결 정보를 제공해야 합니다. 연결을 만들거나 업데이트할 때 네트워크 옵션에서 VPC, 서브넷 및 보안 그룹을 설정합니다.

AWS Glue MongoDB 연결을 생성한 후에는 연결 방법을 호출하기 전에 다음 조치를 수행해야 합니다.

- Secrets Manager 암호을 생성하기로 선택한 경우, AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여하십시오.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

AWS Glue for Spark에서 AWS Glue MongoDB 연결을 사용하려면 연결 방법 호출에서 `connectionName` 옵션을 제공하십시오. 또는, the section called “MongoDB와 통합”의 단계에 따라 AWS Glue 데이터 카탈로그와 함께 연결을 사용할 수도 있습니다.

Glue AWS 연결을 사용하여 MongoDB에서 읽기

사전 조건:

- 읽으려는 MongoDB 컬렉션. 컬렉션에 대한 식별 정보가 필요합니다.

  MongoDB 컬렉션은 데이터베이스 이름 *mongodbName* 및 컬렉션 이름 *mongodbCollection*으로 식별됩니다.
- 인증 정보를 제공하도록 구성된 AWS Glue MongoDB 연결입니다. 인증 정보를 구성하려면 앞 절차인 MongoDB에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
mongodb_read = glueContext.create_dynamic_frame.from_options(
    connection_type="mongodb",
    connection_options={
        "connectionName": "connectionName",
        "database": "mongodbName",
        "collection": "mongodbCollection",
        "partitioner":
 "com.mongodb.spark.sql.connector.read.partitioner.SinglePartitionPartitioner",
        "partitionerOptions.partitionSizeMB": "10",
        "partitionerOptions.partitionKey": "_id",
        "disableUpdateUri": "false",
```

```
    }
)
```

MongoDB 테이블에 쓰기

이 예제에서는 기존 DynamicFrame, *dynamicFrame*의 정보를 MongoDB에 씁니다.

사전 조건:

- 쓰려는 MongoDB 컬렉션. 컬렉션에 대한 식별 정보가 필요합니다.

  MongoDB 컬렉션은 데이터베이스 이름 *mongodbName* 및 컬렉션 이름 *mongodbCollection*으로 식별됩니다.
- 인증 정보를 제공하도록 구성된 AWS Glue MongoDB 연결입니다. 인증 정보를 구성하려면 앞 절차인 MongoDB에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="mongodb",
    connection_options={
        "connectionName": "connectionName",
        "database": "mongodbName",
        "collection": "mongodbCollection",
        "disableUpdateUri": "false",
        "retryWrites": "false",
    },
)
```

MongoDB에서 테이블에 쓰고 쓰는 중

이 예제에서는 기존 DynamicFrame, *dynamicFrame*의 정보를 MongoDB에 씁니다.

사전 조건:

- 읽으려는 MongoDB 컬렉션. 컬렉션에 대한 식별 정보가 필요합니다.

  쓰려는 MongoDB 컬렉션. 컬렉션에 대한 식별 정보가 필요합니다.

MongoDB 컬렉션은 데이터베이스 이름 *mongodbName* 및 컬렉션 이름 *mongodbCollection*으로 식별됩니다.

- MongoDB 인증 정보, *mongodbUser*, *mongodbPassword*.

예:

Python

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

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'], args)

output_path = "s3://some_bucket/output/" + str(time.time()) + "/"
mongo_uri = "mongodb://<mongo-instanced-ip-address>:27017"
mongo_ssl_uri = "mongodb://<mongo-instanced-ip-address>:27017"
write_uri = "mongodb://<mongo-instanced-ip-address>:27017"

read_mongo_options = {
    "uri": mongo_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "username": "mongodbUsername",
    "password": "mongodbPassword",
    "partitioner": "MongoSamplePartitioner",
    "partitionerOptions.partitionSizeMB": "10",
    "partitionerOptions.partitionKey": "_id"}
```

```
ssl_mongo_options = {
    "uri": mongo_ssl_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "ssl": "true",
    "ssl.domain_match": "false"
}

write_mongo_options = {
    "uri": write_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "username": "mongodbUsername",
    "password": "mongodbPassword",
}

# Get DynamicFrame from MongoDB
dynamic_frame =
 glueContext.create_dynamic_frame.from_options(connection_type="mongodb",

 connection_options=read_mongo_options)

# Write DynamicFrame to MongoDB
glueContext.write_dynamic_frame.from_options(dynamicFrame,
 connection_type="mongodb", connection_options=write_mongo_options)

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  val DEFAULT_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  val WRITE_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
```

```
  lazy val defaultJsonOption = jsonOptions(DEFAULT_URI)
  lazy val writeJsonOption = jsonOptions(WRITE_URI)
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Get DynamicFrame from MongoDB
    val dynamicFrame: DynamicFrame = glueContext.getSource("mongodb",
 defaultJsonOption).getDynamicFrame()

    // Write DynamicFrame to MongoDB
    glueContext.getSink("mongodb", writeJsonOption).writeDynamicFrame(dynamicFrame)

    Job.commit()
  }

  private def jsonOptions(uri: String): JsonOptions = {
    new JsonOptions(
      s"""{"uri": "${uri}",
         |"database":"mongodbName",
         |"collection":"mongodbCollection",
         |"username": "mongodbUsername",
         |"password": "mongodbPassword",
         |"ssl":"true",
         |"ssl.domain_match":"false",
         |"partitioner": "MongoSamplePartitioner",
         |"partitionerOptions.partitionSizeMB": "10",
         |"partitionerOptions.partitionKey": "_id"}""".stripMargin)
  }
}
```

MongoDB 연결 옵션 참조

MongoDB에 대한 연결을 지정합니다. 연결 옵션은 소스 연결 및 싱크 연결에 따라 다릅니다.

다음과 같은 연결 속성은 소스 연결과 싱크 연결 간에 공유됩니다.

- `connectionName` - 읽기/쓰기에 사용됩니다. 연결 방법에 인증 및 네트워킹 정보를 제공하도록 구성된 AWS Glue MongoDB 연결의 이름입니다. the section called “MongoDB 구성”에서 설명

한 대로 AWS Glue 연결을 구성하면 connectionName을 제공했을 때"uri", "username" 및 "password" 연결 옵션을 제공할 필요가 없어집니다.

- "uri": (필수 사항) 읽을 소스 MongoDB 호스트로, mongodb://<host>:<port> 형식입니다. AWS Glue 4.0 이전의 AWS Glue 버전에서 사용됩니다.
- "connection.uri": (필수 사항) 읽을 소스 MongoDB 호스트로, mongodb://<host>:<port> 형식입니다. AWS Glue 4.0 이상 버전에서 사용됩니다.
- "username": (필수 사항) MongoDB 사용자 이름입니다.
- "password": (필수 사항) MongoDB 암호입니다.
- "database": (필수 사항) 읽을 MongoDB 데이터베이스입니다. 이 옵션은 작업 스크립트에서 glue_context.create_dynamic_frame_from_catalog를 호출할 때 additional_options로 전달할 수도 있습니다.
- "collection": (필수 사항) 읽을 MongoDB 컬렉션입니다. 이 옵션은 작업 스크립트에서 glue_context.create_dynamic_frame_from_catalog를 호출할 때 additional_options로 전달할 수도 있습니다.

소스로서의 "connectionType": "mongodb"

소스로서의 "connectionType": "mongodb"에는 다음 연결 옵션을 사용합니다.

- "ssl": (선택 사항) true이면 SSL 연결을 시작합니다. 기본값은 false입니다.
- "ssl.domain_match": (선택 사항) true 및 ssl이 true이면 도메인 일치 검사가 수행됩니다. 기본값은 true입니다.
- "batchSize": (선택 사항) 내부 배치의 커서 내에서 사용되는 배치당 반환할 문서 수입니다.
- "partitioner": (선택 사항) MongoDB에서 입력 데이터를 읽는 파티셔너의 클래스 이름입니다. 커넥터는 다음 파티셔너를 제공합니다.
  - MongoDefaultPartitioner(기본값)(AWS Glue 4.0에서는 지원되지 않음)
  - MongoSamplePartitioner(MongoDB 3.2 이상 필요)(AWS Glue 4.0에서는 지원되지 않음)
  - MongoShardedPartitioner(AWS Glue 4.0에서는 지원되지 않음)
  - MongoSplitVectorPartitioner(AWS Glue 4.0에서는 지원되지 않음)
  - MongoPaginateByCountPartitioner(AWS Glue 4.0에서는 지원되지 않음)
  - MongoPaginateBySizePartitioner(AWS Glue 4.0에서는 지원되지 않음)
  - com.mongodb.spark.sql.connector.read.partitioner.SinglePartitionPartitioner
  - com.mongodb.spark.sql.connector.read.partitioner.ShardedPartitioner

  - com.mongodb.spark.sql.connector.read.partitioner.PaginateIntoPartitionsPartit
- "partitionerOptions": (선택 사항) 지정된 파티셔너에 대한 옵션입니다. 각 파티셔너에 대해 다음 옵션이 지원됩니다.
  - MongoSamplePartitioner: partitionKey, partitionSizeMB, samplesPerPartition
  - MongoShardedPartitioner: shardkey
  - MongoSplitVectorPartitioner: partitionKey, partitionSizeMB
  - MongoPaginateByCountPartitioner: partitionKey, numberOfPartitions
  - MongoPaginateBySizePartitioner: partitionKey, partitionSizeMB

  이러한 옵션에 대한 자세한 내용은 MongoDB 설명서의 파티셔너 구성을 참조하십시오.

싱크로서의 "connectionType": "mongodb"

싱크로서의 "connectionType": "mongodb"에는 다음 연결 옵션을 사용합니다.

- "ssl": (선택 사항) true이면 SSL 연결을 시작합니다. 기본값은 false입니다.
- "ssl.domain_match": (선택 사항) true 및 ssl이 true이면 도메인 일치 검사가 수행됩니다. 기본값은 true입니다.
- "extendedBsonTypes": (선택 사항) true이면 데이터를 MongoDB에 쓸 때 확장 BSON 유형을 허용합니다. 기본값은 true입니다.
- "replaceDocument": (선택 사항) true이면 _id 필드가 포함된 데이터 세트를 저장할 때 전체 문서를 대체합니다. false이면 데이터 세트의 필드와 일치하는 문서의 필드만 업데이트됩니다. 기본값은 true입니다.
- "maxBatchSize": (선택 사항) 데이터를 저장할 때 대량 작업의 최대 배치 크기입니다. 기본값은 512입니다.
- "retryWrites": (선택 사항): AWS Glue에서 네트워크 오류가 발생하는 경우 특정 쓰기 작업을 한 번 자동으로 재시도합니다.

SAP HANA 커넥션

AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 SAP HANA의 테이블에서 읽고 쓸 수 있습니다. SQL 쿼리를 사용하여 SAP HANA에서 읽을 내용을 정의할 수 있습니다. AWS Glue SAP HANA 연결을 통해 AWS Secrets Manager에 저장된 JDBC 보안 인증 정보를 사용하여 SAP HANA에 연결합니다.

SAP HANA JDBC에 대한 자세한 내용은 SAP HANA 설명서를 참조하십시오.

SAP HANA 연결 구성

AWS Glue에서 SAP HANA에 연결하려면 AWS Secrets Manager 보안 암호에서 SAP HANA 보안 인증 정보를 생성하고 저장한 다음 해당 보안 암호를 SAP HANA AWS 연결에 연결해야 합니다. SAP HANA 서비스와 AWS Glue 간의 네트워크 연결을 구성해야 합니다.

SAP HANA에 연결하려면 몇 가지 사전 요구 사항이 필요할 수 있습니다.

- SAP HANA 서비스가 Amazon VPC에 있는 경우, 퍼블릭 인터넷을 통과하는 트래픽 없이 AWS Glue 작업이 SAP HANA 서비스와 통신할 수 있도록 Amazon VPC를 구성하십시오.

  Amazon VPC에서 AWS Glue가 작업을 실행하는 동안 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 Amazon VPC가 SAP HANA 엔드포인트와 이 위치 간의 네트워크 트래픽을 허용하도록 구성되어 있는지 확인해야 합니다. 작업을 수행하려면 SAP HANA JDBC 포트와의 TCP 연결을 설정해야 합니다. SAP HANA 포트에 대한 자세한 내용은 SAP HANA 설명서를 참조하십시오. 네트워크 레이아웃에 따라 보안 그룹 규칙, 네트워크 ACL, NAT 게이트웨이 및 피어링 연결을 변경해야 할 수도 있습니다.
- SAP HANA 엔드포인트가 인터넷에 액세스할 수 있는 경우 추가 전제 조건은 없습니다.

SAP HANA에 대한 연결 구성하는 방법:

1. AWS Secrets Manager에서 SAP HANA 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

   - 키/값 페어를 선택하면 값 *saphanaUsername*이 포함된 키 `user`에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *saphanaPassword*가 포함된 키 `password`에 대한 페어를 생성합니다.
2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 AWS Glue에서 이용하기 위해 연결 이름 *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때 SAP HANA를 선택합니다.
   - SAP HANA URL을 제공할 때는 인스턴스의 URL을 제공하십시오.

SAP HANA JDBC URL은 `jdbc:sap://saphanaHostname:saphanaPort/?databaseName=saphanaDBname,Paramete` 형식입니다

AWS Glue에는 다음과 같은 JDBC URL 매개변수가 필요합니다.

- `databaseName` - 연결할 SAP HANA의 기본 데이터베이스입니다.

- AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

AWS Glue SAP HANA 연결을 생성한 후에는 AWS Glue 작업을 실행하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

SAP HANA 테이블에서 읽는 중

사전 조건:

- 읽으려는 SAP HANA 테이블. 테이블에 대한 식별 정보가 필요합니다.

  SAP HANA 테이블 이름 및 스키마 이름을 *schemaName*.*tableName* 형식으로 테이블을 지정할 수 있습니다. 테이블이 기본 스키마 "public"에 있는 경우 스키마 이름과 "." 구분 기호는 필요하지 않습니다. 이것을 *tableIdentifier*라고 합니다. 데이터베이스는 `connectionName`에서 JDBC URL 매개변수로 제공된다는 점에 유의하십시오.
- 인증 정보를 제공하도록 구성된 AWS Glue SAP HANA 연결입니다. 인증 정보를 구성하려면 앞 절차인 SAP HANA에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
saphana_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier",
    }
)
```

SELECT SQL 쿼리에 반환되는 결과를 필터링하는 쿼리를 제공할 수도 있습니다. query을 구성해야 합니다.

예:

```
saphana_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

SAP HANA 테이블에 쓰기

이 예제에서는 기존 DynamicFrame, *dynamicFrame*의 정보를 SAP HANA에 기록합니다. 테이블에 이미 정보가 있는 경우 AWS Glue에서 오류가 발생합니다.

사전 조건:

- 쓰려고 하는 SAP HANA 테이블.

  SAP HANA 테이블 이름 및 스키마 이름을 *schemaName*.*tableName* 형식으로 테이블을 지정할 수 있습니다. 테이블이 기본 스키마 "public"에 있는 경우 스키마 이름과 "." 구분 기호는 필요하지 않습니다. 이것을 *tableIdentifier*라고 합니다. 데이터베이스는 connectionName에서 JDBC URL 매개변수로 제공된다는 점에 유의하십시오.
- SAP HANA 인증 정보. 인증 정보를 구성하려면 앞 절차인 SAP HANA에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
options = {
    "connectionName": "connectionName",
    "dbtable": 'tableIdentifier'
}

    saphana_write = glueContext.write_dynamic_frame.from_options(
        frame=dynamicFrame,
        connection_type="saphana",
        connection_options=options
)
```

SAP HANA 연결 옵션 참조

- `connectionName` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법에 인증 및 네트워킹 정보를 제공하도록 구성된 AWS Glue SAP HANA 연결의 이름입니다.
- `databaseName` - 읽기/쓰기에 사용됩니다. 유효한 값: SAP HANA의 데이터베이스 이름. 연결할 데이터베이스의 이름입니다.
- `dbtable` - 쓰기 시 필수, `query`가 제공되지 않는 한 읽기 전용. 읽기 및 쓰기에 사용됩니다. 유효한 값: SAP HANA SQL FROM 절의 내용입니다. 연결할 SAP HANA의 테이블을 식별합니다. 테이블 이름 이외의 다른 SQL(예: 하위 쿼리)을 제공할 수도 있습니다. 자세한 내용은 SAP HANA 설명서의 From 절을 참조하십시오.
- `query` — 읽기에 사용됩니다. SAP HANA에서 읽을 때 검색해야 하는 내용을 정의하는 SAP HANA SQL SELECT 쿼리입니다.

Snowflake 연결

AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 Snowflake의 테이블에서 읽고 쓸 수 있습니다. SQL 쿼리를 사용하여 Snowflake에서 읽을 수 있습니다. 사용자와 암호를 사용하여 Snowflake에 연결할 수 있습니다. AWS Glue 데이터 카탈로그를 통해 AWS Secrets Manager에 저장된 Snowflake 보안 인증을 참조할 수 있습니다. AWS Glue for Spark용 데이터 카탈로그 Snowflake 보안 인증은 크롤러용 데이터 카탈로그 Snowflake 보안 인증과 별도로 저장됩니다. Snowflake에 연결하도록 구성된 `JDBC` 유형 연결이 아니라 `SNOWFLAKE` 유형 연결을 선택해야 합니다.

Snowball에 대한 자세한 내용은 Snowflake 웹 사이트를 참조하세요. AWS에서 Snowflake에 대한 자세한 내용은 Snowflake Data Warehouse on Amazon Web Services를 참조하세요.

Snowflake 연결 구성

인터넷을 통해 사용할 수 있는 Snowflake 데이터베이스에 연결하기 위한 AWS의 필수 조건은 없습니다.

선택적으로 다음 구성을 수행하여 AWS Glue로 연결 보안 인증을 관리할 수 있습니다.

AWS Glue로 연결 보안 인증을 관리하려면

1. Snowflake에서 사용자, *`snowflakeUser`* 및 암호, *`snowflakePassword`*를 생성합니다.
2. AWS Secrets Manager에서 Snowflake 보안 인증을 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *`secretName`*을 유지합니다.

- 키/값 페어를 선택하면 키가 USERNAME인 *snowflakeUser*에 대한 페어를 생성합니다.
- 키/값 페어를 선택하면 키가 PASSWORD인 *snowflakePassword*에 대한 페어를 생성합니다.
- 키/값 페어를 선택하면 키가 sfWarehouse인 Snowflake 웨어하우스를 제공할 수 있습니다.

3. AWS Glue Data Catalog에서 연결과 연결 생성을 차례로 선택하여 연결을 생성합니다. 다음과 같이 연결 마법사의 단계에 따라 프로세스를 완료합니다.

   - 데이터 소스를 선택할 때 Snowflake를 선택하고 다음을 선택합니다.
   - 호스트, 포트와 같은 연결 세부정보를 입력합니다. 호스트 Snowflake URL을 입력을 때 Snowflake 인스턴스의 URL을 제공합니다. URL은 일반적으로 *account_identifier*.snowflakecomputing.com 양식의 호스트 이름을 사용합니다. 그러나 Snowflake 계정 유형(예: AWS, Azure 또는 Snowflake 호스팅)에 따라 URL 형식이 다를 수 있습니다.
   - IAM 서비스 역할을 선택할 때에는 드롭다운 메뉴에서 선택합니다. 이것은 VPC가 지정된 경우 AWS Secrets Manager에 액세스하고 IP를 할당하는 데 사용되는 계정의 IAM 역할입니다.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.

4. 마법사의 다음 단계에서 Snowflake 연결에 대한 속성을 설정합니다.
5. 마법사의 마지막 단계에서 설정을 검토한 후 프로세스를 완료하여 연결을 생성합니다.

다음과 같은 상황에서 다음이 필요할 수 있습니다.

- Amazon VPC에서 AWS에 호스팅된 Snowflake의 경우
  - Snowflake에 적합한 Amazon VPC 구성이 필요합니다. Amazon VPC를 구성하는 방법에 대한 자세한 내용은 Snowflake 설명서의 AWS PrivateLink & Snowflake를 참조하세요.
  - AWS Glue에 적합한 Amazon VPC 구성이 필요합니다. the section called “AWS Glue에 대한 인터페이스 VPC 엔드포인트(AWS PrivateLink) 구성 ”.
  - Snowflake 보안 인증을 정의하는 AWS Secrets Manager 보안 암호의 ID 외에도 Amazon VPC 연결 정보를 제공하는 AWS Glue Data Catalog 연결을 생성해야 합니다. 이전 항목에 링크된 Snowflake 설명서에 설명된 대로 AWS PrivateLink 사용 시 URL이 변경됩니다.
  - 데이터 카탈로그 연결을 추가 네트워크 연결로 포함하는 작업 구성이 필요합니다.

Snowflake 테이블에서 읽기

사전 조건: 읽으려는 Snowflake 테이블. Snowflake 테이블 이름, *tableName*이 필요합니다. Snowflake URL *snowflakeUrl*, 사용자 이름 *snowflakeUser* 및 암호 *snowflakePassword*가 필

요합니다. Snowflake 사용자에게 기본 네임스페이스가 설정되어 있지 않은 경우 Snowflake 데이터베이스 이름, *databaseName* 및 스키마 이름, *schemaName*이 필요합니다. 또한 Snowflake 사용자에게 기본 웨어하우스가 설정되어 있지 않은 경우 웨어하우스 이름, *warehouseName*이 필요합니다.

예시:

추가 사전 조건: AWS Glue로 연결을 관리하려면 단계를 수행하여 *snowflakeUrl*, *snowflakeUsername* 및 *snowflakePassword*를 구성합니다. 이 단계를 검토하려면 이전 the section called “Snowflake 구성” 섹션을 참조하세요. 연결할 추가 네트워크 연결을 선택하기 위해 connectionName 파라미터를 사용합니다.

```
snowflake_read = glueContext.create_dynamic_frame.from_options(
  connection_type="snowflake",
  connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    }
)
```

또한 autopushdown 및 query 파라미터를 사용하여 Snowflake 테이블의 일부를 읽을 수 있습니다. 이 방법은 결과를 Spark에 로드한 후 필터링하는 것보다 훨씬 더 효율적일 수 있습니다. 모든 판매가 동일한 테이블에 저장되지만 연휴에 특정 매장의 매출만 분석하려는 예제를 살펴봅니다. 해당 정보가 테이블에 저장된 경우 다음과 같이 조건자 푸시다운을 사용하여 결과를 가져올 수 있습니다.

```
snowflake_node = glueContext.create_dynamic_frame.from_options(
    connection_type="snowflake",
    connection_options={
        "autopushdown": "on",
        "query": "select * from sales where store='1' and IsHoliday='TRUE'",
        "connectionName": "snowflake-glue-conn",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    }
)
```

Snowflake 테이블에 쓰기

사전 조건: 쓰려는 Snowflake 데이터베이스. 현재 또는 원하는 테이블 이름, *TableName*이 필요합니다. Snowflake URL *snowflakeUrl*, 사용자 이름 *snowflakeUser* 및 암호 *snowflakePassword*가 필요합니다. Snowflake 사용자에게 기본 네임스페이스가 설정되어 있지 않은 경우 Snowflake 데이터베이스 이름, *databaseName* 및 스키마 이름, *schemaName*이 필요합니다. 또한 Snowflake 사용자에게 기본 웨어하우스가 설정되어 있지 않은 경우 웨어하우스 이름, *warehouseName*이 필요합니다.

예시:

추가 사전 조건: AWS Glue로 연결을 관리하려면 단계를 수행하여 *snowflakeUrl*, *snowflakeUsername* 및 *snowflakePassword*를 구성합니다. 이 단계를 검토하려면 이전 the section called “Snowflake 구성” 섹션을 참조하세요. 연결할 추가 네트워크 연결을 선택하기 위해 connectionName 파라미터를 사용합니다.

```
glueContext.write_dynamic_frame.from_options(
    connection_type="snowflake",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    },
)
```

Snowflake 연결 옵션 참조

Snowflake 연결 유형에서는 다음과 같은 연결 옵션을 사용합니다.

이 섹션의 일부 파라미터는 데이터 카탈로그 연결(sfUrl, sfUser, sfPassword)에서 검색할 수 있으며, 이 경우 사용자가 해당 파라미터를 제공하지 않아도 됩니다. connectionName 파라미터를 제공하여 이 작업을 수행할 수 있습니다.

이 섹션의 일부 파라미터는 AWS Secrets Manager 보안 암호(sfUser, sfPassword)에서 검색할 수 있으며, 이 경우 사용자가 해당 파라미터를 제공하지 않아도 됩니다. 보안 암호는 sfUser 및 sfPassword 키 아래에서 해당 콘텐츠를 제공해야 합니다. secretId 파라미터를 제공하여 이 작업을 수행할 수 있습니다.

다음 파라미터는 보통 Snowflake에 연결할 때 사용됩니다.

- `sfDatabase` - Snowflake에 사용자 기본값이 설정되어 있지 않은 경우 필요합니다. 읽기 및 쓰기에 사용됩니다. 연결 후 세션에 사용할 데이터베이스입니다.
- `sfSchema` - Snowflake에 사용자 기본값이 설정되어 있지 않은 경우 필요합니다. 읽기 및 쓰기에 사용됩니다. 연결 후 세션에 사용할 스키마입니다.
- `sfWarehouse` - Snowflake에 사용자 기본값이 설정되어 있지 않은 경우 필요합니다. 읽기 및 쓰기에 사용됩니다. 연결 후 세션에 사용할 기본 가상 웨어하우스입니다.
- `sfRole` - Snowflake에 사용자 기본값이 설정되어 있지 않은 경우 필요합니다. 읽기 및 쓰기에 사용됩니다. 연결 후 세션에 사용할 기본 보안 암호 역할입니다.
- `sfUrl` -- (필수) 읽기 및 쓰기에 사용됩니다. 다음 *`account_identifier`*`.snowflakecomputing.com` 형식으로 계정의 호스트 이름을 지정합니다. 계정 식별자에 대한 자세한 내용은 Snowflake 설명서의 Account Identifiers를 참조하세요.
- `sfUser` -- (필수) 읽기 및 쓰기에 사용됩니다. Snowflake 사용자의 로그인 이름입니다.
- `sfPassword` - (`pem_private_key`가 제공되지 않은 경우 필수) 읽기/쓰기에 사용됩니다. Snowflake 사용자의 암호입니다.
- `dbtable` - 전체 테이블에 대한 작업을 수행할 때 필요합니다. 읽기 및 쓰기에 사용됩니다. 읽을 테이블 또는 데이터를 쓸 테이블의 이름입니다. 읽을 때 모든 열과 레코드가 검색됩니다.
- `pem_private_key` - 읽기/쓰기에 사용됩니다. 암호화되지 않은 b64 인코딩된 프라이빗 키 문자열입니다. Snowflake 사용자의 프라이빗 키입니다. 일반적으로 이를 PEM 파일에서 복사합니다. 자세한 내용은 Snowflake 설명서의 키 페어 인증 & 키 페어 순환을 참조하세요.
- `query` - 쿼리로 읽을 때 필요합니다. 읽기에 사용됩니다. 실행할 정확한 쿼리(`SELECT` 문)입니다.

다음 옵션은 Snowflake에 연결하는 과정에서 특정 동작을 구성하는 데 사용됩니다.

- `preactions` - 읽기/쓰기에 사용됩니다. 유효한 값: 세미콜론으로 구분된 SQL 문 목록(문자열). SQL 문은 AWS Glue와 Snowflake 사이에서 데이터가 전송되기 전에 실행됩니다. 명령문에 `%s`가 포함된 경우 `%s`는 작업에서 참조하는 테이블 이름으로 바뀝니다.
- `postactions` - 읽기/쓰기에 사용됩니다. SQL 문은 AWS Glue와 Snowflake 사이에서 데이터가 전송된 후에 실행됩니다. 명령문에 `%s`가 포함된 경우 `%s`는 작업에서 참조하는 테이블 이름으로 바뀝니다.
- `autopushdown` - 기본값: `"on"`. 유효한 값: `"on"`, `"off"`. 이 파라미터는 자동 쿼리 푸시다운의 활성화 여부를 제어합니다. 푸시다운이 활성화된 경우 Spark에서 쿼리를 실행할 때 쿼리의 일부를 Snowflake 서버로 '푸시다운'할 수 있으면 해당 쿼리가 푸시다운됩니다. 이렇게 하면 일부 쿼리의 성능이 향상됩니다. 쿼리의 푸시다운 여부에 대한 자세한 내용은 Snowflake 설명서의 Pushdown을 참조하세요.

또한 Snowflake Spark 커넥터에서 사용할 수 있는 일부 옵션이 AWS Glue에서 지원될 수도 있습니다. Snowflake Spark 커넥터에서 사용할 수 있는 옵션에 대한 자세한 내용은 Snowflake 설명서의 Setting Configuration Options for the Connector를 참조하세요.

Snowflake 커넥터 제한

AWS Glue for Spark를 사용하여 Snowflake에 연결하는 경우 다음과 같은 제한 사항이 적용됩니다.

- 이 커넥터는 작업 북마크를 지원하지 않습니다. 작업 북마크에 대한 자세한 내용을 알아보려면 the section called "처리된 데이터를 작업 북마크로 추적"을(를) 참조하세요.
- 이 커넥터는 `create_dynamic_frame.from_catalog` 및 `write_dynamic_frame.from_catalog` 메서드를 사용하여 AWS Glue 데이터 카탈로그의 테이블을 통한 Snowflake 읽기 및 쓰기를 지원하지 않습니다.
- 이 커넥터는 사용자 및 암호 이외의 보안 인증으로 Snowflake에 연결하는 것을 지원하지 않습니다.
- 이 커넥터는 스트리밍 작업 내에서 지원되지 않습니다.
- 이 커넥터는 정보를 검색할 때(예: `query` 파라미터 사용) `SELECT` 문 기반 쿼리를 지원합니다. 다른 종류의 쿼리(예: `SHOW`, `DESC` 또는 DML 문)는 지원되지 않습니다.
- Snowflake는 Snowflake 클라이언트를 통해 제출되는 쿼리 텍스트(즉, SQL 문)의 크기를 명령문당 1MB로 제한합니다. 자세한 내용은 Limits on Query Text Size를 참조하세요.

Teradata Vantage 연결

AWS Glue for Spark를 이용하여 AWS Glue 4.0 이상 버전에서 Teradata Vantage의 테이블에서 읽고 쓸 수 있습니다. SQL 쿼리를 사용하여 Teradata에서 읽을 내용을 정의할 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 사용자 이름 및 암호 보안 인증 정보를 사용하여 Teradata에 연결할 수 있습니다.

Teradata에 대한 자세한 내용은 Teradata 설명서를 참조하십시오.

Teradata 연결 구성

AWS Glue에서 Teradata에 연결하려면 Teradata 보안 인증 정보를 생성하여 AWS Secrets Manager 암호에 저장한 다음 해당 암호를 AWS Glue Teradata 연결에 연결해야 합니다. Teradata 인스턴스가 Amazon VPC에 있는 경우 AWS Glue Teradata 연결에 네트워킹 옵션도 제공해야 합니다.

AWS Glue에서 Teradata에 연결하려면 몇 가지 전제 조건이 필요할 수 있습니다.

- Amazon VPC를 통해 Teradata 환경에 액세스하는 경우, AWS Glue 작업이 Teradata 환경과 통신할 수 있도록 Amazon VPC를 구성하십시오. 퍼블릭 인터넷을 통해 Teradata 환경에 액세스하는 것은 권장하지 않습니다.

  Amazon VPC에서 AWS Glue가 작업을 실행하는 동안 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 Amazon VPC가 Teradata 인스턴스와 이 위치 간의 네트워크 트래픽을 허용하도록 구성되어 있는지 확인해야 합니다. 작업을 수행하려면 Teradata 클라이언트 포트와 TCP 연결을 설정해야 합니다. Teradata 포트에 대한 자세한 내용은 Teradata 설명서를 참조하십시오.

  네트워크 레이아웃에 따라 보안 VPC 연결에는 Amazon VPC 및 기타 네트워킹 서비스를 변경해야 할 수 있습니다. AWS 연결에 대한 자세한 내용은 Teradata 설명서의 AWS연결 옵션을 참조하십시오.

AWS Glue Teradata 연결을 구성하는 방법:

1. Teradata 구성에서 AWS Glue가 *teradataUser* 및 *teradataPassword*와 연결할 사용자 및 암호를 식별하거나 생성합니다. 자세한 내용은 Teradata 설명서의 Vantage 보안 개요를 참조하십시오.
2. AWS Secrets Manager에서 Teradata 보안 인증 정보를 사용하여 보안 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
   - 키/값 페어를 선택하면 값 *teradataUsername*이 포함된 키 `user`에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *teradataPassword*가 포함된 키 `password`에 대한 페어를 생성합니다.
3. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때 Teradata를 선택합니다.
   - JDBC URL을 제공할 때는 인스턴스의 URL을 제공하십시오. 또한 JDBC URL에 쉼표로 구분된 특정 연결 매개변수를 하드코딩할 수 있습니다. URL의 형식: `jdbc:teradata://`*teradataHostname*`/`*ParameterName*`=`*ParameterValue*`,`*ParameterName*

     지원되는 URL 파라미터는 다음과 같습니다.
     - `DATABASE` - 기본으로 액세스하는 호스트의 데이터베이스 이름입니다.
     - `DBS_PORT` - 비표준 포트에서 실행할 때 사용되는 데이터베이스 포트입니다.

- 보안 인증 정보 유형을 선택할 때에는 AWS Secrets Manager을 선택한 다음 AWS보안 암호를 *secretName*으로 설정합니다.

4. 다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.

   - Amazon VPC에서 AWS에 호스팅된 Teradata 인스턴스의 경우
     - Teradata 보안 보안 인증 정보를 정의하는 AWS Glue 연결에 Amazon VPC 연결 정보를 제공해야 합니다. 연결을 만들거나 업데이트할 때 네트워크 옵션에서 VPC, 서브넷 및 보안 그룹을 설정합니다.

AWS Glue Teradata 연결을 생성한 후에는 연결 방법을 호출하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

Teradata에서 읽는 중

사전 조건:

- 읽으려는 Teradata 테이블. 테이블 이름, *tableName*이 필요합니다.
- 인증 정보를 제공하도록 구성된 AWS Glue Teradata 연결입니다. 인증 정보를 구성하려면 Teradata에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예:

```
teradata_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

SELECT SQL 쿼리에 반환되는 결과를 필터링하는 쿼리를 제공할 수도 있습니다. query을 구성해야 합니다.

예:

```
teradata_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Teradata 테이블에 쓰기

전제 조건: 쓰고 싶은 Teradata 테이블, *tableName*. 연결 방법을 호출하기 전에 테이블을 생성해야 합니다.

예:

```
teradata_write = glueContext.write_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

Teradata 연결 옵션 참조

- `connectionName` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법에 인증 및 네트워킹 정보를 제공하도록 구성된 AWS Glue Teradata 연결의 이름입니다.
- `dbtable` - 쓰기 시 필수, `query`가 제공되지 않는 한 읽기 전용. 읽기 및 쓰기에 사용됩니다. 연결 방법이 상호 작용할 테이블의 이름.
- `query` — 읽기에 사용됩니다. Teradata에서 읽을 때 검색해야 하는 내용을 정의하는 SELECT SQL 쿼리입니다.

Teradata Vantage NOS 연결

Teradata NOS(네이티브 객체 스토어) 연결은 Teradata WRITE_NOS 쿼리를 활용하여 기존 테이블에서 읽고 READ_NOS 쿼리를 활용하여 테이블에 쓰는 Teradata Vantage에 대한 새로운 연결입니다. 이 쿼리는 Amazon S3을 스테이징 디렉터리로 사용하므로 특히 대량의 데이터를 처리할 때 Teradata NOS 커넥터가 기존 Teradata 커넥터(JDBC 기반)보다 빠릅니다.

AWS Glue for Spark에서 Teradata NOS 연결을 이용하여 AWS Glue 5.0 이상 버전에서 Teradata Vantage의 테이블에서 읽고 쓸 수 있습니다. SQL 쿼리를 사용하여 Teradata에서 읽을 내용을 정의할 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 사용자 이름 및 암호 자격 증명을 사용하여 Teradata에 연결할 수 있습니다.

Teradata에 대한 자세한 내용은 Teradata 설명서를 참조하세요.

주제

- Teradata NOS 연결 생성
- Teradata 테이블에서 읽기
- Teradata 테이블에 쓰기
- Teradata 연결 옵션 참조
- AWS Glue 시각적 ETL UI에서 옵션 제공

Teradata NOS 연결 생성

AWS Glue에서 Teradata NOS에 연결하려면 Teradata 자격 증명을 생성하여 AWS Secrets Manager 암호에 저장한 다음 해당 암호를 AWS Glue Teradata NOS 연결에 연결해야 합니다. Teradata 인스턴스가 Amazon VPC에 있는 경우 AWS Glue Teradata NOS 연결에 네트워킹 옵션도 제공해야 합니다.

사전 조건:

- Amazon VPC를 통해 Teradata 환경에 액세스하는 경우, AWS Glue 작업이 Teradata 환경과 통신할 수 있도록 Amazon VPC를 구성하십시오. 퍼블릭 인터넷을 통해 Teradata 환경에 액세스하는 것은 권장하지 않습니다.
- Amazon VPC에서 작업을 실행하는 동안 AWS Glue가 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 Amazon VPC가 Teradata 인스턴스와 이 위치 간의 네트워크 트래픽을 허용하도록 구성되어 있는지 확인해야 합니다. 작업을 수행하려면 Teradata 클라이언트 포트와 TCP 연결을 설정해야 합니다. Teradata 포트에 대한 자세한 내용은 Security Groups for Teradata Vantage을 참조하세요.
- 네트워크 레이아웃에 따라 보안 VPC 연결에는 Amazon VPC 및 기타 네트워킹 서비스를 변경해야 할 수 있습니다. AWS 연결에 대한 자세한 내용은 Teradata 설명서의 AWS 연결 옵션을 참조하세요.

AWS Glue Teradata NOS 연결을 구성하는 방법:

1. Teradata 구성에서 AWS Glue가 연결할 *teradataUsername* 및 *teradataPassword*를 식별하거나 생성합니다. 자세한 내용은 Teradata 설명서의 Vantage 보안 개요를 참조하세요.

2. AWS Secrets Manager에서 Teradata 보안 인증 정보를 사용하여 보안 암호를 생성합니다. AWS Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 AWS Secrets Manager 보안 암호 생성에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.

   - 키/값 페어를 선택하면 값 *teradataUsername*이 포함된 키 사용자 이름에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *teradataPassword*가 포함된 키 암호에 대한 페어를 생성합니다.

3. AWS Glue 콘솔에서 AWS Glue 연결 추가의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.

   - 연결 유형을 선택할 때 Teradata Vantage NOS를 선택합니다.
   - JDBC URL을 제공할 때는 인스턴스의 URL을 제공하십시오. 또한 JDBC URL에 쉼표로 구분된 특정 연결 매개변수를 하드코딩할 수 있습니다. URL은 다음 형식을 준수해야 합니다. `jdbc:teradata://teradataHostname/ParameterName=ParameterValue,ParameterName=ParameterValue`
   - 지원되는 URL 파라미터는 다음과 같습니다.
     - `DATABASE` - 기본으로 액세스하는 호스트의 데이터베이스 이름입니다.
     - `DBS_PORT` - 비표준 포트에서 실행할 때 사용되는 데이터베이스 포트입니다.
   - 자격 증명 유형을 선택할 때에는 AWS Secrets Manager을 선택한 다음 AWS 보안 암호를 *secretName*으로 설정합니다.

4. 다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.

   - Amazon VPC의 AWS에 호스팅된 Teradata 인스턴스의 경우 Teradata 보안 자격 증명을 정의하는 AWS Glue 연결에 Amazon VPC 연결 정보를 제공해야 합니다. 연결을 만들거나 업데이트할 때 네트워크 옵션에서 VPC, 서브넷, 보안 그룹을 설정합니다.

AWS Glue Teradata Vantage NOS 연결을 생성한 후에는 연결 방법을 호출하기 전에 다음 단계를 수행해야 합니다.

1. AWS Glue 작업 권한과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
2. AWS Glue 작업 구성에서 연결의 추가 네트워크 연결로 *connectionName*을 제공합니다.

Teradata 테이블에서 읽기

사전 조건:

- 읽으려는 Teradata 테이블. 테이블 이름, *tableName*이 필요합니다.
- Teradata 환경에는 staging_fs_url 옵션인 *stagingFsUrl*에서 지정한 Amazon S3 경로에 대한 쓰기 액세스 권한이 있습니다.
- AWS Glue 작업과 연결된 IAM 역할은 staging_fs_url 옵션에서 지정한 Amazon S3 위치에 대한 쓰기 액세스 권한이 있습니다.
- 인증 정보를 제공하도록 구성된 AWS Glue Teradata NOS 연결입니다. 인증 정보를 구성하는 AWS Glue Teradata NOS 연결을 구성하는 방법: 단계를 완료합니다. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

예시:

```
teradata_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type= "teradatanos",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "staging_fs_url": "stagingFsUrl"
    }
)
```

SELECT SQL 쿼리에 반환되는 결과를 필터링하는 쿼리를 제공할 수도 있습니다. 쿼리를 구성해야 합니다. dbTable과 쿼리를 모두 구성하면 커넥터가 데이터를 읽지 못합니다. 예시:

```
teradata_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="teradatanos",
    connection_options={
        "connectionName": "connectionName",
        "query": "query",
        "staging_fs_url": "stagingFsUrl"
    }
)
```

또한 Spark DataFrame API를 사용하여 Teradata 테이블에서 읽을 수 있습니다. 예시:

```
options = {
```

```
    "url": "JDBC_URL",
    "dbtable": "tableName",
    "user": "teradataUsername", # or use "username" as key here
    "password": "teradataPassword",
    "staging_fs_url": "stagingFsUrl"
}
teradata_read_table = spark.read.format("teradatanos").option(**options).load()
```

Teradata 테이블에 쓰기

사전 조건

- 쓰고 싶은 Teradata 테이블인 *tableName*입니다.
- Teradata 환경에는 staging_fs_url 옵션인 *stagingFsUrl*에 의해 지정된 Amazon S3 위치에 대한 읽기 액세스 권한이 있습니다.
- AWS Glue 작업과 연결된 IAM 역할은 staging_fs_url 옵션에서 지정한 Amazon S3 위치에 대한 쓰기 액세스 권한이 있습니다.
- 인증 정보를 제공하도록 구성된 AWS Glue Teradata 연결입니다. AWS Glue Teradata NOS 연결을 구성하는 방법:의 단계를 완료하여 인증 정보를 구성합니다. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.

  예시:

```
teradata_write = glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type= "teradatanos",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "staging_fs_url": "stagingFsUrl"
    }
)
```

Teradata 연결 옵션 참조

연결 및 작업 옵션:

- connectionName — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법에 인증 및 네트워킹 정보를 제공하도록 구성된 AWS Glue Teradata 연결의 이름입니다.

- `staging_fs_url` — 필수입니다. 읽기 및 쓰기에 사용됩니다. Amazon S3의 쓰기 가능한 위치로, Teradata에서 읽을 때 언로드된 데이터에 사용되며 Teradata에 쓸 때 Parquet 데이터를 Redshift에 로드하는 데 사용됩니다. S3 버킷이 AWS Glue 작업과 동일한 리전에 있어야 합니다.
- `dbtable` - 쓰기 시 필수, `query`가 제공되지 않는 한 읽기 전용. 읽기 및 쓰기에 사용됩니다. 연결 방법이 상호 작용할 테이블의 이름.
- `query` — 읽기에 사용됩니다. Teradata에서 읽을 때 검색해야 하는 내용을 정의하는 SELECT SQL 쿼리입니다. `dbtable` 옵션이 제공된 경우 전달할 수 없습니다.
- `clean_staging_s3_dir`-선택 사항입니다. 읽기 및 쓰기에 사용됩니다. true인 경우 읽기 또는 쓰기 후 스테이징 Amazon S3 객체를 정리합니다. 기본값은 true입니다.
- `pre_actions`-선택 사항입니다. 쓰기에 사용됩니다. Spark와 Teradata Vantage 간에 데이터가 전송되기 전에 실행되는 세미콜론으로 구분된 SQL 명령 목록입니다.
- `post_actions`-선택 사항입니다. 쓰기에 사용됩니다. Spark와 Teradata Vantage 간에 데이터가 전송된 후에 실행되는 세미콜론으로 구분된 SQL 명령 목록입니다.
- `truncate`-선택 사항입니다. 쓰기에 사용됩니다. true인 경우 커넥터는 덮어쓰기 모드로 쓸 때 테이블을 잘라냅니다. false인 경우 커넥터는 덮어쓰기 모드로 쓸 때 테이블을 삭제합니다. 기본값은 false입니다.
- `create_table_script`-선택 사항입니다. 쓰기에 사용됩니다. Teradata Vantage에 쓸 때 테이블을 생성하는 SQL 문입니다. 사용자 지정 메타데이터(예: CREATE MULTISET 또는 SET 테이블 또는 기본 인덱스 변경)가 있는 테이블을 생성하려는 경우에 유용합니다. 테이블 스크립트 생성에 사용되는 테이블 이름은 `dbtable` 옵션에 지정된 테이블 이름과 일치해야 합니다.
- `partition_size_in_mb`-선택 사항입니다. 읽기에 사용됩니다. 스테이징 Amazon S3 객체를 읽는 동안 Spark 파티션의 최대 크기(MB)입니다. 기본값은 128입니다.

Teradata 노드를 생성할 때 고급 옵션을 제공할 수 있습니다. 이 옵션은 Spark 스크립트에 대한 AWS Glue를 프로그래밍할 때 사용할 수 있는 옵션과 동일합니다.

the section called “Teradata Vantage 연결”를 참조하세요.

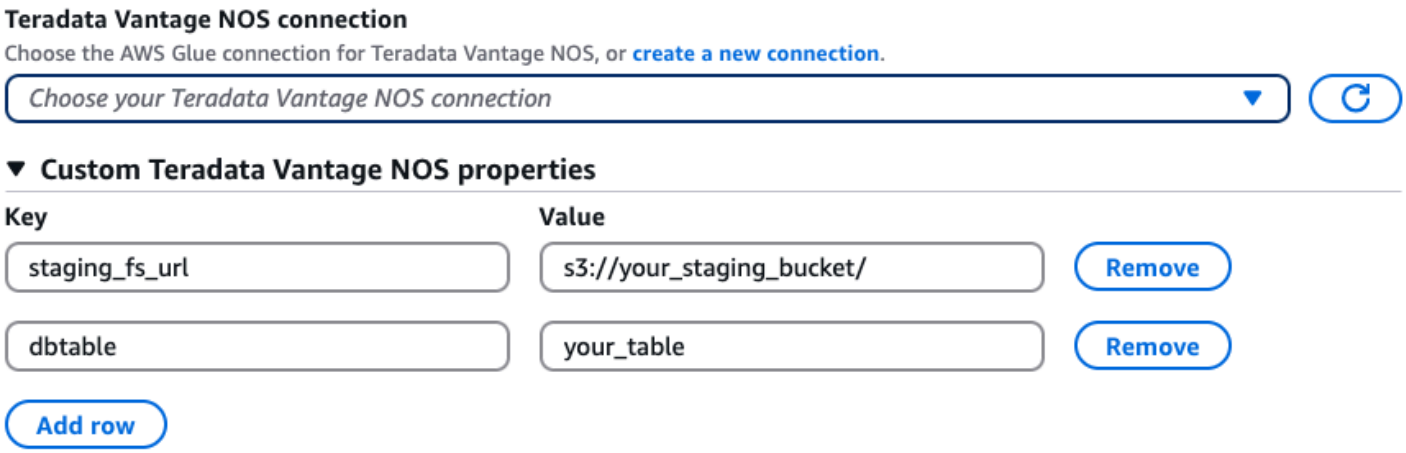
권한 부여 옵션:

다음은 커넥터가 스테이징 Amazon S3 버킷에 액세스하는 데 사용하는 AWS 계정 자격 증명을 제공하는 데 사용되는 옵션입니다. (1) 권한 부여 옵션을 전혀 제공하지 않고 AWS Glue 실행 역할에서 생성된 임시 자격 증명을 사용하는 옵션, (2) 생성한 권한 부여 객체 `auth_object`를 제공하는 옵션, (3) 장기 자격 증명을 사용하는 `aws_access_key_id and aws_secret_access_key`를 제공하거나, 임시 자격 증명을 사용하는 경우 `aws_access_key`, `aws_secret_access_key`, `aws_session_token`을 제공하는 옵션 중에서 선택할 수 있습니다.

- auth_object – 선택 사항입니다. 스테이징 Amazon S3 버킷에 액세스하는 데 사용됩니다. Teradata 인스턴스에서 생성된 권한 부여 객체 문자열입니다. 제공된 경우 커넥터는 이 권한 부여 객체를 사용하여 스테이징 Amazon S3 버킷에 액세스합니다. 제공되지 않은 경우 aws_access_key_id 및 aws_secret_access_key도 제공되지 않았으면 커넥터가 AWS Glue 실행 역할에서 임시 자격 증명이 검색하여 사용합니다. 이 권한 부여 객체와 연결된 AWS 계정은 AWS Glue 작업 및 스테이징 Amazon S3 버킷과 동일한 리전에 있거나 교차 계정 신뢰로 구성되어야 합니다.
- aws_access_key_id – 선택 사항입니다. 스테이징 Amazon S3 버킷에 액세스하는 데 사용됩니다. AWS 계정 보안 자격 증명의 일부입니다. auth_object가 제공되지 않고 aws_access_key_id가 aws_secret_access_key와 함께 제공되는 경우 커넥터는 이들을 사용하여 스테이징 Amazon S3 버킷에 액세스합니다. 이 액세스 키와 연결된 AWS 계정은 AWS Glue 작업 및 스테이징 Amazon S3 버킷과 동일한 리전에 있거나 교차 계정 신뢰로 구성되어야 합니다.
- aws_secret_access_key – 선택 사항입니다. 스테이징 Amazon S3 버킷에 액세스하는 데 사용됩니다. AWS 계정 보안 자격 증명의 일부입니다. auth_object가 제공되지 않고 aws_secret_access_key가 aws_access_key_id와 함께 제공되는 경우 커넥터는 이들을 사용하여 스테이징 Amazon S3 버킷에 액세스합니다. 이 비밀 키와 연결된 AWS 계정은 AWS Glue 작업 및 스테이징 Amazon S3 버킷과 동일한 리전에 있거나 교차 계정 신뢰로 구성되어야 합니다.
- aws_session_token – 선택 사항입니다. 스테이징 Amazon S3 버킷에 액세스하는 데 사용됩니다. 임시 AWS 계정 보안 자격 증명의 일부입니다. aws_access_key_id 및 aws_secret_access_key와 함께 제공되어야 합니다.

AWS Glue 시각적 ETL UI에서 옵션 제공

시각적 ETL 작업 UI에서 위의 옵션을 모두 제공할 수 있습니다. connectionName 옵션의 경우 Teradata Vantage NOS 연결 드롭다운 목록에서 선택해야 합니다. 다른 모든 옵션의 경우 사용자 지정 Teradata Vantage NOS 속성을 통해 키/값 페어로 제공해야 합니다.



수직 연결

AWS Glue for Spark를 사용하여 AWS Glue 4.0 이상 버전에서 Vertica의 테이블에서 읽고 쓸 수 있습니다. SQL 쿼리를 사용하여 Vertica에서 읽을 내용을 정의할 수 있습니다. AWS Glue 연결을 통해 AWS Secrets Manager에 저장된 사용자 이름 및 암호 보안 인증 정보를 사용하여 Vertica에 연결할 수 있습니다.

Vertica에 대한 자세한 내용은 Vertica 설명서를 참조하십시오.

Vertica 연결 구성

AWS Glue에서 Vertica에 연결하려면 AWS Secrets Manager 보안 암호에서 Vertica 보안 인증 정보를 생성하고 저장한 다음 해당 보안 암호를 Vertica AWS Glue 연결에 연결해야 합니다. Vertica 인스턴스가 Amazon VPC에 있는 경우 AWS Glue Vertica 연결에 네트워킹 옵션도 제공해야 합니다. 데이터베이스에서 읽고 쓸 때 임시 스토리지로 이용할 Amazon S3 버킷이나 폴더가 필요합니다.

AWS Glue에서 Vertica에 연결하려면 몇 가지 필수 조건이 필요합니다.

- *tempS3Path*에서 참조하여 데이터베이스에서 읽고 쓸 때 임시 스토리지로 사용하는 Amazon S3 버킷 또는 폴더.

> **Note**
>
> AWS Glue 작업 데이터 미리 보기의 Vertica를 사용하는 경우 임시 파일은 *tempS3Path*에서 자동으로 제거되지 않을 수 있습니다. 임시 파일을 제거하려면 데이터 미리 보기 창에서 세션 종료를 선택하여 데이터 미리 보기 세션을 바로 종료합니다.
> 데이터 미리 보기 세션이 바로 종료되도록 보장할 수 없는 경우 이전 데이터를 제거하도록 Amazon S3 수명 주기 구성을 설정하는 것이 좋습니다. 최대 작업 런타임에 여백을 더한 기

준으로 49시간이 지난 데이터는 제거하는 것이 좋습니다. Amazon S3 수명 주기 구성에 대한 자세한 내용은 Amazon S3 설명서에서 스토리지 수명 주기 관리를 참조하십시오.

- Amazon S3 경로에 대한 적절한 권한이 있는 IAM 정책을 AWS Glue 작업 역할과 연결할 수 있습니다.
- Vertica 인스턴스가 Amazon VPC에 있는 경우, 퍼블릭 인터넷을 통과하는 트래픽 없이 AWS Glue 작업이 Vertica 인스턴스와 통신할 수 있도록 Amazon VPC를 구성하십시오.

  Amazon VPC에서 AWS Glue가 작업을 실행하는 동안 사용할 VPC, 서브넷 및 보안 그룹을 식별하거나 생성합니다. 또한 Vertica 인스턴스와 이 위치 간의 네트워크 트래픽을 허용하도록 Amazon VPC를 구성해야 합니다. 작업을 수행하려면 Vertica 클라이언트 포트(기본값 5433)와의 TCP 연결을 설정해야 합니다. 네트워크 레이아웃에 따라 보안 그룹 규칙, 네트워크 ACL, NAT 게이트웨이 및 피어링 연결을 변경해야 할 수도 있습니다.

그런 다음 Vertica와 함께 사용할 AWS Glue를 구성할 수 있습니다.

Vertica에 대한 연결 구성하는 방법:

1. AWS Secrets Manager에서 Vertica 보안 인증 정보, *verticaUsername*, *verticaPassword*을 사용하여 암호를 생성합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS Secrets Manager 설명서의 Create an AWS Secrets Manager secret에서 제공하는 자습서를 따릅니다. 보안 암호를 생성한 후에는 다음 단계를 위해 보안 암호 이름, *secretName*을 유지합니다.
   - 키/값 페어를 선택하면 값 *verticaUsername*이 포함된 키 `user`에 대한 페어를 생성합니다.
   - 키/값 페어를 선택하면 값 *verticaPassword*이 포함된 키 `password`에 대한 페어를 생성합니다.
2. AWS Glue 콘솔에서 the section called “AWS Glue 연결 추가”의 단계에 따라 연결을 생성합니다. 연결을 생성한 후에는 다음 단계를 위해 연결 이름, *connectionName*을 유지합니다.
   - 연결 유형을 선택할 때 Vertica를 선택합니다.
   - Vertica 호스트를 선택할 때 Vertica 설치의 호스트 이름을 제공합니다.
   - Vertica 포트를 선택하면 해당 포트를 통해 Vertica 설치를 사용할 수 있습니다.
   - AWS 보안 암호를 선택할 때 *secretName*을 입력합니다.
3. 다음과 같은 상황에서는 추가 구성이 필요할 수도 있습니다.
   - Amazon VPC에서 AWS에 호스팅된 Vertica 인스턴스의 경우

- Vertica 보안 보안 인증 정보를 정의하는 Amazon VPC 연결 정보를 AWS Glue 연결에 제공하십시오. 연결을 만들거나 업데이트할 때 네트워크 옵션에서 VPC, 서브넷 및 보안 그룹을 설정합니다.

AWS Glue Vertica 연결을 생성한 후에는 연결 방법을 호출하기 전에 다음 단계를 수행해야 합니다.

- AWS Glue 작업과 권한과 연결된 IAM 역할을 *tempS3Path*에 부여합니다.
- AWS Glue 작업과 연결된 IAM 역할에 *secretName*을 읽을 수 있는 권한을 부여합니다.
- AWS Glue 작업 구성에서 추가 네트워크 연결로 *connectionName*을 제공합니다.

Vertica에서 읽는 중

사전 조건:

- 읽으려는 Vertica 테이블. Vertica 데이터베이스 이름인 *dbName*과 테이블 이름인 *tableName*이 필요합니다.
- 인증 정보를 제공하도록 구성된 AWS Glue Vertica 연결. 인증 정보를 구성하려면 앞 절차인 Vertica에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.
- 앞서 언급한 임시 스토리지로 사용할 Amazon S3 버킷 또는 폴더. *tempS3Path*라는 이름이 필요합니다. `s3a` 프로토콜을 사용하여 이 위치에 연결해야 합니다.

예:

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

SELECT SQL 쿼리를 제공하여 DynamicFrame에 반환되는 결과를 필터링할 수도 있고 여러 테이블의 데이터 세트에 액세스할 수도 있습니다.

예:

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "query": "select * FROM tableName",
    },
)
```

Vertica 테이블에 쓰기

이 예제에서는 기존 DynamicFrame, *dynamicFrame*의 정보를 Vertica에 씁니다. 테이블에 이미 정보가 있는 경우 AWS Glue는 DynamicFrame의 데이터를 추가합니다.

사전 조건:

- 쓰려는 현재 또는 원하는 테이블 이름, *tableName*. 또한 해당하는 Vertica 데이터베이스 이름인 *dbName*도 필요합니다.
- 인증 정보를 제공하도록 구성된 AWS Glue Vertica 연결. 인증 정보를 구성하려면 앞 절차인 Vertica에 대한 연결을 구성하는 방법의 단계를 완료하십시오. AWS Glue 연결의 이름인 *connectionName*이 필요합니다.
- 앞서 언급한 임시 스토리지로 사용할 Amazon S3 버킷 또는 폴더. *tempS3Path*라는 이름이 필요합니다. s3a 프로토콜을 사용하여 이 위치에 연결해야 합니다.

예:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

Vertica 연결 옵션 참조

- `connectionName` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법에 인증 및 네트워킹 정보를 제공하도록 구성된 AWS Glue Vertica 연결의 이름입니다.
- `db` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 연결 방법이 상호 작용하는 Vertica의 데이터베이스 이름.
- `dbSchema` - 테이블을 식별하는 데 필요한 경우 필요합니다. 읽기 및 쓰기에 사용됩니다. 기본값: `public`. 연결 방법이 상호 작용할 스키마의 이름.
- `table` - 쓰기 시 필수, `query`가 제공되지 않는 한 읽기 전용. 읽기 및 쓰기에 사용됩니다. 연결 방법이 상호 작용할 테이블의 이름.
- `query` — 읽기에 사용됩니다. Teradata에서 읽을 때 검색해야 하는 내용을 정의하는 SELECT SQL 쿼리입니다.
- `staging_fs_url` — 필수입니다. 읽기 및 쓰기에 사용됩니다. 유효한 값: `s3a` URL. 임시 스토리지로 사용할 Amazon S3 버킷 또는 폴더의 URL입니다.

Spark용 AWS Glue 5.0의 ETL에 대한 DataFrame 옵션

DataFrame은 이름이 지정된 열로 구성되는 데이터세트이며, 테이블과 비슷한 기능적 스타일(맵/줄임/필터 등) 작업과 SQL 작업(선택, 계획, 집계)을 지원합니다.

Glue에서 지원하는 데이터 소스에 대한 DataFrame을 생성하려면 다음이 필요합니다.

- 데이터 소스 커넥터 `ClassName`
- 데이터 소스 연결 `Options`

마찬가지로 Glue에서 지원하는 데이터 싱크에 DataFrame을 작성하려면 동일한 작업이 필요합니다.

- 데이터 싱크 커넥터 `ClassName`
- 데이터 싱크 연결 `Options`

작업 북마크와 같은 AWS Glue 기능 및 `connectionName`과 같은 DynamicFrame 옵션은 DataFrame에서 지원되지 않습니다. DataFrame 및 지원되는 작업에 대한 자세한 내용은 DataFrame용 Spark 설명서를 참조하세요.

커넥터 ClassName 지정

데이터 소스/싱크의 `ClassName`을 지정하려면 `.format` 옵션을 사용하여 데이터 소스/싱크를 정의하는 해당 커넥터 `ClassName`을 제공합니다.

JDBC 커넥터

JDBC 커넥터의 경우 `.format` 옵션의 값으로 `jdbc`를 지정하고 `driver` 옵션에 JDBC 드라이버 `ClassName`을 제공합니다.

```
df = spark.read.format("jdbc").option("driver", "<DATA SOURCE JDBC DRIVER
 CLASSNAME>")...

df.write.format("jdbc").option("driver", "<DATA SINK JDBC DRIVER CLASSNAME>")...
```

다음 표에는 DataFrames용 AWS Glue에서 지원되는 데이터 소스의 JDBC 드라이버 `ClassName`이 나열되어 있습니다.

데이터 소스	Driver ClassName
PostgreSQL	org.postgresql.Driver
Oracle	oracle.jdbc.driver.OracleDriver
SQLServer	com.microsoft.sqlserver.jdbc.SQLServerDriver
MySQL	com.mysql.jdbc.Driver
SAPHana	com.sap.db.jdbc.Driver
Teradata	com.teradata.jdbc.TeraDriver

Spark 커넥터

Spark 커넥터의 경우 커넥터의 `ClassName`을 `.format` 옵션 값으로 지정합니다.

```
df = spark.read.format("<DATA SOURCE CONNECTOR CLASSNAME>")...

df.write.format("<DATA SINK CONNECTOR CLASSNAME>")...
```

다음 표에는 DataFrames용 AWS Glue에서 지원되는 데이터 소스의 Spark 커넥터 `ClassName`이 나열되어 있습니다.

데이터 소스	ClassName
MongoDB/DocumentDB	glue.spark.mongodb
Redshift	io.github.spark_redshift_community.spark.redshift
AzureCosmos	cosmos.oltp
AzureSQL	com.microsoft.sqlserver.jdbc.spark
BigQuery	com.google.cloud.spark.bigquery
OpenSearch	org.opensearch.spark.sql
Snowflake	net.snowflake.spark.snowflake
Vertica	com.vertica.spark.datasource.VerticaSource

연결 옵션 지정

데이터 소스/싱크에 대한 연결의 `Options`를 지정하려면 `.option(<KEY>, <VALUE>)`을 사용하여 개별 옵션을 제공하거나 `.options(<MAP>)`를 사용하여 여러 옵션을 키-값 맵으로 제공합니다.

각 데이터 소스/싱크는 자체 연결 `Options` 세트를 지원합니다. 사용 가능한 `Options`에 대한 자세한 내용은 다음 표에 나열된 특정 데이터 소스/싱크 Spark 커넥터의 퍼블릭 설명서를 참조하세요.

- JDBC
- MongoDB/DocumentDB
- Redshift
- AzureCosmos
- AzureSQL
- BigQuery
- OpenSearch
- Snowflake

- Vertica

예시

다음 예에서는 PostgreSQL에서 읽고 SnowFlake에 씁니다.

Python

예시:

```
from awsglue.context import GlueContext
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

dataSourceClassName = "jdbc"
dataSourceOptions = {
  "driver": "org.postgresql.Driver",
  "url": "<url>",
  "user": "<user>",
  "password": "<password>",
  "dbtable": "<dbtable>",
}

dataframe = spark.read.format(className).options(**options).load()

dataSinkClassName = "net.snowflake.spark.snowflake"
dataSinkOptions = {
  "sfUrl": "<url>",
  "sfUsername": "<username>",
  "sfPassword": "<password>",
  "sfDatabase" -> "<database>",
  "sfSchema" -> "<schema>",
  "sfWarehouse" -> "<warehouse>"
}

dataframe.write.format(dataSinkClassName).options(**dataSinkOptions).save()
```

Scala

예시:

```
import org.apache.spark.sql.SparkSession
```

```
val spark = SparkSession.builder().getOrCreate()

val dataSourceClassName = "jdbc"
val dataSourceOptions = Map(
  "driver" -> "org.postgresql.Driver",
  "url" -> "<url>",
  "user" -> "<user>",
  "password" -> "<password>",
  "dbtable" -> "<dbtable>"
)

val dataframe =
 spark.read.format(dataSourceClassName).options(dataSourceOptions).load()

val dataSinkClassName = "net.snowflake.spark.snowflake"
val dataSinkOptions = Map(
  "sfUrl" -> "<url>",
  "sfUsername" -> "<username>",
  "sfPassword" -> "<password>",
  "sfDatabase" -> "<database>",
  "sfSchema" -> "<schema>",
  "sfWarehouse" -> "<warehouse>"
)

dataframe.write.format(dataSinkClassName).options(dataSinkOptions).save()
```

사용자 정의 및 AWS Marketplace 연결 유형 값

여기에는 다음이 포함됩니다.

- `"connectionType": "marketplace.athena"`: Amazon Athena 데이터 스토어에 대한 연결을 지정합니다. 연결은 AWS Marketplace의 커넥터를 사용합니다.
- `"connectionType": "marketplace.spark"`: Apache Spark 데이터 스토어에 대한 연결을 지정합니다. 연결은 AWS Marketplace의 커넥터를 사용합니다.
- `"connectionType": "marketplace.jdbc"`: JDBC 데이터 스토어에 대한 연결을 지정합니다. 연결은 AWS Marketplace의 커넥터를 사용합니다.
- `"connectionType": "custom.athena"`: Amazon Athena 데이터 스토어에 대한 연결을 지정합니다. 연결은 AWS Glue Studio에 업로드하는 사용자 지정 커넥터를 사용합니다.
- `"connectionType": "custom.spark"`: Apache Spark 데이터 스토어에 대한 연결을 지정합니다. 연결은 AWS Glue Studio에 업로드하는 사용자 지정 커넥터를 사용합니다.

- `"connectionType": "custom.jdbc"`: JDBC 데이터 스토어에 대한 연결을 지정합니다. 연결은 AWS Glue Studio에 업로드하는 사용자 지정 커넥터를 사용합니다.

custom.jdbc 또는 marketplace.jdbc 유형에 대한 연결 옵션

- `className` - 문자열, 필수, 드라이버 클래스 이름입니다.
- `connectionName` - 문자열, 필수, 커넥터와 연결된 연결 이름입니다.
- `url`- 문자열, 필수, 데이터 원본에 대한 연결을 구축하는 데 사용되는 자리 표시자(`${}`)가 있는 JDBC URL입니다. 자리 표시자 `${secretKey}`는 AWS Secrets Manager에서 같은 이름의 보안 암호로 대체됩니다. URL 구성에 대한 자세한 내용은 데이터 스토어 설명서를 참조하세요.
- `secretId` 또는 `user/password` – 문자열, 필수, URL에 대한 자격 증명을 검색하는 데 사용됩니다.
- `dbTable` 또는 `query` - 문자열, 필수, 데이터를 가져올 테이블 또는 SQL 쿼리입니다. `dbTable` 또는 `query`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.
- `partitionColumn`- 문자열, 선택 사항, 파티셔닝에 사용되는 정수 열의 이름입니다. 이 옵션은 `lowerBound`, `upperBound` 및 `numPartitions`에 포함되는 경우에만 작동합니다. 이 옵션은 Spark SQL JDBC 리더에서와 같은 방식으로 작동합니다. 자세한 내용은 Apache Spark SQL, DataFrames and Datasets Guide의 JDBC To Other Databases를 참조하세요.

  `lowerBound` 및 `upperBound` 값은 테이블의 행을 필터링하는 것이 아니라 파티션 스트라이드를 결정하는 데 사용됩니다. 테이블의 모든 행이 분할되어 반환됩니다.

  > **Note**
  >
  > 테이블 이름 대신 쿼리를 사용하는 경우 쿼리가 지정된 분할 조건에서 작동하는지 확인해야 합니다. 예:
  >
  > - 쿼리 포맷이 `"SELECT col1 FROM table1"`이면 파티션 열을 사용하는 쿼리 끝에 `WHERE` 절을 추가하여 쿼리를 테스트합니다.
  > - 쿼리 포맷이 `"SELECT col1 FROM table1 WHERE col2=val"`이면 `AND`와 파티션 열을 사용하는 표현식으로 `WHERE` 절을 확장하여 쿼리를 테스트합니다.

- `lowerBound` - 정수, 선택 사항, 파티션 스트라이드를 결정하는 데 사용되는 `partitionColumn`의 최소값입니다.
- `upperBound` - 정수, 선택 사항, 파티션 스트라이드를 결정하는 데 사용되는 `partitionColumn`의 최대값입니다.

- numPartitions- 정수, 선택 사항, 파티션 수입니다. 이 값은 lowerBound(포함) 및 upperBound(배타)와 함께 partitionColumn을 분할하는 데 사용되는 생성된 WHERE 절 표현에 대한 파티션 스트라이드를 형성합니다.

  > **Important**
  >
  > 파티션이 너무 많으면 외부 데이터베이스 시스템에 문제가 발생할 수 있으므로 파티션 수에 주의합니다.

- filterPredicate - 문자열, 선택 사항, 소스에서 데이터를 필터링하기 위한 추가 조건 절입니다. 예:

  ```
  BillingCity='Mountain View'
  ```

  테이블 이름 대신 쿼리를 사용하는 경우 쿼리가 지정된 filterPredicate에서 작동하는지 검증해야 합니다. 예:

  - 쿼리 포맷이 "SELECT col1 FROM table1"이면 필터 조건자를 사용하는 쿼리 끝에 WHERE 절을 추가하여 쿼리를 테스트합니다.
  - 쿼리 포맷이 "SELECT col1 FROM table1 WHERE col2=val"이면 AND로 WHERE 절을 확장하고 필터 조건자를 사용하는 표현식을 사용하여 쿼리를 테스트합니다.

- dataTypeMapping – 딕셔너리, 선택 사항, JDBC 데이터 유형에서 Glue 데이터 유형으로의 매핑을 구축하는 사용자 정의 데이터 유형 매핑입니다. 예를 들어 "dataTypeMapping": {"FLOAT":"STRING"} 옵션은 드라이버의 ResultSet.getString() 메서드를 호출하여 JDBC 유형 FLOAT의 데이터 필드를 Java String 유형으로 매핑하고 이를 AWS Glue 레코드를 구축하는 데 사용합니다. ResultSet 객체는 각 드라이버에 의해 구현되므로 동작은 사용하는 드라이버에 따라 다릅니다. 드라이버가 변환을 수행하는 방법을 이해하려면 JDBC 드라이버에 대한 설명서를 참조하세요.
- 현재 지원되는 AWS Glue 데이터 유형은 다음과 같습니다.
  - DATE
  - STRING
  - TIMESTAMP
  - INT
  - FLOAT
  - LONG
  - BIGDECIMAL

- BYTE
- SHORT
- DOUBLE

지원되는 JDBC 데이터 유형은 Java8 java.sql.types입니다.

기본 데이터 유형 매핑(JDBC에서 AWS Glue로)은 다음과 같습니다.

- DATE -> DATE
- VARCHAR -> STRING
- CHAR -> STRING
- LONGNVARCHAR -> STRING
- TIMESTAMP -> TIMESTAMP
- INTEGER -> INT
- FLOAT -> FLOAT
- REAL -> FLOAT
- BIT -> BOOLEAN
- BOOLEAN -> BOOLEAN
- BIGINT -> LONG
- DECIMAL -> BIGDECIMAL
- NUMERIC -> BIGDECIMAL
- TINYINT -> SHORT
- SMALLINT -> SHORT
- DOUBLE -> DOUBLE

옵션 `dataTypeMapping`과 함께 사용자 정의 데이터 유형 매핑을 사용하는 경우 기본 데이터 유형 매핑을 재정의할 수 있습니다. `dataTypeMapping` 옵션에 나열된 JDBC 데이터 유형만 영향을 받습니다. 기본 매핑은 다른 모든 JDBC 데이터 유형에 사용됩니다. 필요한 경우 추가 JDBC 데이터 유형에 대한 매핑을 추가할 수 있습니다. JDBC 데이터 유형이 기본 매핑이나 사용자 지정 매핑에 포함되지 않은 경우 데이터 유형은 기본값으로 AWS Glue `STRING` 데이터 유형으로 변환됩니다.

다음 Python 코드 예제는 AWS Marketplace JDBC 드라이버를 사용하여 JDBC 데이터베이스에서 읽는 방법을 보여줍니다. 데이터베이스에서 읽고 S3 위치에 쓰는 방법을 보여줍니다.

```
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

    ## @params: [JOB_NAME]
    args = getResolvedOptions(sys.argv, ['JOB_NAME'])

    sc = SparkContext()
    glueContext = GlueContext(sc)
    spark = glueContext.spark_session
    job = Job(glueContext)
    job.init(args['JOB_NAME'], args)
    ## @type: DataSource
    ## @args: [connection_type = "marketplace.jdbc", connection_options =
     {"dataTypeMapping":{"INTEGER":"STRING"},"upperBound":"200","query":"select id,
       name, department from department where id < 200","numPartitions":"4",
       "partitionColumn":"id","lowerBound":"0","connectionName":"test-connection-
jdbc"},
        transformation_ctx = "DataSource0"]
    ## @return: DataSource0
    ## @inputs: []
    DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
      "marketplace.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"},
      "upperBound":"200","query":"select id, name, department from department where
       id < 200","numPartitions":"4","partitionColumn":"id","lowerBound":"0",
       "connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
    ## @type: ApplyMapping
    ## @args: [mappings = [("department", "string", "department", "string"), ("name",
 "string",
      "name", "string"), ("id", "int", "id", "int")], transformation_ctx =
 "Transform0"]
    ## @return: Transform0
    ## @inputs: [frame = DataSource0]
    Transform0 = ApplyMapping.apply(frame = DataSource0, mappings = [("department",
 "string",
      "department", "string"), ("name", "string", "name", "string"), ("id", "int",
 "id", "int")],
       transformation_ctx = "Transform0")
    ## @type: DataSink
    ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
     "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
    ## @return: DataSink0
```

```
    ## @inputs: [frame = Transform0]
    DataSink0 = glueContext.write_dynamic_frame.from_options(frame = Transform0,
      connection_type = "s3", format = "json", connection_options = {"path":
      "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
    job.commit()
```

custom.athena 또는 marketplace.athena 유형에 대한 연결 옵션

- `className` - 문자열, 필수, 드라이버 클래스 이름입니다. Athena-CloudWatch 커넥터를 사용하는 경우 이 파라미터 값은 클래스 이름의 접두사(예: `"com.amazonaws.athena.connectors"`)입니다. Athena-CloudWatch 커넥터는 메타데이터 핸들러와 레코드 핸들러의 두 가지 클래스로 구성됩니다. 여기에 공통 접두사를 제공하면 API가 해당 접두사를 기반으로 올바른 클래스를 로드합니다.
- `tableName` - 문자열, 필수, 읽을 CloudWatch 로그 스트림의 이름입니다. 이 코드 조각은 특수 보기 이름 `all_log_streams`을 사용합니다. 즉, 반환된 동적 데이터 프레임에는 로그 그룹의 모든 로그 스트림 데이터가 포함됩니다.
- `schemaName` - 문자열, 필수, 읽을 CloudWatch 로그 그룹의 이름입니다. 예: `/aws-glue/jobs/output`.
- `connectionName` - 문자열, 필수, 커넥터와 연결된 연결 이름입니다.

이 커넥터에 대한 추가 옵션은 GitHub의 Amazon Athena CloudWatch 커넥터 README를 참조하세요.

다음 Python 코드 예제는 AWS Marketplace 커넥터를 사용하여 Athena 데이터 스토어에서 읽는 방법을 보여줍니다. Athena에서 읽고 S3 위치에 쓰는 방법을 보여줍니다.

```
    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

    ## @params: [JOB_NAME]
    args = getResolvedOptions(sys.argv, ['JOB_NAME'])

    sc = SparkContext()
    glueContext = GlueContext(sc)
    spark = glueContext.spark_session
    job = Job(glueContext)
    job.init(args['JOB_NAME'], args)
    ## @type: DataSource
```

```
    ## @args: [connection_type = "marketplace.athena", connection_options =
     {"tableName":"all_log_streams","schemaName":"/aws-glue/jobs/output",
      "connectionName":"test-connection-athena"}, transformation_ctx = "DataSource0"]
    ## @return: DataSource0
    ## @inputs: []
    DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
      "marketplace.athena", connection_options = {"tableName":"all_log_streams",,
      "schemaName":"/aws-glue/jobs/output","connectionName":
      "test-connection-athena"}, transformation_ctx = "DataSource0")
    ## @type: ApplyMapping
    ## @args: [mappings = [("department", "string", "department", "string"), ("name",
"string",
     "name", "string"), ("id", "int", "id", "int")], transformation_ctx =
"Transform0"]
    ## @return: Transform0
    ## @inputs: [frame = DataSource0]
    Transform0 = ApplyMapping.apply(frame = DataSource0, mappings = [("department",
"string",
     "department", "string"), ("name", "string", "name", "string"), ("id", "int",
"id", "int")],
      transformation_ctx = "Transform0")
    ## @type: DataSink
    ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
     "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
    ## @return: DataSink0
    ## @inputs: [frame = Transform0]
    DataSink0 = glueContext.write_dynamic_frame.from_options(frame = Transform0,
      connection_type = "s3", format = "json", connection_options = {"path":
      "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
    job.commit()
```

custom.spark 또는 marketplace.spark 유형에 대한 연결 옵션

- `className` - 문자열, 필수, 커넥터 클래스 이름입니다.
- `secretId` - 문자열, 선택 사항, 커넥터 연결에 대한 자격 증명을 검색하는 데 사용됩니다.
- `connectionName` - 문자열, 필수, 커넥터와 연결된 연결 이름입니다.
- 다른 옵션은 데이터 스토어에 따라 다릅니다. 예를 들어 OpenSearch 구성 옵션은 Apache Hadoop 용 Elasticsearch 설명서에 설명된 대로 접두사 `es`로 시작합니다. Snowflake에 대한 Spark 연결은 Connecting to Snowflake 가이드의 Using the Spark Connector에 설명된 대로 `sfUser` 및 `sfPassword`와 같은 옵션을 사용합니다.

다음 Python 코드 예제는 marketplace.spark 연결을 사용하여 OpenSearch 데이터 스토어에서 읽는 방법을 보여줍니다.

```
   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

   ## @params: [JOB_NAME]
   args = getResolvedOptions(sys.argv, ['JOB_NAME'])

   sc = SparkContext()
   glueContext = GlueContext(sc)
   spark = glueContext.spark_session
   job = Job(glueContext)
   job.init(args['JOB_NAME'], args)
   ## @type: DataSource
   ## @args: [connection_type = "marketplace.spark", connection_options =
{"path":"test",
     "es.nodes.wan.only":"true","es.nodes":"https://<AWS endpoint>",
     "connectionName":"test-spark-es","es.port":"443"}, transformation_ctx =
"DataSource0"]
   ## @return: DataSource0
   ## @inputs: []
   DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
     "marketplace.spark", connection_options = {"path":"test","es.nodes.wan.only":
     "true","es.nodes":"https://<AWS endpoint>","connectionName":
     "test-spark-es","es.port":"443"}, transformation_ctx = "DataSource0")
   ## @type: DataSink
   ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
        "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
   ## @return: DataSink0
   ## @inputs: [frame = DataSource0]
   DataSink0 = glueContext.write_dynamic_frame.from_options(frame = DataSource0,
      connection_type = "s3", format = "json", connection_options = {"path":
      "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
   job.commit()
```

일반 옵션

이 섹션의 옵션은 connection_options로 제공되지만 특별히 하나의 커넥터에만 적용되지는 않습니다.

다음 파라미터는 보통 북마크를 구성할 때 사용됩니다. Amazon S3 또는 JDBC 워크플로에 적용될 수도 있습니다. 자세한 내용은 the section called “작업 북마크 사용” 단원을 참조하십시오.

- jobBookmarkKeys - 열 이름의 배열입니다.
- jobBookmarkKeysSortOrder - 정렬 순서에 따라 값을 비교하는 방법을 정의하는 문자열입니다. 유효한 값: "asc", "desc".
- useS3ListImplementation - Amazon S3 버킷 콘텐츠를 나열할 때 메모리 성능을 관리하는 데 사용됩니다. 자세한 내용은 Optimize memory management in AWS Glue를 참조하세요.

AWS Glue for Spark에서 입력 및 출력의 데이터 형식 옵션

이 페이지는 AWS Glue for Spark에서 지원하는 데이터 형식에 대한 기능 지원 및 구성 파라미터에 대한 정보를 제공합니다. 이 정보의 사용 및 적용 가능성에 대한 설명은 다음을 참조하세요.

AWS Glue의 데이터 형식 전반에 걸친 기능 지원

각 데이터 형식은 다른 AWS Glue 기능을 지원할 수 있습니다. 다음 공통 기능은 해당 형식 유형에 따라 지원되거나 지원되지 않을 수 있습니다. 요구 사항을 충족하기 위해 기능을 활용하는 방법을 이해하려면 데이터 형식에 대한 설명서를 참조하세요.

읽기	AWS Glue는 커넥터와 같은 추가 리소스 없이 이 데이터 형식을 인식하고 해석할 수 있습니다.
쓰기	AWS Glue는 추가 리소스 없이 이 형식으로 데이터를 쓸 수 있습니다. 다른 Spark 환경에서와 같이 작업에 타사 라이브러리를 포함하고 표준 Apache Spark 기능을 사용하여 데이터를 작성할 수 있습니다. 라이브러리를 포함한 자세한 내용은 the section called “Python 라이브러리” 단원을 참조하세요.
스트리밍 읽기	AWS Glue는 Apache Kafka, Amazon Managed Streaming for Apache Kafka 또는 Amazon Kinesis 메시지 스트림에서 이 데이터 형식을 인

	식하고 해석할 수 있습니다. 스트림이 일관된 형식으로 데이터를 표시할 것으로 예상하므로 `DataFrames` 로 읽습니다.
작은 파일 그룹화	AWS Glue는 AWS Glue 변환을 수행할 때 각 노드로 전송되는 일괄 작업으로 파일을 그룹화할 수 있습니다. 이렇게 하면 많은 양의 작은 파일이 포함된 워크로드의 성능이 크게 향상될 수 있습니다. 자세한 내용은 the section called “입력 파일 그룹화” 단원을 참조하십시오.
작업 북마크	AWS Glue는 작업 북마크를 사용하여 작업 실행 전반에 걸쳐 동일한 데이터 세트에서 동일한 작업을 수행하는 변환의 진행 상황을 추적할 수 있습니다. 이렇게 하면 마지막 작업 실행 이후 새 데이터에 대해서만 작업을 수행해야 하는 데이터 세트와 관련된 워크로드의 성능을 향상시킬 수 있습니다. 자세한 내용은 the section called “처리된 데이터를 작업 북마크로 추적” 단원을 참조하십시오.

AWS Glue의 데이터 형식과 상호 작용하는 데 사용되는 파라미터

특정 AWS Glue 연결 유형은 여러 `format` 유형을 지원하므로 `GlueContext.write_dynamic_frame.from_options`와 같은 방법을 사용할 때 `format_options` 객체로 데이터 형식에 대한 정보를 지정해야 합니다.

- `s3` - 자세한 내용은 AWS Glue: S3 연결 파라미터의 ETL 연결 유형 및 옵션을 참조하세요. 이 연결 유형(Python에서 the section called “create_dynamic_frame_from_options” 및 the section called “write_dynamic_frame_from_options”, 그에 해당하는 스칼라 메서드 the section called “getSourceWithFormat” 및 the section called “getSinkWithFormat”)을 지원하는 메서드에 대한 문서도 볼 수 있습니다.

- `kinesis` - 자세한 내용은 AWS Glue: Kinesis 연결 파라미터의 ETL 연결 유형 및 옵션을 참조하세요. 이 연결 유형(the section called “create_data_frame_from_options” 및 그에 해당하는 스칼라 메서드 the section called “createDataFrameFromOptions”)을 지원하는 메서드에 대한 문서도 볼 수 있습니다.

- `kafka` - 자세한 내용은 AWS Glue: Kafka 연결 파라미터의 ETL 연결 유형 및 옵션을 참조하세요. 이 연결 유형(the section called “create_data_frame_from_options” 및 그에 해당하는 스칼라 메서드 the section called “createDataFrameFromOptions”)을 지원하는 메서드에 대한 문서도 볼 수 있습니다.

일부 연결 유형은 `format_options`가 필요하지 않습니다. 예를 들어 일반적인 사용 과정에서 관계형 데이터베이스에 대한 JDBC 연결은 일관된 테이블형 데이터 형식으로 데이터를 검색하므로 JDBC 연결에서 읽을 때 `format_options`가 필요하지 않습니다.

Glue에서 데이터를 읽고 쓰는 일부 방법은 `format_options`가 필요하지 않습니다. 예를 들어, Glue 크롤러에서 `GlueContext.create_dynamic_frame.from_catalog` 및 AWS를 사용하는 경우. 크롤러는 데이터의 형태를 결정합니다. 크롤러를 사용할 때 AWS Glue 분류기는 데이터를 검사하여 데이터 형식을 표현하는 방법에 대한 현명한 결정을 내립니다. 그런 다음 AWS Glue ETL 스크립트 내에서 `GlueContext.create_dynamic_frame.from_catalog` 메서드로 데이터를 검색하는 데 사용할 수 있는 AWS Glue 데이터 카탈로그에 데이터 표현을 저장합니다. 크롤러를 사용하면 데이터 형식에 대한 정보를 수동으로 지정할 필요가 없습니다.

AWS Lake Formation 관리형 테이블에 액세스하는 작업의 경우 AWS Glue는 Lake Formation 관리형 테이블에서 지원되는 모든 포맷을 읽고 쓸 수 있도록 지원합니다. AWS Lake Formation 관리형 테이블에서 현재 지원되는 포맷 목록은 AWS Lake Formation 개발자 가이드의 관리형 테이블에 대한 참고 사항 및 제한 사항을 참조하세요.

> **Note**
>
> Apache Parquet 쓰기의 경우 AWS Glue ETL은 Dynamic Frames에 최적화된 사용자 지정 Parquet 라이터 유형에 대한 옵션을 지정하여 관리형 테이블에 쓰는 기능만을 지원합니다. `parquet` 포맷을 사용하는 관리형 테이블에 쓰는 경우 테이블 파라미터의 `true` 값과 함께 `useGlueParquetWriter` 키를 추가해야 합니다.

주제

- AWS Glue에서 CSV 형식 사용
- AWS Glue에서 Parquet 형식 사용
- AWS Glue에서 XML 형식 사용
- AWS Glue에서 Avro 형식 사용
- AWS Glue에서 grokLog 형식 사용
- AWS Glue에서 Ion 형식 사용

- AWS Glue에서 JSON 형식 사용
- AWS Glue에서 ORC 형식 사용
- AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용
- 공유 구성 참조

AWS Glue에서 CSV 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 CSV 데이터 형식으로 저장 또는 전송되는 경우 이 문서에서는 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 CSV(쉼표로 분리된 값) 형식 사용만 지원합니다. 이 형식은 최소 행 기반 데이터 형식입니다. CSV는 표준을 엄격하게 준수하지 않는 경우가 많지만 RFC 4180과 RFC 7111에서 자세한 정보를 참조할 수 있습니다.

AWS Glue를 사용하여 Amazon S3와 스트리밍 소스에서 CSV를 읽을 수 있을 뿐만 아니라 Amazon S3에 CSV를 쓸 수 있습니다. S3에서 CSV 파일이 포함된 `bzip` 및 `gzip` 아카이브를 읽고 쓸 수 있습니다. 이 페이지에서 설명하는 구성 대신 S3 연결 파라미터에서 압축 동작을 구성할 수 있습니다.

다음 표에서는 CSV 형식 옵션을 지원하는 일반적인 AWS Glue 기능을 보여줍니다.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크
지원	지원	지원	지원	지원

예: S3에서 CSV 파일 또는 폴더 읽기

사전 조건: 읽고자 하는 CSV 파일 또는 폴더에 대한 S3 경로(`s3path`)가 필요합니다.

구성: 함수 옵션에서 `format="csv"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 리더와 S3가 상호 작용하는 방식을 구성할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션 참조: S3 연결 파라미터 리더에서 `format_options`의 CSV 파일을 해석하는 방법을 구성할 수 있습니다. 자세한 내용은 CSV 구성 참조를 참조하십시오.

다음 AWS Glue ETL 스크립트는 S3에서 CSV 파일 또는 폴더를 읽는 프로세스를 보여줍니다.

optimizePerformance 구성 키를 통해 사용자 지정 CSV 리더에 일반적인 워크플로우에 대한 성능 최적화가 제공됩니다. 이 리더가 워크로드에 적합한지 확인하려면 the section called "최적화된 CSV 리더 사용" 단원을 참조하십시오.

Python

이 예에서는 create_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Read CSV from S3
# For show, we handle a CSV with a header row.  Set the withHeader option.
# Consider whether optimizePerformance is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="csv",
    format_options={
        "withHeader": True,
        # "optimizePerformance": True,
    },
)
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
dataFrame = spark.read\
    .format("csv")\
    .option("header", "true")\
    .load("s3://s3path")
```

Scala

이 예에서는 getSourceWithFormat 작업을 사용합니다.

```
// Example: Read CSV from S3
// For show, we handle a CSV with a header row.  Set the withHeader option.
```

```
// Consider whether optimizePerformance is right for your workflow.

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

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"withHeader": true}"""),
      connectionType="s3",
      format="csv",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

또한 스크립트(`org.apache.spark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
val dataFrame = spark.read
  .option("header","true")
  .format("csv")
  .load("s3://s3path“)
```

예: S3에 CSV 파일 및 폴더 쓰기

사전 조건: 초기화된 DataFrame(`dataFrame`) 또는 DynamicFrame(`dynamicFrame`)이 필요합니다. 예상되는 S3 출력 경로(`s3path`)도 필요합니다.

구성: 함수 옵션에서 `format="csv"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 라이터와 S3가 상호 작용하는 방식을 구성할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션 참조: S3 연결 파라미터 작업에서 `format_options`에 있는 파일의 내용을 쓰는 방법을 구성할 수 있습니다. 자세한 내용은 CSV 구성 참조를 참조하십시오. 다음 AWS Glue ETL 스크립트는 S3로 CSV 파일 및 폴더를 쓰는 프로세스를 보여줍니다.

Python

이 예에서는 write_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Write CSV to S3
# For show, customize how we write string type values.  Set quoteChar to -1 so our
 values are not quoted.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    connection_options={"path": "s3://s3path"},
    format="csv",
    format_options={
        "quoteChar": -1,
    },
)
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
dataFrame.write\
    .format("csv")\
    .option("quote", None)\
    .mode("append")\
    .save("s3://s3path")
```

Scala

이 예에서는 getSinkWithFormat 메서드를 사용합니다.

```
// Example: Write CSV to S3
// For show, customize how we write string type values. Set quoteChar to -1 so our
 values are not quoted.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
```

```
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="csv"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

또한 스크립트(`org.apache.spark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
dataFrame.write
    .format("csv")
    .option("quote", null)
    .mode("Append")
    .save("s3://s3path")
```

CSV 구성 참조

AWS Glue 라이브러리가 `format="csv"`를 지정한 곳이라면 어디에서든 다음 `format_options`을 사용할 수 있습니다.

- `separator` - 구분 기호 문자열을 지정합니다. 기본값은 쉼표(,)이지만, 다른 문자도 지정할 수 있습니다.
  - 유형: 텍스트, 기본값: `","`
- `escaper` - 이스케이프 처리에 사용할 문자를 지정합니다. 이 옵션은 CSV 파일을 읽을 때만 사용되며 쓸 때는 사용되지 않습니다. 활성화된 경우 바로 다음에 나오는 문자가 잘 알려진 이스케이프 세트(`\n`, `\r`, `\t` 및 `\0`)를 제외하고는 있는 그대로 사용됩니다.
  - 유형: 텍스트, 기본값: 없음
- `quoteChar` - 인용에 사용할 문자를 지정합니다. 기본값은 큰 따옴표(")입니다. 전체 인용을 해제하려면 이 값을 `-1`로 설정합니다.
  - 유형: 텍스트, 기본값: `'"'`

- `multiLine` - 단일 기록이 다양한 라인을 포괄할 수 있는지 여부를 지정합니다. 필드가 인용된 새로운 라인 문자를 포함할 때 발생합니다. 이 옵션을 `True`로 설정해야 기록이 여러 라인을 포괄할 수 있습니다. `multiLine`을 활성화하면 구문 분석하는 동안 더 신중한 파일 분할이 필요하므로 성능이 저하될 수 있습니다.
  - 유형: 부울, 기본값: `false`
- `withHeader` - 첫 번째 라인을 헤더로 취급할지 여부를 지정합니다. 이 옵션은 `DynamicFrameReader` 클래스에서 사용할 수 있습니다.
  - 유형: 부울, 기본값: `false`
- `writeHeader` - 헤더를 작성하여 출력할지 여부를 지정합니다. 이 옵션은 `DynamicFrameWriter` 클래스에서 사용할 수 있습니다.
  - 유형: 부울, 기본값: `true`
- `skipFirst`- 첫 번째 데이터 라인을 건너뛸지 여부를 지정합니다.
  - 유형: 부울, 기본값: `false`
- `optimizePerformance` - Apache Arrow 기반 열 포맷 메모리 포맷과 함께 고급 SIMD CSV 리더를 사용할지 여부를 지정합니다. AWS Glue 3.0 이상에서만 사용 가능합니다.
  - 유형: 부울, 기본값: `false`
- `strictCheckForQuoting` - CSV를 쓸 때 Glue는 문자열로 해석되는 값에 따옴표를 추가할 수 있습니다. 이는 기록된 내용이 모호하지 않도록 하기 위해서입니다. 기록할 내용을 결정할 때 시간을 절약하기 위해 Glue는 따옴표가 필요하지 않은 특정 상황에서 인용할 수 있습니다. 엄격한 검사를 활성화하면 보다 컴퓨팅 집약적인 작업을 수행하고 필요한 경우에만 인용합니다. AWS Glue 3.0 이상에서만 사용 가능합니다.
  - 유형: 부울, 기본값: `false`

벡터화된 SIMD CSV 리더로 읽기 성능 최적화

AWS Glue 버전 3.0에는 행 기반 CSV 리더에 비해 전반적인 작업 속도를 크게 높일 수 있는 최적화된 CSV 리더가 추가되었습니다.

최적화된 리더:

- CPU SIMD 명령을 사용하여 디스크에서 읽기
- 레코드를 열 기반 형식으로 메모리에 즉시 쓰기(Apache Arrow)
- 레코드를 배치로 분할

이렇게 하면 나중에 레코드가 일괄 처리되거나 열 기반 형식으로 변환될 때 처리 시간이 절약됩니다. 스키마를 변경하거나 열별로 데이터를 검색하는 경우를 예로 들 수 있습니다.

최적화된 리더를 사용하려면 `format_options` 또는 테이블 속성에서 `"optimizePerformance"`를 `true`로 설정합니다.

```
glueContext.create_dynamic_frame.from_options(
    frame = datasource1,
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path"]},
    format = "csv",
    format_options={
        "optimizePerformance": True,
        "separator": ","
        },
    transformation_ctx = "datasink2")
```

벡터화된 CSV 리더에 대한 제한 사항

벡터화된 CSV 리더의 제한 사항:

- `multiLine` 및 `escaper` 포맷 옵션은 지원되지 않습니다. 큰따옴표 문자(`'"'`)의 기본값 `escaper`가 사용됩니다. 이러한 옵션을 설정하면 AWS Glue는 행 기반 CSV 리더를 사용하는 상태로 자동으로 돌아갑니다.
- ChoiceType으로 DynamicFrame 생성이 지원되지 않습니다.
- 오류 레코드로 DynamicFrame 생성이 지원되지 않습니다.
- 일본어 또는 중국어와 같은 멀티바이트 문자가 포함된 CSV 파일 읽기가 지원되지 않습니다.

AWS Glue에서 Parquet 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 Parquet 데이터 형식으로 저장 또는 전송되는 경우 이 문서에서는 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 Parquet 형식 사용을 지원합니다. 이 형식은 성능 중심의 열 기반 데이터 형식입니다. 표준 기관의 형식에 대한 소개는 Apache Parquet 설명서 개요를 참조하세요.

AWS Glue를 사용하여 Amazon S3와 스트리밍 소스에서 Parquet 파일을 읽을 수 있을 뿐만 아니라 Amazon S3에 Parquet 파일을 쓸 수 있습니다. S3에서 Parquet 파일이 포함된 `bzip` 및 `gzip` 아카이

브를 읽고 쓸 수 있습니다. 이 페이지에서 설명하는 구성 대신 S3 연결 파라미터에서 압축 동작을 구성할 수 있습니다.

다음 표에서는 Parquet 형식 옵션을 지원하는 일반적인 AWS Glue 기능을 보여줍니다.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크
지원	지원	지원	지원되지 않음	지원*

* AWS Glue 버전 1.0 이상에서 지원

예: S3에서 Parquet 파일 또는 폴더 읽기

사전 조건: 읽고자 하는 Parquet 파일 또는 폴더에 대한 S3 경로(`s3path`)가 필요합니다.

구성: 함수 옵션에서 `format="parquet"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다.

`connection_options`에서 리더와 S3가 상호 작용하는 방식을 구성할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션 참조: S3 연결 파라미터

리더에서 `format_options`의 Parquet 파일을 해석하는 방법을 구성할 수 있습니다. 자세한 내용은 Parquet 구성 참조를 참조하십시오.

다음 AWS Glue ETL 스크립트는 S3에서 Parquet 파일 또는 폴더를 읽는 프로세스를 보여줍니다.

Python

이 예에서는 create_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Read Parquet from S3

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type = "s3",
```

```
    connection_options = {"paths": ["s3://s3path/"]},
    format = "parquet"
)
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
dataFrame = spark.read.parquet("s3://s3path/")
```

Scala

이 예에서는 getSourceWithFormat 메서드를 사용합니다.

```
// Example: Read Parquet from S3

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

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="parquet",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
}
```

또한 스크립트(org.apache.spark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
spark.read.parquet("s3://s3path/")
```

예: S3에 Parquet 파일 및 폴더 쓰기

사전 조건: 초기화된 DataFrame(dataFrame) 또는 DynamicFrame(dynamicFrame)이 필요합니다. 예상되는 S3 출력 경로(s3path)도 필요합니다.

구성: 함수 옵션에서 `format="parquet"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다.

`connection_options`에서 라이터가 S3와 상호 작용하는 방식을 추가로 변경할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션 참조: S3 연결 파라미터 작업에서 `format_options`에 있는 파일의 내용을 쓰는 방법을 구성할 수 있습니다. 자세한 내용은 Parquet 구성 참조를 참조하십시오.

다음 AWS Glue ETL 스크립트는 S3로 Parquet 파일 및 폴더를 쓰는 프로세스를 보여줍니다.

`useGlueParquetWriter` 구성 키를 통해 사용자 지정 Parquet 라이터에 DynamicFrame에 대한 성능 최적화가 제공됩니다. 이 라이터가 워크로드에 적합한지 확인하려면 Glue Parquet 라이터를 참조하세요.

Python

이 예에서는 write_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Write Parquet to S3
# Consider whether useGlueParquetWriter is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="parquet",
    connection_options={
        "path": "s3://s3path",
    },
    format_options={
        # "useGlueParquetWriter": True,
    },
)
```

또한 스크립트(`pyspark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
df.write.parquet("s3://s3path/")
```

Scala

이 예에서는 getSinkWithFormat 메서드를 사용합니다.

```
// Example: Write Parquet to S3
// Consider whether useGlueParquetWriter is right for your workflow.

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

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="parquet"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

또한 스크립트(`org.apache.spark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
df.write.parquet("s3://s3path/")
```

Parquet 구성 참조

AWS Glue 라이브러리가 `format="parquet"`을 지정한 어디에서나 다음 `format_options`를 사용할 수 있습니다.

- `useGlueParquetWriter` - DynamicFrame 워크플로에 대한 성능 최적화가 포함된 사용자 지정 Parquet 라이터의 사용을 지정합니다. 자세한 내용은 Glue Parquet 라이터를 참조하세요.
  - 유형: 부울, 기본값:`false`
- `compression` - 사용되는 압축 코덱을 지정합니다. 값은 `org.apache.parquet.hadoop.metadata.CompressionCodecName`과 완벽하게 호환됩니다.
  - 유형: 열거 텍스트, 기본값: `"snappy"`

  - 값: "uncompressed", "snappy", "gzip" 및 "lzo"
- blockSize - 메모리에 버퍼링되는 행 그룹의 크기(바이트 단위)를 지정합니다. 성능 튜닝에 사용합니다. 크기는 정확히 메가바이트 수로 나눠야 합니다.
  - 유형: 숫자, 기본값:134217728
  - 기본값은 128MB입니다.
- pageSize - 페이지의 크기를 바이트 단위로 지정합니다. 성능 튜닝에 사용합니다. 페이지는 단일 레코드에 액세스하기 위해 완전하게 읽어야 하는 가장 작은 단위입니다.
  - 유형: 숫자, 기본값:1048576
  - 기본값은 1MB입니다.

> **Note**
>
> 또한 SparkSQL 기초 코드에 의해 수락된 옵션은 connection_options 맵 파라미터를 거쳐 SparkSQL 기초 코드로 넘겨질 수 있습니다. 예를 들어 AWS Glue Spark 리더에 대해 mergeSchema와 같은 Spark 구성을 설정하여 모든 파일의 스키마를 병합할 수 있습니다.

AWS Glue Parquet 라이터를 사용한 쓰기 성능 최적화

> **Note**
>
> AWS Glue Parquet 라이터는 예전에 glueparquet 형식 유형을 통해 액세스되었습니다. 이 액세스 패턴은 더 이상 지원되지 않습니다. 대신 useGlueParquetWriter이 활성화된 parquet 유형을 사용합니다.

AWS Glue Parquet 라이터는 더 빠른 Parquet 파일 쓰기가 가능하도록 성능이 향상되었습니다. 기존 라이터는 쓰기 전에 스키마를 컴퓨팅합니다. Parquet 형식은 빠르게 검색할 수 있는 방식으로 스키마를 저장하지 않으므로 시간이 걸릴 수 있습니다. AWS Glue Parquet 라이터를 사용하는 경우 사전 컴퓨팅된 스키마가 필요하지 않습니다. 데이터가 들어오면 라이터는 스키마를 동적으로 컴퓨팅 및 수정합니다.

useGlueParquetWriter를 지정하는 경우 다음 제한에 유의하세요.

- 라이터는 열 추가 또는 제거와 같은 스키마 개선만 지원하고 ResolveChoice와 같은 열 유형 변경은 지원하지 않습니다.

- 작성기는 빈 DataFrames 작성(예: 스키마 전용 파일 작성)을 지원하지 않습니다. `enableUpdateCatalog=True`를 설정하여 AWS Glue 데이터 카탈로그와 통합할 때 빈 DataFrame을 작성하려고 해도 데이터 카탈로그가 업데이트되지 않습니다. 그러면 스키마 없이 데이터 카탈로그에 테이블이 생성됩니다.

변형에 이러한 제한이 필요하지 않은 경우 AWS Glue Parquet 라이터를 켜면 성능이 향상됩니다.

AWS Glue에서 XML 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 XML 데이터 형식으로 저장 또는 전송되는 경우 이 문서에서는 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 XML 형식 사용을 지원합니다. 이 형식은 행이나 열을 기반으로 하지 않는 고도로 구성 가능하고 엄격하게 정의된 데이터 구조를 나타냅니다. XML은 고도로 표준화되어 있습니다. 표준 기관의 형식에 대한 소개는 XML 필수 사항을 참조하십시오.

AWS Glue를 사용하여 Amazon S3는 물론 XML 파일이 포함된 `bzip` 및 `gzip` 아카이브에서 XML 파일을 읽을 수 있습니다. 이 페이지에서 설명하는 구성 대신 S3 연결 파라미터에서 압축 동작을 구성할 수 있습니다.

다음 표에서는 XML 형식 옵션을 지원하는 일반적인 AWS Glue 기능을 보여줍니다.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크
지원	지원되지 않음	지원되지 않음	지원	지원

예: S3에서 XML 읽기

XML 리더는 XML 태그 이름을 사용합니다. 입력 내에 해당 태그가 있는 요소를 검사하여 스키마를 추론하고 DynamicFrame을 해당 값으로 채웁니다. AWS Glue XML 기능은 Apache Spark용 XML 데이터 소스와 비슷하게 작동합니다. 이 리더를 해당 프로젝트의 문서와 비교하면 기본 동작에 대한 통찰력을 얻을 수 있습니다.

사전 조건: 읽고자 하는 CSV 파일 또는 폴더에 대한 S3 경로(`s3path`), 그리고 XML 파일에 관한 약간의 정보가 필요합니다. 읽으려는 XML 요소에 대한 태그(`xmlTag`)도 필요합니다.

구성: 함수 옵션에서 `format="xml"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 리더와 S3가 상호 작용하는 방식을

추가로 구성할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션 참조: S3 연결 파라미터 `format_options`에서 `rowTag` 키를 사용하여 `xmlTag`를 지정합니다. 리더에서 `format_options`의 XML 파일을 해석하는 방법을 추가로 구성할 수 있습니다. 자세한 내용은 XML 구성 참조를 참조하십시오.

다음 AWS Glue ETL 스크립트는 S3에서 XML 파일 또는 폴더를 읽는 프로세스를 보여줍니다.

Python

이 예에서는 create_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Read XML from S3
# Set the rowTag option to configure the reader.

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="xml",
    format_options={"rowTag": "xmlTag"},
)
```

또한 스크립트(`pyspark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
dataFrame = spark.read\
    .format("xml")\
    .option("rowTag", "xmlTag")\
    .load("s3://s3path")
```

Scala

이 예에서는 getSourceWithFormat 작업을 사용합니다.

```
// Example: Read XML from S3
// Set the rowTag option to configure the reader.
```

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkSession

val glueContext = new GlueContext(SparkContext.getOrCreate())
val sparkSession: SparkSession = glueContext.getSparkSession

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"rowTag": "xmlTag"}"""),
      connectionType="s3",
      format="xml",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
}
```

또한 스크립트(`org.apache.spark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
val dataFrame = spark.read
  .option("rowTag", "xmlTag")
  .format("xml")
  .load("s3://s3path“)
```

XML 구성 참조

AWS Glue 라이브러리가 `format="xml"`를 지정한 곳이라면 어디에서든 다음 `format_options`을 사용할 수 있습니다.

- `rowTag` - 파일의 XML 태그를 지정하여 태그를 열로 취급합니다. 열 태그는 자기 복제를 할 수 없습니다.
  - 유형: 텍스트, 필수
- `encoding` - 문자 인코딩을 지정합니다. 런타임 환경에서 지원하는 Charset의 이름 또는 별칭이 될 수 있습니다. 인코딩 지원에 대해 구체적으로 보장하지는 않지만 주요 인코딩은 작동할 것입니다.
  - 유형: 텍스트, 기본값: `"UTF-8"`
- `excludeAttribute` - 요소에서 속성을 제외할지 여부를 지정합니다.
  - 유형: 부울, 기본값: `false`
- `treatEmptyValuesAsNulls` - 공백을 null 값으로 취급할지 여부를 지정합니다.

  - 유형: 부울, 기본값: `false`
- `attributePrefix` - 하위 요소 텍스트와 구별하기 위한 속성용 접두사. 이 접두사는 필드 이름으로 사용됩니다.
  - 유형: 텍스트, 기본값: `"_"`
- `valueTag` - 하위 요소가 없는 요소에 속성이 있는 값에 사용되는 태그.
  - 유형: 텍스트, 기본값: `"_VALUE"`
- `ignoreSurroundingSpaces` - 값을 둘러싸는 공백을 무시할지 여부를 지정합니다.
  - 유형: 부울, 기본값: `false`
- `withSchema` - 추론된 스키마를 재정의하려는 상황에서 예상되는 스키마를 포함합니다. 이 옵션을 사용하지 않으면 AWS Glue는 XML 데이터에서 스키마를 유추합니다.
  - 유형: 텍스트, 기본값: 해당 사항 없음
  - 값은 `StructType`을 나타내는 JSON 객체여야 합니다.

XML 스키마를 수동으로 지정

수동 XML 스키마 예

`withSchema` 포맷 옵션을 사용하여 XML 데이터에 대한 스키마를 지정하는 예입니다.

```
from awsglue.gluetypes import *

schema = StructType([
  Field("id", IntegerType()),
  Field("name", StringType()),
  Field("nested", StructType([
    Field("x", IntegerType()),
    Field("y", StringType()),
    Field("z", ChoiceType([IntegerType(), StringType()]))
  ]))
])

datasource0 = create_dynamic_frame_from_options(
    connection_type,
    connection_options={"paths": ["s3://xml_bucket/someprefix"]},
    format="xml",
    format_options={"withSchema": json.dumps(schema.jsonValue())},
    transformation_ctx = ""
)
```

AWS Glue에서 Avro 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 Avro 데이터 형식으로 저장 또는 전송되는 경우 이 문서에서는 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 Avro 형식 사용을 지원합니다. 이 형식은 성능 중심의 행 기반 데이터 형식입니다. 표준 기관의 형식에 대한 소개는 Apache Avro 1.8.2 설명서 개요를 참조하세요.

AWS Glue를 사용하여 Amazon S3와 스트리밍 소스에서 Avro 파일을 읽을 수 있을 뿐만 아니라 Amazon S3에 Avro 파일을 쓸 수 있습니다. S3에서 Avro 파일이 포함된 `bzip2` 및 `gzip` 아카이브를 읽고 쓸 수 있습니다. 또한 Avro 파일이 포함된 `deflate`, `snappy`, `xz` 아카이브를 쓸 수 있습니다. 이 페이지에서 설명하는 구성 대신 S3 연결 파라미터에서 압축 동작을 구성할 수 있습니다.

다음 표에서는 Avro 형식 옵션을 지원하는 일반적인 AWS Glue 기능을 보여 줍니다.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크
지원	지원	지원*	지원되지 않음	지원

*제한적으로 지원됩니다. 자세한 내용은 the section called “Avro 스트리밍 소스에 대한 참고 사항 및 제한 사항” 단원을 참조하십시오.

예: S3에서 Avro 파일 또는 폴더 읽기

사전 조건: 읽으려는 Avro 파일 또는 폴더에 대한 S3 경로(`s3path`)가 필요합니다.

구성: 함수 옵션에서 `format="avro"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 리더와 S3가 상호 작용하는 방식을 구성할 수 있습니다. 자세한 내용은 AWS Glue: the section called “S3 연결 파라미터”의 ETL 입력 및 출력에 대한 데이터 형식 옵션을 참조하세요. 리더에서 `format_options`의 Avro 파일을 해석하는 방법을 구성할 수 있습니다. 자세한 내용은 Avro 구성 참조를 참조하세요.

다음 AWS Glue ETL 스크립트는 S3에서 Avro 파일 또는 폴더를 읽는 프로세스를 보여 줍니다.

Python

이 예에서는 create_dynamic_frame.from_options 메서드를 사용합니다.

```
from pyspark.context import SparkContext
```

```
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="avro"
)
```

Scala

이 예에서는 getSourceWithFormat 작업을 사용합니다.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="avro",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
```

예: S3에 Avro 파일 및 폴더 쓰기

사전 조건: 초기화된 DataFrame(`dataFrame`) 또는 DynamicFrame(`dynamicFrame`)이 필요합니다. 예상되는 S3 출력 경로(`s3path`)도 필요합니다.

구성: 함수 옵션에서 `format="avro"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 라이터가 S3와 상호 작용하는 방식을 추가로 변경할 수 있습니다. 자세한 내용은 AWS Glue: the section called “S3 연결 파라미터”의 ETL 입력 및 출력에 대한 데이터 형식 옵션을 참조하세요. 작가가 `format_options`에서 Avro 파일을 해석하는 방식을 변경할 수 있습니다. 자세한 내용은 Avro 구성 참조를 참조하세요.

다음 AWS Glue ETL 스크립트는 S3로 Avro 파일 및 폴더를 쓰는 프로세스를 보여줍니다.

Python

이 예에서는 write_dynamic_frame.from_options 메서드를 사용합니다.

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

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="avro",
    connection_options={
        "path": "s3://s3path"
    }
)
```

Scala

이 예에서는 getSinkWithFormat 메서드를 사용합니다.

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

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
      connectionType="s3",
      options=JsonOptions("""{"path": "s3://s3path"}"""),
      format="avro"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

Avro 구성 참조

AWS Glue 라이브러리가 `format="avro"`를 지정한 곳이라면 어디에서든 다음 `format_options` 값을 사용할 수 있습니다.

- `version` - 지원하는 Apache Avro 리더/라이터 포맷의 버전을 지정합니다. 기본값은 "1.7"입니다. `format_options={"version": “1.8”}`를 지정해 Avro 논리적 유형 읽기 및 쓰기를 활성화할 수 있습니다. 자세한 내용은 Apache Avro 1.7.7 사양 및 Apache Avro 1.8.2 사양을 참조하십시오.

  Apache Avro 1.8 커넥터는 다음과 같은 논리적 유형 변환을 지원합니다.

리더의 경우: 다음 테이블은 Avro 리더 1.7 및 1.8을 위한 Avro 데이터 유형(논리적 유형, Avro 기본 유형)과 AWS Glue `DynamicFrame` 데이터 유형 간 변환을 보여줍니다.

Avro 데이터 형식: 논리적 유형	Avro 데이터 형식: Avro 기본 유형	GlueDynamicFrame 데이터 유형: Avro 리더 1.7	GlueDynamicFrame 데이터 유형: Avro 리더 1.8
10진수	bytes	BINARY	10진수
10진수	고정	BINARY	10진수
날짜	int	INT	날짜
시간(밀리초)	int	INT	INT
시간(마이크로초)	long	LONG	LONG
타임스탬프(밀리초)	long	LONG	Timestamp
타임스탬프(마이크로초)	long	LONG	LONG
기간(논리적 유형 아님)	고정(12)	BINARY	BINARY

라이터의 경우: 다음 테이블은 Avro 라이터 1.7 및 1.8을 위한 AWS Glue `DynamicFrame` 데이터 유형과 Avro 데이터 유형 간 변환을 보여줍니다.

AWS Glue **DynamicFrame** 데이터 유형	Avro 데이터 형식: Avro 라이터 1.7	Avro 데이터 형식: Avro 라이터 1.8
10진수	String	decimal
날짜	String	date
Timestamp	String	timestamp-micros

Avro Spark DataFrame 지원

Spark DataFrame API에서 Avro를 사용하려면 해당 Spark 버전에 대한 Spark Avro 플러그인을 설치해야 합니다. 작업에서 사용할 수 있는 Spark 버전은 AWS Glue 버전에 따라 결정됩니다. Spark 버전에 대한 자세한 내용은 the section called “AWS Glue 버전” 섹션을 참조하세요. 이 플러그인은 Apache에서 유지 관리하며 구체적인 지원을 보장하지는 않습니다.

AWS Glue 2.0에서 - Spark Avro 플러그인의 버전 2.4.3을 사용하세요. 이 JAR는 Maven Central에서 찾을 수 있습니다. org.apache.spark:spark-avro_2.12:2.4.3을 참조하세요.

AWS Glue 3.0에서 - Spark Avro 플러그인의 버전 3.1.1을 사용하세요. 이 JAR는 Maven Central에서 찾을 수 있습니다. org.apache.spark:spark-avro_2.12:3.1.1을 참조하세요.

추가 JAR을 AWS Glue ETL 작업에 포함하려면 `--extra-jars` 작업 파라미터를 사용하세요. 작업 파라미터에 대한 자세한 내용을 알아보려면 the section called “작업 파라미터” 섹션을 참조하세요. AWS Management Console에서 이 매개 변수를 구성할 수도 있습니다.

AWS Glue에서 grokLog 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 느슨하게 구조화된 일반 텍스트 형식으로 저장되거나 전송되는 경우 이 문서에서는 Grok 패턴을 통해 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 Grok 패턴 사용을 지원합니다. Grok 패턴은 정규 표현식 캡처 그룹과 유사합니다. 이들은 일반 텍스트 파일에서 문자 시퀀스의 패턴을 인식하여 유형과 목적을 부여합니다. AWS Glue에서 주요 목적은 로그를 읽는 것입니다. 저자의 Grok에 대한 소개는 Logstash 참조: Grok 필터 플러그인을 참조하세요.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크
지원	해당 사항 없음	지원	지원	지원되지 않음

grokLog 구성 참조

`format="grokLog"`으로 다음 `format_options` 값을 사용할 수 있습니다.

- `logFormat` - 로그 포맷과 일치하는 Grok 패턴을 지정합니다.
- `customPatterns` - 여기서 사용된 Grok 패턴을 지정합니다.
- `MISSING` - 누락된 값을 식별하는 데 사용된 신호를 지정합니다. 기본값은 `'-'`입니다.
- `LineCount` - 각 로그 기록의 라인 수를 지정합니다. 기본값은 `'1'`이며 현재 단일 라인 기록만 지원합니다.
- `StrictMode` - 제한 모드를 설정할지 여부를 지정하는 부울 값. 제한 모드에서 리더는 자동적으로 유형을 전환하거나 복구하지 않습니다. 기본값은 `"false"`입니다.

AWS Glue에서 Ion 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 Ion 데이터 형식으로 저장 또는 전송되는 경우 이 문서에서는 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 Ion 형식 사용을 지원합니다. 이 형식은 교환 가능한 이진 및 일반 텍스트 표현으로 데이터 구조(행 또는 열 기반이 아님)를 나타냅니다. 저자의 형식에 대한 소개는 Amazon Ion을 참조하세요. (자세한 내용은 Amazon Ion 상세 정보를 참조하십시오.)

AWS Glue를 사용하여 Amazon S3에서 Ion 파일을 읽을 수 있습니다. S3에서 Ion 파일이 포함된 `bzip` 및 `gzip` 아카이브를 읽을 수 있습니다. 이 페이지에서 설명하는 구성 대신 S3 연결 파라미터에서 압축 동작을 구성할 수 있습니다.

다음 표에서는 Ion 형식 옵션을 지원하는 일반적인 AWS Glue 기능을 보여 줍니다.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크
지원	지원되지 않음	지원되지 않음	지원	지원되지 않음

예: S3에서 Ion 파일 및 폴더 읽기

사전 조건: 읽으려는 Ion 파일 또는 폴더에 대한 S3 경로(s3path)가 필요합니다.

구성: 함수 옵션에서 format="json"를 지정합니다. connection_options에서 paths 키를 사용하여 s3path를 지정합니다. connection_options에서 리더와 S3가 상호 작용하는 방식을 구성할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션 참조: the section called “S3 연결 파라미터”

다음 AWS Glue ETL 스크립트는 S3에서 Ion 파일 또는 폴더를 읽는 프로세스를 보여 줍니다.

Python

이 예에서는 create_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Read ION from S3

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="ion"
)
```

Scala

이 예에서는 getSourceWithFormat 작업을 사용합니다.

```
// Example: Read ION from S3

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
```

```
    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="ion",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

Ion 구성 참조

`format="ion"`을 위한 `format_options` 값은 없습니다.

AWS Glue에서 JSON 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 JSON 데이터 형식으로 저장 또는 전송되는 경우 이 문서에서는 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 JSON 형식 사용을 지원합니다. 이 형식은 행이나 열을 기반으로 하지 않는 일관된 모양이지만 유연한 내용의 데이터 구조를 나타냅니다. JSON은 여러 기관에서 발행한 병렬 표준으로 정의되며 그 중 하나는 ECMA-404입니다. 일반적으로 참조되는 소스의 형식에 대한 소개는 JSON 소개를 참조하세요.

AWS Glue를 사용하여 Amazon S3에서 JSON 파일과 `bzip` 및 `gzip` 압축된 JSON 파일을 읽을 수 있습니다. 이 페이지에서 설명하는 구성 대신 S3 연결 파라미터에서 압축 동작을 구성할 수 있습니다.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크	
지원	지원	지원	지원	지원	

예: S3에서 JSON 파일 또는 폴더 읽기

사전 조건: 읽으려는 JSON 파일 또는 폴더에 대한 S3 경로(`s3path`)가 필요합니다.

구성: 함수 옵션에서 `format="json"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. 연결 옵션에서 읽기 작업이 s3를 통과하는 방법을 추가로 변경할 수 있습니다. 자세한 내용은 the section called “S3 연결 파라미터”를 참조하세요. 리더에서

format_options의 JSON 파일을 해석하는 방법을 구성할 수 있습니다. 자세한 내용은 JSON 구성 참조를 참조하세요.

다음 AWS Glue ETL 스크립트는 S3에서 JSON 파일 또는 폴더를 읽는 프로세스를 보여 줍니다.

Python

이 예에서는 create_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Read JSON from S3
# For show, we handle a nested JSON file that we can limit with the JsonPath
 parameter
# For show, we also handle a JSON where a single entry spans multiple lines
# Consider whether optimizePerformance is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="json",
    format_options={
        "jsonPath": "$.id",
        "multiline": True,
        # "optimizePerformance": True, -> not compatible with jsonPath, multiline
    }
)
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
dataFrame = spark.read\
    .option("multiline", "true")\
    .json("s3://s3path")
```

Scala

이 예에서는 getSourceWithFormat 작업을 사용합니다.

```
// Example: Read JSON from S3
// For show, we handle a nested JSON file that we can limit with the JsonPath
 parameter
// For show, we also handle a JSON where a single entry spans multiple lines
// Consider whether optimizePerformance is right for your workflow.

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

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"jsonPath": "$.id", "multiline": true,
 "optimizePerformance":false}"""),
      connectionType="s3",
      format="json",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

또한 스크립트(`pyspark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
val dataFrame = spark.read
    .option("multiline", "true")
    .json("s3://s3path")
```

예: S3에 JSON 파일 및 폴더 쓰기

사전 조건: 초기화된 DataFrame(`dataFrame`) 또는 DynamicFrame(`dynamicFrame`)이 필요합니다. 예상되는 S3 출력 경로(`s3path`)도 필요합니다.

구성: 함수 옵션에서 `format="json"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 라이터가 S3와 상호 작용하는 방식을 추가로 변경할 수 있습니다. 자세한 내용은 AWS Glue: the section called “S3 연결 파라미터”의 ETL 입력 및 출력에 대한 데이터 형식 옵션을 참조하세요. 작성기에서 `format_options`의 JSON 파일을 해석하는 방법을 구성할 수 있습니다. 자세한 내용은 JSON 구성 참조를 참조하세요.

다음 AWS Glue ETL 스크립트는 S3에서 JSON 파일 또는 폴더를 쓰는 프로세스를 보여 줍니다.

Python

이 예에서는 write_dynamic_frame.from_options 메서드를 사용합니다.

```
# Example: Write JSON to S3

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    connection_options={"path": "s3://s3path"},
    format="json"
)
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
df.write.json("s3://s3path/")
```

Scala

이 예에서는 getSinkWithFormat 메서드를 사용합니다.

```
// Example: Write JSON to S3

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

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
```

```
        format="json"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
df.write.json("s3://s3path")
```

JSON 구성 참조

format="json"으로 다음 format_options 값을 사용할 수 있습니다.

- jsonPath - 레코드로 읽을 객체를 식별하는 JsonPath 표현식. 이것은 외부 배열 안에 중첩된 레코드가 파일에 있을 때 특히 유용합니다. 예를 들어, 다음 JsonPath 표현식은 JSON 객체의 id 필드를 대상으로 합니다.

```
format="json", format_options={"jsonPath": "$.id"}
```

- multiline - 단일 기록이 다양한 라인을 포괄할 수 있는지 여부를 지정하는 부울 값. 필드가 인용된 새로운 라인 문자를 포함할 때 발생합니다. 이 옵션을 "true"로 설정해야 기록이 여러 라인을 포괄할 수 있습니다. 기본값은 "false"이라서 파싱 동안 더 많은 공격적 파일 쪼개기가 가능합니다.
- optimizePerformance - Apache Arrow 기반 열 포맷 메모리 포맷과 함께 고급 SIMD JSON 리더를 사용할지 여부를 지정하는 부울 값입니다. AWS Glue 3.0에서만 사용 가능합니다. multiline 또는 jsonPath와 호환되지 않습니다. 이러한 옵션 중 하나를 제공하면 AWS Glue에서 표준 리더로 돌아갑니다.
- withSchema - the section called “XML 스키마 지정”에 설명된 형식으로 테이블 스키마를 지정하는 문자열 값입니다. 비카탈로그 연결에서 읽을 때 optimizePerformance에만 사용됩니다.

Apache Arrow 열 포맷으로 벡터화된 SIMD JSON 리더 사용

AWS Glue 버전 3.0은 JSON 데이터에 대한 벡터화된 리더를 추가합니다. 이 리더는 특정 조건에서 표준 리더에 비해 2배 더 빠른 성능을 발휘합니다. 이 리더에는 사용하기 전에 사용자가 알아야 할 특정 제한 사항이 있습니다(이 섹션에 설명되어 있음).

최적화된 리더를 사용하려면 format_options 또는 테이블 속성에서 "optimizePerformance"를 True로 설정합니다. 또한 카탈로그에서 읽지 않는 한 withSchema를 제공해야 합니다. withSchema는 the section called “XML 스키마 지정”에 설명된 대로 입력을 기대합니다.

```
// Read from S3 data source
glueContext.create_dynamic_frame.from_options(
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path"]},
    format = "json",
    format_options={
        "optimizePerformance": True,
        "withSchema": SchemaString
        })

// Read from catalog table
glueContext.create_dynamic_frame.from_catalog(
    database = database,
    table_name = table,
    additional_options = {
    // The vectorized reader for JSON can read your schema from a catalog table
 property.
        "optimizePerformance": True,
        })
```

AWS Glue 라이브러리에서 *SchemaString* 구축에 대한 자세한 내용은 the section called “타입” 섹션을 참조하세요.

벡터화된 CSV 리더에 대한 제한 사항

다음과 같은 제한 사항이 있습니다.

- 중첩된 객체 또는 배열 값이 있는 JSON 요소는 지원되지 않습니다. 제공 시 AWS Glue는 표준 리더로 돌아갑니다.
- 카탈로그에서 또는 withSchema를 사용하여 스키마를 제공해야 합니다.
- multiline 또는 jsonPath와 호환되지 않습니다. 이러한 옵션 중 하나를 제공하면 AWS Glue에서 표준 리더로 돌아갑니다.
- 입력 스키마와 일치하지 않는 입력 레코드를 제공하면 리더가 실패할 수 있습니다.
- 오류 레코드는 생성되지 않습니다.
- 일본어 또는 중국어와 같은 멀티바이트 문자가 포함된 JSON 파일은 지원되지 않습니다.

AWS Glue에서 ORC 형식 사용

AWS Glue는 소스에서 데이터를 검색하고 다양한 데이터 형식으로 저장 및 전송되는 대상에 데이터를 씁니다. 데이터가 ORC 데이터 형식으로 저장 또는 전송되는 경우 이 문서에서는 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능을 소개합니다.

AWS Glue는 ORC 형식 사용을 지원합니다. 이 형식은 성능 중심의 열 기반 데이터 형식입니다. 표준 기관의 형식에 대한 소개는 Apache Orc를 참조하세요.

AWS Glue를 사용하여 Amazon S3와 스트리밍 소스에서 ORC 파일을 읽을 수 있을 뿐만 아니라 Amazon S3에 ORC 파일을 쓸 수 있습니다. S3에서 ORC 파일이 포함된 `bzip` 및 `gzip` 아카이브를 읽고 쓸 수 있습니다. 이 페이지에서 설명하는 구성 대신 S3 연결 파라미터에서 압축 동작을 구성할 수 있습니다.

다음 표에서는 ORC 형식 옵션을 지원하는 일반적인 AWS Glue 기능을 보여 줍니다.

읽기	쓰기	스트리밍 읽기	작은 파일 그룹화	작업 북마크
지원	지원	지원	지원되지 않음	지원*

* AWS Glue 버전 1.0 이상에서 지원

예: S3에서 ORC 파일 또는 폴더 읽기

사전 조건: 읽으려는 ORC 파일 또는 폴더에 대한 S3 경로(`s3path`)가 필요합니다.

구성: 함수 옵션에서 `format="orc"`를 지정합니다. `connection_options`에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 리더와 S3가 상호 작용하는 방식을 구성할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션 참조: the section called “S3 연결 파라미터”

다음 AWS Glue ETL 스크립트는 S3에서 ORC 파일 또는 폴더를 읽는 프로세스를 보여 줍니다.

Python

이 예에서는 create_dynamic_frame.from_options 메서드를 사용합니다.

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

```
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="orc"
)
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
dataFrame = spark.read\
    .orc("s3://s3path")
```

Scala

이 예에서는 getSourceWithFormat 작업을 사용합니다.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="orc",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
}
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
val dataFrame = spark.read
    .orc("s3://s3path")
```

예: S3에 ORC 파일 및 폴더 쓰기

사전 조건: 초기화된 DataFrame(`dataFrame`) 또는 DynamicFrame(`dynamicFrame`)이 필요합니다. 예상되는 S3 출력 경로(`s3path`)도 필요합니다.

구성: 함수 옵션에서 `format="orc"`를 지정합니다. 연결 옵션에서 `paths` 키를 사용하여 `s3path`를 지정합니다. `connection_options`에서 라이터가 S3와 상호 작용하는 방식을 추가로 변경할 수 있습니다. 자세한 내용은 AWS Glue: the section called “S3 연결 파라미터”의 ETL 입력 및 출력에 대한 데이터 형식 옵션을 참조하세요. 다음 코드 예제는 프로세스를 보여 줍니다.

Python

이 예에서는 write_dynamic_frame.from_options 메서드를 사용합니다.

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

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="orc",
    connection_options={
        "path": "s3://s3path"
    }
)
```

또한 스크립트(`pyspark.sql.DataFrame`)에서 DataFrame을 사용합니다.

```
df.write.orc("s3://s3path/")
```

Scala

이 예에서는 getSinkWithFormat 메서드를 사용합니다.

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

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
      connectionType="s3",
      options=JsonOptions("""{"path": "s3://s3path"}"""),
      format="orc"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

또한 스크립트(pyspark.sql.DataFrame)에서 DataFrame을 사용합니다.

```
df.write.orc("s3://s3path/")
```

XML 구성 참조

format="orc"을 위한 format_options 값은 없습니다. 그러나 SparkSQL 기초 코드에 의해 수락된 옵션은 connection_options 맵 파라미터를 거쳐 SparkSQL 기초 코드로 넘겨질 수 있습니다.

AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용

오픈 소스 데이터 레이크 프레임워크는 Amazon S3에 빌드된 데이터 레이크에 저장된 파일의 증분 데이터 처리를 간소화합니다. AWS Glue 3.0 이상에서는 다음과 같은 오픈 소스 데이터 레이크 프레임워크를 지원합니다.

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

Amazon S3에 저장된 데이터를 트랜잭션적으로 일관된 방식으로 읽고 쓸 수 있도록 이러한 프레임워크에 대한 기본 지원을 제공합니다. AWS Glue ETL 작업에 이러한 프레임워크를 사용하기 위해 별도의 커넥터를 설치하거나 추가 구성 단계를 완료할 필요가 없습니다.

AWS Glue Data Catalog를 통해 데이터 세트를 관리하는 경우 Spark DataFrames에서 AWS Glue 메서드를 사용하여 데이터 레이크 테이블을 읽고 쓸 수 있습니다. Spark DataFrame API를 사용하여 Amazon S3 데이터를 읽고 쓸 수도 있습니다.

이 비디오에서는 Apache Hudi, Apache Iceberg 및 Delta Lake work의 작동 방식에 대한 기본 사항을 살펴봅니다. 데이터 레이크에서 데이터를 삽입, 업데이트 및 삭제하는 방법과 각 프레임워크의 작동 방식을 확인할 수 있습니다.

주제

- 제한 사항
- AWS Glue에서 Hudi 프레임워크 사용
- AWS Glue에서 Delta Lake 프레임워크 사용
- AWS Glue에서 Iceberg 프레임워크 사용

제한 사항

AWS Glue에서 데이터 레이크 프레임워크를 사용하기 전에 다음 제한 사항을 고려하세요.

- 다음 DynamicFrame의 AWS Glue `GlueContext` 방법은 데이터 레이크 프레임워크 테이블의 읽기 및 쓰기를 지원하지 않습니다. 대신 DataFrame 또는 Spark DataFrame API용 `GlueContext` 방법을 사용하십시오.
  - `create_dynamic_frame.from_catalog`
  - `write_dynamic_frame.from_catalog`
  - `getDynamicFrame`
  - `writeDynamicFrame`
- Lake Formation 권한 제어에서는 다음과 같은 DataFrame `GlueContext` 방법이 지원됩니다.
  - `create_data_frame.from_catalog`
  - `write_data_frame.from_catalog`
  - `getDataFrame`
  - `writeDataFrame`
- 작은 파일 그룹화는 지원되지 않습니다.
- 작업 북마크는 지원되지 않습니다.
- Apache Hudi 0.10.1 for AWS Glue 3.0은 Hudi 읽을 때 병합(MoR) 테이블을 지원하지 않습니다.
- `ALTER TABLE … RENAME TO`는 Apache Iceberg 0.13.1 for AWS Glue 3.0에서 사용할 수 없습니다.

Lake Formation 권한으로 관리되는 데이터 레이크 형식 테이블에 대한 제한

데이터 레이크 형식은 Lake Formation 권한을 통해 AWS Glue ETL과 통합됩니다. `create_dynamic_frame`를 사용하여 DynamicFrame을 생성하는 것은 지원되지 않습니다. 자세한 내용은 다음 예를 참조하세요.

- 예: Lake Formation 권한 제어가 포함된 Iceberg 테이블 읽기 및 쓰기
- 예: Lake Formation 권한 제어를 사용하여 Hudi 테이블 읽기 및 쓰기
- 예: Lake Formation 권한 제어가 포함된 Delta Lake 테이블 읽기 및 쓰기

> **Note**
>
> Apache Hudi, Apache Iceberg, Delta Lake에 대한 Lake Formation 권한을 통한 AWS Glue ETL과의 통합은 AWS Glue 버전 4.0에서만 지원됩니다.

Apache Iceberg는 Lake Formation 권한을 통해 AWS Glue ETL과 가장 잘 통합됩니다. 거의 모든 작업을 지원하며 SQL 지원을 포함합니다.

Hudi는 관리 작업을 제외한 대부분의 기본 작업을 지원합니다. 이러한 옵션은 일반적으로 데이터 프레임 작성을 통해 수행되고 `additional_options`를 통해 지정되기 때문입니다. SparkSQL은 지원되지 않으므로 AWS Glue API를 사용하여 작업을 위한 DataFrames을 생성해야 합니다.

Delta Lake는 테이블 데이터의 읽기, 추가, 덮어쓰기만 지원합니다. Delta Lake에서는 업데이트와 같은 다양한 작업을 수행하려면 자체 라이브러리를 사용해야 합니다.

Lake Formation 권한으로 관리되는 Iceberg 테이블에는 다음 기능을 사용할 수 없습니다.

- AWS Glue ETL을 사용한 압축
- AWS Glue ETL을 통한 Spark SQL 지원

Lake Formation 권한으로 관리되는 Hudi 테이블의 제한 사항은 다음과 같습니다.

- 분리된 파일 제거

Lake Formation 권한으로 관리되는 Delta Lake 테이블의 제한 사항은 다음과 같습니다.

- Delta Lake 테이블에서 삽입 및 읽기 이외의 모든 기능.

AWS Glue에서 Hudi 프레임워크 사용

AWS Glue 3.0 이상에서는 데이터 레이크에 대한 Apache Hudi 프레임워크를 지원합니다. Hudi는 증분 데이터 처리 및 데이터 파이프라인 개발을 간소화하는 오픈 소스 데이터 레이크 스토리지 프레임워크입니다. 이 항목에서는 Hudi 테이블에서 데이터를 전송하거나 저장할 때 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능에 대해 설명합니다. Hudi에 대한 자세한 내용은 공식 Apache Hudi 설명서를 참조하세요.

AWS Glue를 사용하여 Amazon S3의 Hudi 테이블에서 읽기 및 쓰기 작업을 수행하거나 AWS Glue 데이터 카탈로그를 사용하여 Hudi 테이블로 작업할 수 있습니다. 삽입, 업데이트, 모든 Apache Spark 작업을 포함한 추가 작업도 지원됩니다.

> **Note**
>
> AWS Glue 5.0의 Apache Hudi 0.15.0 구현은 HUDI-7001을 내부적으로 되돌립니다. 레코드 키가 단일 필드로 구성된 경우 복합 키 생성과 관련된 회귀를 나타내지 않습니다. 그러나 이 동작은 OSS Apache Hudi 0.15.0과 다릅니다.
> Apache Hudi 0.10.1 for AWS Glue 3.0은 Hudi 읽을 때 병합(MoR) 테이블을 지원하지 않습니다.

다음 표에는 각 AWS Glue 버전에 포함된 Hudi 버전이 나와 있습니다.

AWS Glue 버전	지원되는 Hudi 버전
5.0	0.15.0
4.0	0.12.1
3.0	0.10.1

AWS Glue가 지원하는 데이터 레이크 프레임워크에 대한 자세한 내용은 AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용 섹션을 참조하세요.

Hudi 활성화

Hudi for AWS Glue를 활성화하려면 다음 작업을 완료합니다.

- hudi를 `--datalake-formats` 작업 파라미터의 값으로 지정합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용 섹션을 참조하세요.
- AWS Glue 작업에 대한 `--conf` 키를 생성하고 다음 값으로 설정합니다. 또는 스크립트에서 `SparkConf`를 사용하여 다음 구성을 설정할 수 있습니다. 이러한 설정은 Apache Spark에서 Hudi 테이블을 올바르게 처리하는 데 도움이 됩니다.

```
spark.serializer=org.apache.spark.serializer.KryoSerializer
```

- Hudi에 대한 Lake Formation 권한 지원은 AWS Glue 4.0에서 기본으로 활성화되어 있습니다. Lake Formation에 등록된 Hudi 테이블을 읽고 쓰는 데 추가 구성이 필요하지 않습니다. 등록된 Hudi 테이블을 읽으려면 AWS Glue 작업 IAM 역할에 SELECT 권한이 있어야 합니다. 등록된 Hudi 테이블에 글을 쓰려면 AWS Glue 작업 IAM 역할에 SUPER 권한이 있어야 합니다. Lake Formation 권한 관리에 대해 자세히 알아보려면 Data Catalog 리소스에 대한 권한 부여 및 취소를 참조하십시오.

다른 Hudi 버전 사용

AWS Glue에서 지원하지 않는 Hudi 버전을 사용하려면 `--extra-jars` 작업 파라미터를 사용하여 고유한 Hudi JAR 파일을 지정하세요. `--datalake-formats` 작업 파라미터의 값으로 hudi를 포함하지 마세요. AWS Glue 5.0을 사용하는 경우, `--user-jars-first true` 작업 파라미터를 설정해야 합니다.

예: Amazon S3에 Hudi 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록

이 예제 스크립트는 Amazon S3에 Hudi 테이블을 작성하고 AWS Glue 데이터 카탈로그에 테이블을 등록하는 방법을 보여줍니다. 이 예제에서는 Hudi Hive Sync 도구를 사용하여 테이블을 등록합니다.

> **Note**
>
> 이 예에서는 AWS Glue 데이터 카탈로그를 Apache Spark Hive 메타스토어로 사용하기 위해 `--enable-glue-datacatalog` 작업 파라미터를 설정해야 합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용를 참조하세요.

Python

```
# Example: Create a Hudi table from a DataFrame
# and register the table to Glue Data Catalog
```

```
additional_options={
    "hoodie.table.name": "<your_table_name>",
    "hoodie.database.name": "<your_database_name>",
    "hoodie.datasource.write.storage.type": "COPY_ON_WRITE",
    "hoodie.datasource.write.operation": "upsert",
    "hoodie.datasource.write.recordkey.field": "<your_recordkey_field>",
    "hoodie.datasource.write.precombine.field": "<your_precombine_field>",
    "hoodie.datasource.write.partitionpath.field": "<your_partitionkey_field>",
    "hoodie.datasource.write.hive_style_partitioning": "true",
    "hoodie.datasource.hive_sync.enable": "true",
    "hoodie.datasource.hive_sync.database": "<your_database_name>",
    "hoodie.datasource.hive_sync.table": "<your_table_name>",
    "hoodie.datasource.hive_sync.partition_fields": "<your_partitionkey_field>",
    "hoodie.datasource.hive_sync.partition_extractor_class":
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
    "hoodie.datasource.hive_sync.use_jdbc": "false",
    "hoodie.datasource.hive_sync.mode": "hms",
    "path": "s3://<s3Path/>"
}

dataFrame.write.format("hudi") \
    .options(**additional_options) \
    .mode("overwrite") \
    .save()
```

Scala

```
// Example: Example: Create a Hudi table from a DataFrame
// and register the table to Glue Data Catalog

val additionalOptions = Map(
  "hoodie.table.name" -> "<your_table_name>",
  "hoodie.database.name" -> "<your_database_name>",
  "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
  "hoodie.datasource.write.operation" -> "upsert",
  "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
  "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
  "hoodie.datasource.write.partitionpath.field" -> "<your_partitionkey_field>",
  "hoodie.datasource.write.hive_style_partitioning" -> "true",
  "hoodie.datasource.hive_sync.enable" -> "true",
  "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
  "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
  "hoodie.datasource.hive_sync.partition_fields" -> "<your_partitionkey_field>",
```

```
  "hoodie.datasource.hive_sync.partition_extractor_class" ->
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
  "hoodie.datasource.hive_sync.use_jdbc" -> "false",
  "hoodie.datasource.hive_sync.mode" -> "hms",
  "path" -> "s3://<s3Path/>")

dataFrame.write.format("hudi")
  .options(additionalOptions)
  .mode("append")
  .save()
```

예: AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에서 Hudi 테이블 읽기

이 예에서는 Amazon S3의 예: Amazon S3에 Hudi 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록에서 생성한 Hudi 테이블을 읽습니다.

> **Note**
>
> 이 예에서는 AWS Glue 데이터 카탈로그를 Apache Spark Hive 메타스토어로 사용하기 위해 `--enable-glue-datacatalog` 작업 파라미터를 설정해야 합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용를 참조하세요.

Python

이 예에서는 `GlueContext.create_data_frame.from_catalog()` 메서드를 사용합니다.

```
# Example: Read a Hudi table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

dataFrame = glueContext.create_data_frame.from_catalog(
    database = "<your_database_name>",
    table_name = "<your_table_name>"
)
```

Scala

이 예에서는 getCatalogSource 메서드를 사용합니다.

```
// Example: Read a Hudi table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dataFrame = glueContext.getCatalogSource(
      database = "<your_database_name>",
      tableName = "<your_table_name>"
    ).getDataFrame()
  }
}
```

예: Amazon S3의 Hudi 테이블에서 **DataFrame** 업데이트 및 삽입

이 예제에서는 AWS Glue 데이터 카탈로그를 사용하여 예: Amazon S3에 Hudi 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록에서 생성한 Hudi 테이블에 DataFrame을 삽입합니다.

> **Note**
>
> 이 예에서는 AWS Glue 데이터 카탈로그를 Apache Spark Hive 메타스토어로 사용하기 위해 `--enable-glue-datacatalog` 작업 파라미터를 설정해야 합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용를 참조하세요.

Python

이 예에서는 `GlueContext.write_data_frame.from_catalog()` 메서드를 사용합니다.

```
# Example: Upsert a Hudi table from Glue Data Catalog

from awsglue.context import GlueContext
```

```
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame = dataFrame,
    database = "<your_database_name>",
    table_name = "<your_table_name>",
    additional_options={
        "hoodie.table.name": "<your_table_name>",
        "hoodie.database.name": "<your_database_name>",
        "hoodie.datasource.write.storage.type": "COPY_ON_WRITE",
        "hoodie.datasource.write.operation": "upsert",
        "hoodie.datasource.write.recordkey.field": "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field": "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field": "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning": "true",
        "hoodie.datasource.hive_sync.enable": "true",
        "hoodie.datasource.hive_sync.database": "<your_database_name>",
        "hoodie.datasource.hive_sync.table": "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields": "<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class":
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc": "false",
        "hoodie.datasource.hive_sync.mode": "hms"
    }
)
```

Scala

이 예에서는 getCatalogSink 메서드를 사용합니다.

```
// Example: Upsert a Hudi table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.JsonOptions
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
```

```
      additionalOptions = JsonOptions(Map(
        "hoodie.table.name" -> "<your_table_name>",
        "hoodie.database.name" -> "<your_database_name>",
        "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
        "hoodie.datasource.write.operation" -> "upsert",
        "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field" ->
 "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning" -> "true",
        "hoodie.datasource.hive_sync.enable" -> "true",
        "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
        "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields" ->
 "<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class" ->
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc" -> "false",
        "hoodie.datasource.hive_sync.mode" -> "hms"
      )))
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

예: Spark를 사용하여 Amazon S3에서 Hudi 테이블 읽기

이 예에서는 Spark DataFrame API를 사용하여 Amazon S3에서 Hudi 테이블을 읽습니다.

Python

```
# Example: Read a Hudi table from S3 using a Spark DataFrame

dataFrame = spark.read.format("hudi").load("s3://<s3path/>")
```

Scala

```
// Example: Read a Hudi table from S3 using a Spark DataFrame

val dataFrame = spark.read.format("hudi").load("s3://<s3path/>")
```

예: Spark를 사용하여 Amazon S3에 Hudi 테이블 쓰기

이 예에서는 Spark를 사용하여 Amazon S3에 Hudi 테이블을 씁니다.

Python

```
# Example: Write a Hudi table to S3 using a Spark DataFrame

dataFrame.write.format("hudi") \
    .options(**additional_options) \
    .mode("overwrite") \
    .save("s3://<s3Path/>")
```

Scala

```
// Example: Write a Hudi table to S3 using a Spark DataFrame

dataFrame.write.format("hudi")
  .options(additionalOptions)
  .mode("overwrite")
  .save("s3://<s3path/>")
```

예: Lake Formation 권한 제어를 사용하여 Hudi 테이블 읽기 및 쓰기

이 예제에서는 Lake Formation 권한 제어가 있는 Hudi 테이블을 읽고 씁니다.

1. Hudi 테이블을 생성하여 Lake Formation에 등록합니다.

   a. Lake Formation 권한 제어를 활성화하려면 먼저 Lake Formation에 Amazon S3 경로를 등록해야 합니다. 자세한 내용을 알아보려면 Registering an Amazon S3 location(Amazon S3 위치 등록)을 참조하세요. Lake Formation 콘솔에서 등록하거나 AWS CLI를 사용하여 등록할 수 있습니다.

   ```
   aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-
   folder> --use-service-linked-role --region <REGION>
   ```

   Amazon S3 위치를 등록하면 해당 위치(또는 하위 위치)를 가리키는 AWS Glue 테이블이 `GetTable` 호출 시 `IsRegisteredWithLakeFormation` 파라미터 값을 true로 반환합니다.

   b. Spark 데이터프레임 API를 통해 등록된 Amazon S3 경로를 가리키는 Hudi 테이블을 생성합니다.

   ```
   hudi_options = {
   ```

```
    'hoodie.table.name': table_name,
    'hoodie.database.name': database_name,
    'hoodie.datasource.write.storage.type': 'COPY_ON_WRITE',
    'hoodie.datasource.write.recordkey.field': 'product_id',
    'hoodie.datasource.write.table.name': table_name,
    'hoodie.datasource.write.operation': 'upsert',
    'hoodie.datasource.write.precombine.field': 'updated_at',
    'hoodie.datasource.write.hive_style_partitioning': 'true',
    'hoodie.upsert.shuffle.parallelism': 2,
    'hoodie.insert.shuffle.parallelism': 2,
    'path': <S3_TABLE_LOCATION>,
    'hoodie.datasource.hive_sync.enable': 'true',
    'hoodie.datasource.hive_sync.database': database_name,
    'hoodie.datasource.hive_sync.table': table_name,
    'hoodie.datasource.hive_sync.use_jdbc': 'false',
    'hoodie.datasource.hive_sync.mode': 'hms'
}

df_products.write.format("hudi")  \
    .options(**hudi_options)  \
    .mode("overwrite")  \
    .save()
```

2. AWSGlue 작업 IAM 역할에 Lake Formation 권한을 부여하십시오. Lake Formation 콘솔에서 권한을 부여하거나 AWS CLI를 사용하여 권한을 부여할 수 있습니다. 자세한 내용을 알아보려면 Lake Formation 콘솔 및 명명된 리소스 방법을 사용하여 데이터베이스 권한 부여를 참조하십시오

3. Lake Formation에 등록된 Hudi 테이블을 읽어보십시오. 코드는 등록되지 않은 Hudi 테이블을 읽는 것과 같습니다. 읽기에 성공하려면 AWS Glue 작업 IAM 역할에 SELECT 권한이 있어야 한다는 점에 유의하십시오.

```
val dataFrame = glueContext.getCatalogSource(
     database = "<your_database_name>",
     tableName = "<your_table_name>"
   ).getDataFrame()
```

4. Lake Formation에 등록된 Hudi 테이블에 글을 작성하십시오. 코드는 등록되지 않은 Hudi 테이블에 쓰는 것과 같습니다. 참고로 AWS Glue 작업 IAM 역할에 SUPER 권한이 있어야 쓰기가 성공합니다.

```
glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = JsonOptions(Map(
        "hoodie.table.name" -> "<your_table_name>",
        "hoodie.database.name" -> "<your_database_name>",
```

```
        "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
        "hoodie.datasource.write.operation" -> "<write_operation>",
        "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field" -> "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning" -> "true",
        "hoodie.datasource.hive_sync.enable" -> "true",
        "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
        "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields" ->
"<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class" ->
"org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc" -> "false",
        "hoodie.datasource.hive_sync.mode" -> "hms"
      )))
      .writeDataFrame(dataFrame, glueContext)
```

AWS Glue에서 Delta Lake 프레임워크 사용

AWS Glue 3.0 이상은 Linux Foundation Delta Lake 프레임워크를 지원합니다. Delta Lake는 ACID 트랜잭션을 수행하고, 메타데이터 처리를 확장하고, 스트리밍 및 배치 데이터 처리를 통합하는 데 도움이 되는 오픈 소스 데이터 레이크 스토리지 프레임워크입니다. 이 항목에서는 Delta Lake 테이블에서 데이터를 전송하거나 저장할 때 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능에 대해 설명합니다. Delta Lake에 대한 자세한 내용은 공식 Delta Lake 설명서를 참조하세요.

AWS Glue를 사용하여 Amazon S3의 Delta Lake 테이블에서 읽기 및 쓰기 작업을 수행하거나 AWS Glue 데이터 카탈로그를 사용하여 Delta Lake 테이블로 작업할 수 있습니다. 삽입, 업데이트, 테이블 배치 읽기 및 쓰기와 같은 추가 작업도 지원됩니다. Delta Lake 테이블을 사용할 경우 Delta Lake Python 라이브러리(예: `DeltaTable.forPath`)의 메서드를 사용할 수도 있습니다. Delta Lake Python 라이브러리에 대한 자세한 내용은 Delta Lake의 Python 설명서를 참조하세요.

다음 표에는 각 AWS Glue 버전에 포함된 Delta Lake의 버전이 나와 있습니다.

AWS Glue 버전	지원되는 Delta Lake 버전
5.0	3.3.0
4.0	2.1.0

AWS Glue 버전	지원되는 Delta Lake 버전
3.0	1.0.0

AWS Glue가 지원하는 데이터 레이크 프레임워크에 대한 자세한 내용은 AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용 섹션을 참조하세요.

Delta Lake for AWS Glue 활성화

Delta Lake for AWS Glue를 활성화하려면 다음 작업을 완료합니다.

- `delta`를 `--datalake-formats` 작업 파라미터의 값으로 지정합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용 섹션을 참조하세요.
- AWS Glue 작업에 대한 `--conf` 키를 생성하고 다음 값으로 설정합니다. 또는 스크립트에서 `SparkConf`를 사용하여 다음 구성을 설정할 수 있습니다. 이러한 설정은 Apache Spark에서 Delta Lake 테이블을 올바르게 처리하는 데 도움이 됩니다.

```
spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog --
conf
 spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDriverLogStore
```

- Delta 테이블에 대한 Lake Formation 권한 지원은 AWS Glue 4.0에서 기본적으로 활성화되어 있습니다. Lake Formation에 등록된 Delta 테이블을 읽고 쓰는 데 추가 구성이 필요하지 않습니다. 등록된 Delta 테이블을 읽으려면 AWS Glue 작업 IAM 역할에 SELECT 권한이 있어야 합니다. 등록된 Delta 테이블에 쓰려면 AWS Glue 작업 IAM 역할에 SUPER 권한이 있어야 합니다. Lake Formation 권한 관리에 대해 자세히 알아보려면 Data Catalog 리소스에 대한 권한 부여 및 취소를 참조하십시오.

다른 Delta Lake 버전 사용

AWS Glue에서 지원하지 않는 Delta Lake 버전을 사용하려면 `--extra-jars` 작업 파라미터를 사용하여 고유한 Delta Lake JAR 파일을 지정하세요. `--datalake-formats` 작업 파라미터의 값으로 `delta`를 포함하지 마세요. AWS Glue 5.0을 사용하는 경우, `--user-jars-first true` 작업 파라미터를 설정해야 합니다. 이 경우에 Delta Lake Python 라이브러리를 사용하려면 `--extra-py-files` 작업 파라미터를 사용하여 라이브러리 JAR 파일을 지정해야 합니다. Python 라이브러리는 Delta Lake JAR 파일에 패키징되어 제공됩니다.

예: Amazon S3에 Delta Lake 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록

다음 AWS Glue ETL 스크립트는 Amazon S3에 Delta Lake 테이블을 작성하고 AWS Glue 데이터 카탈로그에 테이블을 등록하는 방법을 보여줍니다.

Python

```
# Example: Create a Delta Lake table from a DataFrame
# and register the table to Glue Data Catalog

additional_options = {
    "path": "s3://<s3Path>"
}
dataFrame.write \
    .format("delta") \
    .options(**additional_options) \
    .mode("append") \
    .partitionBy("<your_partitionkey_field>") \
    .saveAsTable("<your_database_name>.<your_table_name>")
```

Scala

```
// Example: Example: Create a Delta Lake table from a DataFrame
// and register the table to Glue Data Catalog

val additional_options = Map(
  "path" -> "s3://<s3Path>"
)
dataFrame.write.format("delta")
  .options(additional_options)
  .mode("append")
  .partitionBy("<your_partitionkey_field>")
  .saveAsTable("<your_database_name>.<your_table_name>")
```

예: AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에서 Delta Lake 테이블 읽기

다음 AWS Glue ETL 스크립트는 예: Amazon S3에 Delta Lake 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록에서 생성한 Delta Lake 테이블을 읽습니다.

Python

이 예에서는 create_data_frame.from_catalog 메서드를 사용합니다.

```
# Example: Read a Delta Lake table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

이 예에서는 getCatalogSource 메서드를 사용합니다.

```
// Example: Read a Delta Lake table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val df = glueContext.getCatalogSource("<your_database_name>",
 "<your_table_name>",
      additionalOptions = additionalOptions)
      .getDataFrame()
  }
}
```

예: AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에서 Delta Lake 테이블에 **`DataFrame`** 삽입

이 예에서는 예: Amazon S3에 Delta Lake 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록에서 생성한 Delta Lake 테이블에 데이터를 삽입합니다.

> **Note**
>
> 이 예에서는 AWS Glue 데이터 카탈로그를 Apache Spark Hive 메타스토어로 사용하기 위해 `--enable-glue-datacatalog` 작업 파라미터를 설정해야 합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용를 참조하세요.

Python

이 예에서는 write_data_frame.from_catalog 메서드를 사용합니다.

```
# Example: Insert into a Delta Lake table in S3 using Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

이 예에서는 getCatalogSink 메서드를 사용합니다.

```
// Example: Insert into a Delta Lake table in S3 using Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = additionalOptions)
      .writeDataFrame(dataFrame, glueContext)
```

```
  }
}
```

예: Spark API를 사용하여 Amazon S3에서 Delta Lake 테이블 읽기

이 예에서는 Spark API를 사용하여 Amazon S3에서 Delta Lake 테이블을 읽습니다.

Python

```
# Example: Read a Delta Lake table from S3 using a Spark DataFrame

dataFrame = spark.read.format("delta").load("s3://<s3path/>")
```

Scala

```
// Example: Read a Delta Lake table from S3 using a Spark DataFrame

val dataFrame = spark.read.format("delta").load("s3://<s3path/>")
```

예: Spark를 사용하여 Amazon S3에 Delta Lake 테이블 쓰기

이 예에서는 Spark를 사용하여 Amazon S3에 Delta Lake 테이블을 씁니다.

Python

```
# Example: Write a Delta Lake table to S3 using a Spark DataFrame

dataFrame.write.format("delta") \
    .options(**additional_options) \
    .mode("overwrite") \
    .partitionBy("<your_partitionkey_field>") \
    .save("s3://<s3Path>")
```

Scala

```
// Example: Write a Delta Lake table to S3 using a Spark DataFrame

dataFrame.write.format("delta")
  .options(additionalOptions)
  .mode("overwrite")
  .partitionBy("<your_partitionkey_field>")
```

```
  .save("s3://<s3path/>")
```

예: Lake Formation 권한 제어가 포함된 Delta Lake 테이블 읽기 및 쓰기

이 예제에서는 Lake Formation 권한 제어를 사용하여 Delta Lake 테이블을 읽고 씁니다.

1. Delta 테이블을 생성하여 Lake Formation에 등록하기

   a. Lake Formation 권한 제어를 활성화하려면 먼저 Lake Formation에 Amazon S3 경로를 등록해야 합니다. 자세한 내용을 알아보려면 Registering an Amazon S3 location(Amazon S3 위치 등록)을 참조하세요. Lake Formation 콘솔에서 등록하거나 AWS CLI를 사용하여 등록할 수 있습니다.

   ```
   aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-
   folder> --use-service-linked-role --region <REGION>
   ```

   Amazon S3 위치를 등록하면 해당 위치(또는 하위 위치)를 가리키는 AWS Glue 테이블이 `GetTable` 호출 시 `IsRegisteredWithLakeFormation` 파라미터 값을 true로 반환합니다.

   b. Spark를 통해 등록된 Amazon S3 경로를 가리키는 Delta 테이블을 생성합니다.

   > **Note**
   >
   > Python의 예를 들면 다음과 같습니다.

   ```
   dataFrame.write \
    .format("delta") \
    .mode("overwrite") \
    .partitionBy("<your_partitionkey_field>") \
    .save("s3://<the_s3_path>")
   ```

   Amazon S3에 데이터를 기록한 후 AWS Glue 크롤러를 사용하여 새 Delta 카탈로그 테이블을 생성합니다. 자세한 내용은 AWSGlue 크롤러를 통한 네이티브 Delta Lake 테이블 지원 소개를 참조하십시오.

   AWSGlue `CreateTable` API를 통해 테이블을 수동으로 생성할 수도 있습니다.

2. AWSGlue 작업 IAM 역할에 Lake Formation 권한을 부여하십시오. Lake Formation 콘솔에서 권한을 부여하거나 AWS CLI를 사용하여 권한을 부여할 수 있습니다. 자세한 내용을 알아보려면 Lake Formation 콘솔 및 명명된 리소스 방법을 사용하여 데이터베이스 권한 부여를 참조하십시오

3. Lake Formation에 등록된 Delta 테이블을 읽어보십시오. 코드는 등록되지 않은 Delta 테이블을 읽는 것과 같습니다. 읽기에 성공하려면 AWS Glue 작업 IAM 역할에 SELECT 권한이 있어야 한다는 점에 유의하십시오.

```
# Example: Read a Delta Lake table from Glue Data Catalog

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

4. Lake Formation에 등록된 Delta 테이블에 글을 씁니다. 코드는 등록되지 않은 Delta 테이블에 쓰는 것과 같습니다. 참고로 AWS Glue 작업 IAM 역할에 SUPER 권한이 있어야 쓰기가 성공합니다.

   기본 설정상 AWS Glue는 Append를 saveMode로 사용합니다. additional_options에서 saveMode 옵션을 설정하여 변경할 수 있습니다. Delta 테이블의 saveMode 지원에 대한 자세한 내용은 테이블에 쓰기를 참조하십시오.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

AWS Glue에서 Iceberg 프레임워크 사용

AWS Glue 3.0 이상에서는 데이터 레이크에 대한 Apache Iceberg 프레임워크를 지원합니다. Iceberg는 SQL 테이블처럼 작동하는 고성능 테이블 형식을 제공합니다. 이 항목에서는 Iceberg 테이블에서 데이터를 전송하거나 저장할 때 AWS Glue에서 데이터를 사용하는 데 사용할 수 있는 기능에 대해 설명합니다. Iceberg에 대한 자세한 내용은 공식 Apache Iceberg 설명서를 참조하세요.

AWS Glue를 사용하여 Amazon S3의 Iceberg 테이블에서 읽기 및 쓰기 작업을 수행하거나 AWS Glue 데이터 카탈로그를 사용하여 Iceberg 테이블로 작업할 수 있습니다. 삽입 및 모든 Spark 쿼리 Spark 쓰기를 포함한 추가 작업도 지원됩니다. Iceberg 테이블에 대한 업데이트는 지원되지 않습니다.

> **Note**
>
> `ALTER TABLE … RENAME TO`는 Apache Iceberg 0.13.1 for AWS Glue 3.0에서 사용할 수 없습니다.

다음 표에는 각 AWS Glue 버전에 포함된 Iceberg의 버전이 나와 있습니다.

AWS Glue 버전	지원되는 Iceberg 버전
5.0	1.7.1
4.0	1.0.0
3.0	0.13.1

AWS Glue가 지원하는 데이터 레이크 프레임워크에 대한 자세한 내용은 AWS Glue ETL 작업에서 데이터 레이크 프레임워크 사용 섹션을 참조하세요.

Iceberg 프레임워크 활성화

Iceberg for AWS Glue를 활성화하려면 다음 작업을 완료합니다.

- `iceberg`를 `--datalake-formats` 작업 파라미터의 값으로 지정합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용 섹션을 참조하세요.
- AWS Glue 작업에 대한 `--conf` 키를 생성하고 다음 값으로 설정합니다. 또는 스크립트에서 `SparkConf`를 사용하여 다음 구성을 설정할 수 있습니다. 이러한 설정은 Apache Spark에서 Iceberg 테이블을 올바르게 처리하는 데 도움이 됩니다.

```
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
--conf spark.sql.catalog.glue_catalog=org.apache.iceberg.spark.SparkCatalog
--conf spark.sql.catalog.glue_catalog.warehouse=s3://<your-warehouse-dir>/
--conf spark.sql.catalog.glue_catalog.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog
--conf spark.sql.catalog.glue_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO
```

  Lake Formation에 등록된 Iceberg 테이블을 읽거나 쓰는 경우 AWS Glue 5.0 이상 the section called "FGAC를 위한 Lake Formation"에서의 지침을 따르세요. AWS Glue 4.0에서 다음 구성을 추가하여 Lake Formation 지원을 활성화합니다.

```
--conf spark.sql.catalog.glue_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.glue_catalog.glue.id=<table-catalog-id>
```

AWS Glue 3.0을 Iceberg 0.13.1과 함께 사용하는 경우 Amazon DynamoDB 잠금 관리자를 사용하여 원자성 트랜잭션을 보장하려면 다음과 같은 추가 구성을 설정해야 합니다. AWS Glue 4.0 이상은 기본적으로 낙관적 잠금을 사용합니다. 자세한 내용은 공식 Apache Iceberg 설명서의 Iceberg AWS 통합을 참조하세요.

```
--conf spark.sql.catalog.glue_catalog.lock-
impl=org.apache.iceberg.aws.glue.DynamoLockManager
--conf spark.sql.catalog.glue_catalog.lock.table=<your-dynamodb-table-name>
```

다른 Iceberg 버전 사용

AWS Glue에서 지원하지 않는 Iceberg 버전을 사용하려면 `--extra-jars` 작업 파라미터를 사용하여 고유한 Iceberg JAR 파일을 지정하세요. `--datalake-formats` 파라미터의 값으로 `iceberg`를 포함하지 마세요. AWS Glue 5.0을 사용하는 경우, `--user-jars-first true` 작업 파라미터를 설정해야 합니다.

Iceberg 테이블에 대한 암호화 활성화

> **Note**
>
> Iceberg 테이블에는 서버 측 암호화를 활성화하는 자체 메커니즘이 있습니다. AWS Glue의 보안 구성 외에도 이 구성을 활성화해야 합니다.

Iceberg 테이블에서 서버 측 암호화를 활성화하려면 Iceberg 설명서의 지침을 검토하세요.

Iceberg 교차 리전에 대한 Spark 구성 추가

AWS Glue Data Catalog 및 AWS Lake Formation을 사용하여 Iceberg 교차 리전 테이블 액세스에 대한 스파크 구성을 추가하려면 아래 단계를 따르세요.

1. 다중 리전 액세스 포인트를 생성합니다.
2. 다음 Spark 속성을 설정합니다.

```
-----
```

```
    --conf spark.sql.catalog.my_catalog.s3.use-arn-region-enabled=true \
    --conf spark.sql.catalog.{CATALOG}.s3.access-points.bucket1",
 "arn:aws:s3::<account-id>:accesspoint/<mrap-id>.mrap \
    --conf spark.sql.catalog.{CATALOG}.s3.access-points.bucket2",
 "arn:aws:s3::<account-id>:accesspoint/<mrap-id>.mrap
-----
```

예: Amazon S3에 Iceberg 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록

이 예제 스크립트는 Amazon S3에 Iceberg 테이블을 쓰는 방법을 보여줍니다. 이 예제에서는 Iceberg AWS 통합을 사용하여 테이블을 AWS Glue 데이터 카탈로그에 등록합니다.

Python

```
# Example: Create an Iceberg table from a DataFrame
# and register the table to Glue Data Catalog

dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

query = f"""
CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
USING iceberg
TBLPROPERTIES ("format-version"="2")
AS SELECT * FROM tmp_<your_table_name>
"""
spark.sql(query)
```

Scala

```
// Example: Example: Create an Iceberg table from a DataFrame
// and register the table to Glue Data Catalog

dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

val query = """CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
USING iceberg
TBLPROPERTIES ("format-version"="2")
AS SELECT * FROM tmp_<your_table_name>
"""
spark.sql(query)
```

대신에 Spark 메서드를 사용하여 Amazon S3 및 데이터 카탈로그에 Iceberg 테이블을 작성할 수 있습니다.

사전 조건: Iceberg 라이브러리에서 사용할 카탈로그를 프로비저닝해야 합니다. AWS Glue 데이터 카탈로그를 사용할 때 AWS Glue를 사용하면 이 작업을 간편하게 수행할 수 있습니다. AWS Glue 데이터 카탈로그는 Spark 라이브러리에서 `glue_catalog`로 사용하도록 미리 구성되어 있습니다. 데이터 카탈로그 테이블은 *`databaseName`* 및 *`tableName`*으로 식별됩니다. AWS Glue 데이터 카탈로그에 대한 자세한 내용은 데이터 검색 및 카탈로그 작성 섹션을 참조하세요.

AWS Glue 데이터 카탈로그를 사용하지 않는 경우 Spark API를 통해 카탈로그를 프로비저닝해야 합니다. 자세한 내용은 Iceberg 설명서의 Spark Configuration을 참조하세요.

이 예시에서는 Spark를 사용하여 데이터 카탈로그에서 Amazon S3의 Iceberg 테이블을 작성합니다.

Python

```
# Example: Write an Iceberg table to S3 on the Glue Data Catalog

# Create (equivalent to CREATE TABLE AS SELECT)
dataFrame.writeTo("glue_catalog.databaseName.tableName") \
    .tableProperty("format-version", "2") \
    .create()

# Append (equivalent to INSERT INTO)
dataFrame.writeTo("glue_catalog.databaseName.tableName") \
    .tableProperty("format-version", "2") \
    .append()
```

Scala

```
// Example: Write an Iceberg table to S3 on the Glue Data Catalog

// Create (equivalent to CREATE TABLE AS SELECT)
dataFrame.writeTo("glue_catalog.databaseName.tableName")
    .tableProperty("format-version", "2")
    .create()

// Append (equivalent to INSERT INTO)
dataFrame.writeTo("glue_catalog.databaseName.tableName")
    .tableProperty("format-version", "2")
    .append()
```

예: AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에서 Iceberg 테이블 읽기

이 예제에서는 예: Amazon S3에 Iceberg 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록에서 생성한 Iceberg 테이블을 읽습니다.

Python

이 예에서는 `GlueContext.create_data_frame.from_catalog()` 메서드를 사용합니다.

```
# Example: Read an Iceberg table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

이 예에서는 getCatalogSource 메서드를 사용합니다.

```
// Example: Read an Iceberg table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val df = glueContext.getCatalogSource("<your_database_name>",
 "<your_table_name>",
      additionalOptions = additionalOptions)
      .getDataFrame()
  }
}
```

예: AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에서 Iceberg 테이블에 **DataFrame** 삽입

이 예에서는 예: Amazon S3에 Iceberg 테이블을 작성하고 AWS Glue 데이터 카탈로그에 등록에서 생성한 Iceberg 테이블에 데이터를 삽입합니다.

> **Note**
>
> 이 예에서는 AWS Glue 데이터 카탈로그를 Apache Spark Hive 메타스토어로 사용하기 위해 `--enable-glue-datacatalog` 작업 파라미터를 설정해야 합니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용를 참조하세요.

Python

이 예에서는 `GlueContext.write_data_frame.from_catalog()` 메서드를 사용합니다.

```
# Example: Insert into an Iceberg table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

이 예에서는 getCatalogSink 메서드를 사용합니다.

```
// Example: Insert into an Iceberg table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
```

```
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = additionalOptions)
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

예: Spark를 사용하여 Amazon S3에서 Iceberg 테이블 읽기

사전 조건: Iceberg 라이브러리에서 사용할 카탈로그를 프로비저닝해야 합니다. AWS Glue 데이터 카탈로그를 사용할 때 AWS Glue를 사용하면 이 작업을 간편하게 수행할 수 있습니다. AWS Glue 데이터 카탈로그는 Spark 라이브러리에서 `glue_catalog`로 사용하도록 미리 구성되어 있습니다. 데이터 카탈로그 테이블은 *`databaseName`* 및 *`tableName`*으로 식별됩니다. AWS Glue 데이터 카탈로그에 대한 자세한 내용은 데이터 검색 및 카탈로그 작성 섹션을 참조하세요.

AWS Glue 데이터 카탈로그를 사용하지 않는 경우 Spark API를 통해 카탈로그를 프로비저닝해야 합니다. 자세한 내용은 Iceberg 설명서의 Spark Configuration을 참조하세요.

이 예제에서는 Spark를 사용하여 데이터 카탈로그에서 Amazon S3의 Iceberg 테이블을 읽습니다.

Python

```
# Example: Read an Iceberg table on S3 as a DataFrame from the Glue Data Catalog

dataFrame = spark.read.format("iceberg").load("glue_catalog.databaseName.tableName")
```

Scala

```
// Example: Read an Iceberg table on S3 as a DataFrame from the Glue Data Catalog

val dataFrame =
 spark.read.format("iceberg").load("glue_catalog.databaseName.tableName")
```

예: Lake Formation 권한 제어가 포함된 Iceberg 테이블 읽기 및 쓰기

이 예제에서는 Lake Formation 권한 제어를 사용하여 Iceberg 테이블을 읽고 씁니다.

> **Note**
>
> 이 예제는 AWS Glue 4.0에서만 작동합니다. AWS Glue 5.0 이상에서는 the section called "FGAC를 위한 Lake Formation"의 지침을 따릅니다.

1. Iceberg 테이블을 만들고 Lake Formation에 등록하십시오.
   a. Lake Formation 권한 제어를 활성화하려면 먼저 Lake Formation에 Amazon S3 경로를 등록해야 합니다. 자세한 내용을 알아보려면 Registering an Amazon S3 location(Amazon S3 위치 등록)을 참조하세요. Lake Formation 콘솔에서 등록하거나 AWS CLI를 사용하여 등록할 수 있습니다.

```
aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-
folder> --use-service-linked-role --region <REGION>
```

   Amazon S3 위치를 등록하면 해당 위치(또는 하위 위치)를 가리키는 AWS Glue 테이블이 `GetTable` 호출 시 `IsRegisteredWithLakeFormation` 파라미터 값을 true로 반환합니다.

   b. Spark SQL을 통해 등록된 경로를 가리키는 Iceberg 테이블을 생성하십시오.

> **Note**
>
> Python의 예를 들면 다음과 같습니다.

```
dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

query = f"""
CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
USING iceberg
AS SELECT * FROM tmp_<your_table_name>
"""
spark.sql(query)
```

   AWS Glue `CreateTable` API를 통해 테이블을 수동으로 생성할 수도 있습니다. 자세한 내용은 Apache Iceberg 테이블 생성을 참조하십시오.

> **Note**
>
> 현재 `UpdateTable` API는 작업에 대한 입력으로 Iceberg 테이블 형식을 지원하지 않습니다.

2. 작업 IAM 역할에 Lake Formation 권한을 부여하십시오. Lake Formation 콘솔에서 권한을 부여하거나 AWS CLI를 사용하여 권한을 부여할 수 있습니다. 자세한 내용은 https://docs.aws.amazon.com/lake-formation/latest/dg/granting-table-permissions.html 단원을 참조하십시오.
3. Lake Formation에 등록된 Iceberg 테이블을 참조하십시오. 코드는 등록되지 않은 Iceberg 테이블을 읽는 것과 같습니다. 읽기에 성공하려면 AWS Glue Job IAM 역할에 SELECT 권한이 있어야 한다는 점에 유의하십시오.

```
# Example: Read an Iceberg table from the AWS Glue Data Catalog
from awsglue.context import GlueContextfrom pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

4. Lake Formation에 등록된 Iceberg 테이블에 씁니다. 코드는 등록되지 않은 Iceberg 테이블에 쓰는 것과 같습니다. 글쓰기가 성공하려면 AWS Glue 작업 IAM 역할에 SUPER 권한이 있어야 한다는 점을 참고하십시오.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

공유 구성 참조

모든 형식 유형의 다음과 같은 `format_options` 값을 사용할 수 있습니다.

- `attachFilename` - 열 이름으로 사용할 적절한 형식의 문자열입니다. 이 옵션을 제공하면 레코드의 소스 파일 이름이 레코드에 추가됩니다. 파라미터 값이 열 이름으로 사용됩니다.
- `attachTimestamp` - 열 이름으로 사용할 적절한 형식의 문자열입니다. 이 옵션을 제공하면 레코드의 소스 파일 수정 시간이 레코드에 추가됩니다. 파라미터 값이 열 이름으로 사용됩니다.

Spark SQL 작업에 대해 AWS Glue Data Catalog 지원

AWS Glue Data Catalog는 Apache Hive 메타스토어와 호환되는 카탈로그입니다. Data Catalog를 외부 Apache Hive 메타스토어로 사용하도록 AWS Glue 작업과 개발 엔드포인트를 구성할 수 있습니다. 그런 다음, Data Catalog에 저장된 테이블에 대해 Apache Spark SQL 쿼리를 직접 실행할 수 있습니다. AWS Glue 동적 프레임은 기본값으로 Data Catalog와 통합됩니다. 한편, Spark SQL 작업은 이 기능을 통해 외부 Hive 메타스토어로 Data Catalog를 사용하기 시작할 수 있습니다.

이 기능을 사용하려면 AWS Glue API 엔드포인트에 대한 네트워크 액세스가 필요합니다. 프라이빗 서브넷에 있는 연결을 사용하는 AWS Glue 작업의 경우 네트워크 액세스를 제공하도록 VPC 엔드포인트 또는 NAT 게이트웨이를 구성해야 합니다. VPC 엔드포인트 구성 방법에 대한 자세한 내용은 데이터 스토어에 대한 네트워크 액세스 설정 섹션을 참조하세요. NAT 게이트웨이를 생성하려면 Amazon VPC 사용 설명서의 NAT 게이트웨이를 참조하세요.

작업 인수 및 개발 엔드포인트 인수에 각각 `"--enable-glue-datacatalog": ""`를 추가하여 AWS Glue 작업 및 개발 엔드포인트를 구성할 수 있습니다. 이 인수를 전달하면 외부 Hive 메타스토어로 Data Catalog에 액세스 할 수 있도록 Spark에서 특정 구성이 설정됩니다. 또한 AWS Glue 작업 또는 개발 엔드포인트에서 생성된 `SparkSession` 객체에서 Hive 지원을 사용합니다.

Data Catalog 액세스를 사용하려면 콘솔의 작업 추가(Add job) 또는 엔드포인트 추가(Add endpoint) 페이지에 있는 카탈로그 옵션(Catalog options) 그룹에서 AWS Glue Data Catalog를 Hive 메타스토어로 사용(Use Glue Data Catalog as the Hive metastore) 확인란을 선택합니다. 작업 또는 개발 엔드포인트에 사용되는 IAM 역할은 `glue:CreateDatabase` 권한을 가져야 합니다. "`default`"라는 데이터베이스가 없는 경우에 Data Catalog에서 생성됩니다.

Spark SQL 작업에서 이 기능을 사용할 수 있는 방법의 예제를 살펴보십시오. 다음 예에서는 `s3://awsglue-datasets/examples/us-legislators`에서 사용 가능한 미국 법률 기관의 데이터 세트를 크롤링했다고 가정합니다.

AWS Glue Data Catalog에 정의된 테이블에서 데이터를 직렬화/역직렬화하려면 Spark 작업의 클래스 경로의 AWS Glue Data Catalog에 정의된 포맷을 위한 Hive SerDe 클래스가 Spark SQL에 있어야 합니다.

특정 공통 형식을 위한 SerDe는 AWS Glue에 의해 배포됩니다. 다음은 이들에 대한 Amazon S3 링크입니다.

- JSON
- XML
- Grok

개발 엔드포인트에 대한 외부 JAR로서 JSON SerDe를 추가합니다. 작업 시, 인수 필드의 `--extra-jars` 인수를 사용하여 SerDe를 추가할 수 있습니다. 자세한 내용은 AWS Glue 작업에서 작업 파라미터 사용 단원을 참조하십시오.

다음은 Spark SQL에서 Data Catalog를 사용할 수 있도록 설정된 개발 엔드포인트를 생성하기 위한 입력 JSON의 예입니다.

```
{
    "EndpointName": "Name",
    "RoleArn": "role_ARN",
    "PublicKey": "public_key_contents",
    "NumberOfNodes": 2,
    "Arguments": {
      "--enable-glue-datacatalog": ""
    },
    "ExtraJarsS3Path": "s3://crawler-public/json/serde/json-serde.jar"
}
```

이제 Spark SQL을 사용하여 미국 법률 기관 데이터 세트에서 생성된 테이블을 쿼리합니다.

```
>>> spark.sql("use legislators")
DataFrame[]
>>> spark.sql("show tables").show()
+-----------+-----------------+-----------+
|   database|        tableName|isTemporary|
+-----------+-----------------+-----------+
|legislators|       areas_json|      false|
|legislators|   countries_json|      false|
|legislators|      events_json|      false|
|legislators| memberships_json|      false|
|legislators|organizations_json|      false|
|legislators|     persons_json|      false|
```

```
+-----------+-----------------+-----------+
>>> spark.sql("describe memberships_json").show()
+--------------------+---------+-----------------+
|            col_name|data_type|          comment|
+--------------------+---------+-----------------+
|             area_id|   string|from deserializer|
|     on_behalf_of_id|   string|from deserializer|
|     organization_id|   string|from deserializer|
|                role|   string|from deserializer|
|           person_id|   string|from deserializer|
|legislative_perio...|   string|from deserializer|
|          start_date|   string|from deserializer|
|            end_date|   string|from deserializer|
+--------------------+---------+-----------------+
```

이 형식에 대한 SerDe 클래스를 작업의 클래스 경로에서 사용할 수 없는 경우에는 아래와 비슷한 오류가 나타납니다.

```
>>> spark.sql("describe memberships_json").show()

Caused by: MetaException(message:java.lang.ClassNotFoundException Class
 org.openx.data.jsonserde.JsonSerDe not found)
    at
 org.apache.hadoop.hive.metastore.MetaStoreUtils.getDeserializer(MetaStoreUtils.java:399)
    at
 org.apache.hadoop.hive.ql.metadata.Table.getDeserializerFromMetaStore(Table.java:276)
    ... 64 more
```

`memberships` 테이블에서 별도의 `organization_id`만 보려면 다음 SQL 쿼리를 실행합니다.

```
>>> spark.sql("select distinct organization_id from memberships_json").show()
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

동적 프레임에서 똑같이 하려면 다음을 실행합니다.

```
>>> memberships = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="memberships_json")
```

```
>>> memberships.toDF().createOrReplaceTempView("memberships")
>>> spark.sql("select distinct organization_id from memberships").show()
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

DynamicFrame이 ETL 작업에 최적화된 상태에서 Spark SQL이 Data Catalog에 액세스할 수 있도록 하면 손쉽게 복잡한 SQL 문을 실행하거나 기존 애플리케이션을 포팅할 수 있습니다.

작업 북마크 사용

AWS Glue for Spark는 작업 북마크를 사용하여 이미 진행된 데이터를 추적합니다. 작업 북마크 기능에 대한 요약 및 지원 내용은 the section called “처리된 데이터를 작업 북마크로 추적” 섹션을 참조하세요. 북마크를 사용해 AWS Glue 작업을 프로그래밍하면 시각적 작업에서 얻을 수 없었던 유연성을 활용할 수 있습니다.

- JDBC에서 읽을 때 AWS Glue 스크립트에서 북마크 키로 사용할 열을 지정할 수 있습니다.
- 각 메서드의 직접 호출에 적용할 `transformation_ctx`를 선택할 수 있습니다.

스크립트 시작 부분에서는 항상 *`job.init`*를 호출하고 스크립트 끝 부분에서는 적절하게 구성된 파라미터를 사용하여 *`job.commit`*을 호출하세요. 이 두 함수는 북마크 서비스를 초기화하고 서비스에 대한 상태 변경을 업데이트합니다. 북마크는 호출하지 않으면 작동하지 않습니다.

북마크 키 지정

JDBC 워크플로의 경우 북마크는 키 필드의 값을 북마크된 값과 비교하여 작업에서 읽은 행을 추적합니다. 이는 Amazon S3 워크플로에 필요하지 않거나 적용될 수 없습니다. 시각적 편집기 없이 AWS Glue 스크립트를 작성할 때 북마크로 추적할 열을 지정할 수 있습니다. 열을 여러 개 지정할 수도 있습니다. 사용자 정의 북마크 키를 지정할 때는 값 순서에 간격을 둘 수 있습니다.

> **Warning**
>
> 사용자 정의 북마크 키를 사용하는 경우 각각 엄격하게 단순 증가 또는 감소해야 합니다. 복합 키에 대해 추가 필드를 선택할 때 '마이너 버전' 또는 '개정 번호'와 같은 개념에 대한 필드는 해당 값이 데이터 세트 전체에서 재사용되므로 이 기준을 충족하지 않습니다.

다음과 같은 방법으로 jobBookmarkKeys 및 jobBookmarkKeysSortOrder를 지정할 수 있습니다.

- create_dynamic_frame.from_catalog - additional_options를 사용합니다.
- create_dynamic_frame.from_options - connection_options를 사용합니다.

변환 컨텍스트

대부분의 AWS Glue PySpark 동적 프레임 메서드에는 ETL 연산자 인스턴스의 고유한 식별자인 조건부 파라미터 transformation_ctx가 들어 있습니다. transformation_ctx 파라미터는 작업 북마크 내에서 지정된 연산자의 상태 정보를 식별하는 데 사용됩니다. 특히 AWS Glue는 transformation_ctx를 사용해 북마크 상태에 키를 인덱싱합니다.

> **Warning**
>
> 이 transformation_ctx는 스크립트에서 특정 소스에 대한 책갈피 상태를 검색하는 키 역할을 합니다. 북마크가 제대로 작동하려면 항상 소스와 관련 transformation_ctx를 일치시켜야 합니다. 소스 속성이나 transformation_ctx의 이름을 변경하면 이전 북마크를 유효하지 않게 만들 수 있으며 타임스탬프 기반 필터링으로 인해 올바른 결과가 나오지 않을 수 있습니다.

작업 북마크가 제대로 작동하려면 작업 북마크 파라미터를 활성화하고 transformation_ctx 파라미터를 설정해야 합니다. transformation_ctx 파라미터를 전달하지 않으면 메서드에서 사용하는 테이블이나 동적 프레임에 대해 작업 북마크가 활성화되지 않습니다. 예를 들어, ETL 작업이 2개의 Amazon S3 소스를 읽고 조인하면 transformation_ctx 파라미터를 사용하고자 하는 북마크의 방법에만 전달할 수도 있습니다. 작업의 작업 북마크를 재설정하면 어떤 transformation_ctx를 사용하든 간에 해당 작업과 연관된 모든 변환이 재설정됩니다.

DynamicFrameReader 클래스에 대한 자세한 내용은 DynamicFrameReader 클래스를 참조하십시오. PySpark 확장에 대한 자세한 내용은 AWS Glue PySpark 확장 참조 단원을 참조하십시오.

예시

Example

다음은 Amazon S3 데이터 원본에 대해 생성된 스크립트의 예입니다. 작업 북마크를 사용하는 데 필요한 스크립트 부분은 기울임꼴로 표시됩니다. 이러한 요소에 대한 자세한 내용은 GlueContext 클래스 API 및 DynamicFrameWriter 클래스 API를 참조하세요.

```
# Sample Script
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)

datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = "database",
    table_name = "relatedqueries_csv",
    transformation_ctx = "datasource0"
)

applymapping1 = ApplyMapping.apply(
    frame = datasource0,
    mappings = [("col0", "string", "name", "string"), ("col1", "string", "number",
 "string")],
    transformation_ctx = "applymapping1"
)

datasink2 = glueContext.write_dynamic_frame.from_options(
    frame = applymapping1,
    connection_type = "s3",
    connection_options = {"path": "s3://input_path"},
    format = "json",
    transformation_ctx = "datasink2"
)


job.commit()
```

Example

다음은 JDBC 소스에 대해 생성된 스크립트의 예입니다. 소스 테이블은 기본 키로 empno 열이 있는 직원 테이블입니다. 북마크 키가 지정되지 않은 경우 작업은 기본적으로 순차 기본 키를 북마크 키로 사

용하지만 empno는 반드시 순차적이지 않기 때문에(값에 공백이 있을 수 있음) 기본 북마크 키로 적합하지 않습니다. 따라서 스크립트는 empno를 북마크 키로 명시적으로 지정합니다. 코드의 해당 부분은 기울임꼴로 표시됩니다.

```
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)

datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = "hr",
    table_name = "emp",
    transformation_ctx = "datasource0",
    additional_options = {"jobBookmarkKeys":["empno"],"jobBookmarkKeysSortOrder":"asc"}
)

applymapping1 = ApplyMapping.apply(
    frame = datasource0,
    mappings = [("ename", "string", "ename", "string"), ("hrly_rate", "decimal(38,0)",
 "hrly_rate", "decimal(38,0)"), ("comm", "decimal(7,2)", "comm", "decimal(7,2)"),
 ("hiredate", "timestamp", "hiredate", "timestamp"), ("empno", "decimal(5,0)", "empno",
 "decimal(5,0)"), ("mgr", "decimal(5,0)", "mgr", "decimal(5,0)"), ("photo", "string",
 "photo", "string"), ("job", "string", "job", "string"), ("deptno", "decimal(3,0)",
 "deptno", "decimal(3,0)"), ("ssn", "decimal(9,0)", "ssn", "decimal(9,0)"), ("sal",
 "decimal(7,2)", "sal", "decimal(7,2)")],
    transformation_ctx = "applymapping1"
)

datasink2 = glueContext.write_dynamic_frame.from_options(
    frame = applymapping1,
    connection_type = "s3",
    connection_options = {"path": "s3://hr/employees"},
    format = "csv",
```

```
    transformation_ctx = "datasink2"
)

job.commit()
```

AWS Glue Studio 외부에서 민감한 데이터 감지 사용

AWS Glue Studio는 민감한 데이터 감지를 허용하지만 AWS Glue Studio 외부에서 민감한 데이터 감지 기능을 사용할 수도 있습니다.

민감한 관리형 데이터 형식의 목록은 관리형 데이터 형식을 참조하세요.

AWS 관리형 PII 유형을 사용한 민감한 데이터 감지 탐지

AWS Glue에서는 AWS Glue ETL 작업에서 2개의 API를 제공합니다. 이러한 항목은 `detect()` 및 `classifyColumns()`입니다.

```
 detect(frame: DynamicFrame,
     entityTypesToDetect: Seq[String],
     outputColumnName: String = "DetectedEntities",
     detectionSensitivity: String = "LOW"): DynamicFrame

detect(frame: DynamicFrame,
     detectionParameters: JsonOptions,
     outputColumnName: String = "DetectedEntities",
     detectionSensitivity: String = "LOW"): DynamicFrame

 classifyColumns(frame: DynamicFrame,
     entityTypesToDetect: Seq[String],
     sampleFraction: Double = 0.1,
     thresholdFraction: Double = 0.1,
     detectionSensitivity: String = "LOW")
```

`detect()` API를 사용하여 AWS 관리형 PII 유형 및 사용자 지정 엔터티 유형을 식별할 수 있습니다. 새로운 열이 감지 결과와 함께 자동 생성됩니다. `classifyColumns()` API는 키가 열 이름이고 값이 감지된 엔터티 유형 목록인 맵을 반환합니다. `SampleFraction`은 PII 엔터티를 검색할 때 샘플링할 데이터의 비율을 나타내는 반면, `ThresholdFraction`은 열이 PII 데이터로 식별되기 위해 충족되어야 하는 데이터의 비율을 나타냅니다.

행 수준 탐지

이 예제에서는 작업이 `detect()` 및`classifyColumns()` API를 사용하여 다음 작업을 수행합니다.

- Amazon S3 버킷에서 데이터를 읽고 이를 DynamicFrame으로 변환
- DynamicFrame에서 '이메일' 및 '신용 카드' 인스턴스 검색
- 원래 값과 각 행에 대한 감지 결과를 포함하는 하나의 열이 포함된 DynamicFrame 반환
- 반환된 DynamicFrame을 다른 Amazon S3 경로에 쓰기

```
 import com.amazonaws.services.glue.GlueContext
 import com.amazonaws.services.glue.MappingSpec
 import com.amazonaws.services.glue.errors.CallSite
 import com.amazonaws.services.glue.util.GlueArgParser
 import com.amazonaws.services.glue.util.Job
 import com.amazonaws.services.glue.util.JsonOptions
 import org.apache.spark.SparkContext
 import scala.collection.JavaConverters._
 import com.amazonaws.services.glue.ml.EntityDetector

 object GlueApp {
   def main(sysArgs: Array[String]) {
     val spark: SparkContext = new SparkContext()
     val glueContext: GlueContext = new GlueContext(spark)
     val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
     Job.init(args("JOB_NAME"), glueContext, args.asJava)
     val frame=
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
"withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
transformationContext="AmazonS3_node1650160158526").getDynamicFrame()

     val frameWithDetectedPII = EntityDetector.detect(frame, Seq("EMAIL",
"CREDIT_CARD"))

     glueContext.getSinkWithFormat(connectionType="s3",
options=JsonOptions("""{"path": "s3://pathToOutput/", "partitionKeys": []}"""),
transformationContext="someCtx",
format="json").writeDynamicFrame(frameWithDetectedPII)

     Job.commit()
```

```
    }
  }
```

세분화된 동작을 통한 행 수준 탐지

이 예제에서는 작업이 `detect()` API를 사용하여 다음 작업을 수행합니다.

- Amazon S3 버킷에서 데이터를 읽고 이를 dynamicFrame으로 변환
- dynamicFrame에서 “USA_PTIN”, “ BANK_ACCOUNT”, “USA_SSN”, “USA_PASSPORT_NUMBER”, “PHONE_NUMBER”에 대한 민감한 데이터 유형 탐지
- 수정된 마스크 값과 각 행에 대한 감지 결과를 포함하는 하나의 열이 포함된 dynamicFrame 반환
- 반환된 dynamicFrame을 다른 Amazon S3 경로에 쓰기

위의 `detect()` API와 달리 이 API에서는 개체 유형이 탐지할 수 있도록 세분화된 동작을 사용합니다. 자세한 내용은 [detect() 사용을 위한 탐지 파라미터] 단원을 참조하십시오.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.ml.EntityDetector

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    val frame =
 glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
 "withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
 options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
 transformationContext="AmazonS3_node_source").getDynamicFrame()

    val detectionParameters = JsonOptions(
      """
```

```
        {
          "USA_DRIVING_LICENSE": [{
            "action": "PARTIAL_REDACT",
            "sourceColumns": ["Driving License"],
            "actionOptions": {
              "matchPattern": "[0-9]",
              "redactChar": "*"
            }
          }],
          "BANK_ACCOUNT": [{
            "action": "DETECT",
            "sourceColumns": ["*"]
          }],
          "USA_SSN": [{
            "action": "SHA256_HASH",
            "sourceColumns": ["SSN"]
          }],
          "IP_ADDRESS": [{
            "action": "REDACT",
            "sourceColumns": ["IP Address"],
            "actionOptions": {"redactText": "*****"}
          }],
          "PHONE_NUMBER": [{
            "action": "PARTIAL_REDACT",
            "sourceColumns": ["Phone Number"],
            "actionOptions": {
              "numLeftCharsToExclude": 1,
              "numRightCharsToExclude": 0,
              "redactChar": "*"
            }
          }]
        }
      """
    )

    val frameWithDetectedPII = EntityDetector.detect(frame, detectionParameters,
"DetectedEntities", "HIGH")

    glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path":
"s3://pathToOutput/", "partitionKeys": []}"""),
transformationContext="AmazonS3_node_target",
format="json").writeDynamicFrame(frameWithDetectedPII)

    Job.commit()
```

```
  }
}
```

열 수준 탐지

이 예제에서는 작업이 `classifyColumns()` API를 사용하여 다음 작업을 수행합니다.

- Amazon S3 버킷에서 데이터를 읽고 이를 dynamicFrame으로 변환
- DynamicFrame에서 '이메일' 및 '신용 카드' 인스턴스 검색
- 열을 100% 샘플링하고, 셀의 10% 에 존재하며, 민감도가 “LOW”이면 개체를 탐지된 것으로 표시하도록 파라미터를 설정합니다
- 키가 열 이름이고 값이 탐지된 개체 유형 목록인 맵을 반환합니다
- 반환된 dynamicFrame을 다른 Amazon S3 경로에 쓰기

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.DynamicFrame
import com.amazonaws.services.glue.ml.EntityDetector

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    val frame =
 glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar":
 "\"", "withHeader": true, "separator": ",", "optimizePerformance": false}"""),
 connectionType="s3", format="csv", options=JsonOptions("""{"paths": ["s3://
pathToSource"], "recurse": true}"""), transformationContext="frame").getDynamicFrame()

    import glueContext.sparkSession.implicits._
```

```
    val detectedDataFrame = EntityDetector.classifyColumns(
        frame,
        entityTypesToDetect = Seq("CREDIT_CARD", "PHONE_NUMBER"),
        sampleFraction = 1.0,
        thresholdFraction = 0.1,
        detectionSensitivity = "LOW"
    )
    val detectedDF = (detectedDataFrame).toSeq.toDF("columnName", "entityTypes")
    val DetectSensitiveData_node = DynamicFrame(detectedDF, glueContext)

    glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path":
 "s3://pathToOutput", "partitionKeys": []}"""), transformationContext="someCtx",
 format="json").writeDynamicFrame(DetectSensitiveData_node)

    Job.commit()
  }
}
```

AWS CustomEntityType PII 유형을 사용한 민감한 데이터 감지 탐지

AWS Studio를 통해 사용자 지정 엔터티를 정의할 수 있습니다. 하지만 AWS Studio에서 이 기능을 사용하기 위해서는 먼저 사용자 지정 엔터티 유형을 정의한 다음 정의된 사용자 지정 엔터티 유형을 `entityTypesToDetect` 목록에 추가해야 합니다.

데이터에 특정 민감한 데이터 유형(예: '직원 ID')이 있는 경우에는 `CreateCustomEntityType()` API를 호출하여 사용자 지정 엔터티를 만들 수 있습니다. 다음 예제에서는 요청 파라미터를 사용하여 `CreateCustomEntityType()` API에 사용자 지정 엔터티 유형 'EMPLOYE_ID'를 정의합니다.

```
  {
      "name": "EMPLOYEE_ID",
      "regexString": "\d{4}-\d{3}",
      "contextWords": ["employee"]
  }
```

그리고 나서 `EntityDetector()` API에 사용자 지정 엔터티 유형(EMPLOYEE_ID)을 추가하여 새로운 사용자 지정 민감한 데이터 유형을 사용하도록 작업을 수정합니다.

```
 import com.amazonaws.services.glue.GlueContext
 import com.amazonaws.services.glue.MappingSpec
 import com.amazonaws.services.glue.errors.CallSite
 import com.amazonaws.services.glue.util.GlueArgParser
 import com.amazonaws.services.glue.util.Job
 import com.amazonaws.services.glue.util.JsonOptions
 import org.apache.spark.SparkContext
 import scala.collection.JavaConverters._
 import com.amazonaws.services.glue.ml.EntityDetector

 object GlueApp {
   def main(sysArgs: Array[String]) {
     val spark: SparkContext = new SparkContext()
     val glueContext: GlueContext = new GlueContext(spark)
     val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
     Job.init(args("JOB_NAME"), glueContext, args.asJava)
     val frame=
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
"withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
transformationContext="AmazonS3_node1650160158526").getDynamicFrame()

     val frameWithDetectedPII = EntityDetector.detect(frame, Seq("EMAIL",
"CREDIT_CARD", "EMPLOYEE_ID"))

     glueContext.getSinkWithFormat(connectionType="s3",
options=JsonOptions("""{"path": "s3://pathToOutput/", "partitionKeys": []}"""),
transformationContext="someCtx",
format="json").writeDynamicFrame(frameWithDetectedPII)

     Job.commit()
   }
 }
```

> **Note**
>
> 사용자 지정 민감한 데이터 유형이 기존 관리형 엔터티 유형과 동일한 이름을 사용하여 정의된 경우 사용자 지정 민감한 데이터 유형이 우선 적용되고 관리형 엔터티 유형의 논리를 덮어씁니다.

`detect()` 사용을 위한 탐지 파라미터

이 방법은 DynamicFrame에서 개체를 감지하는 데 사용됩니다. 원래 값과 PII 탐지 메타데이터가 있는 outputColumnName이라는 추가 열이 포함된 새 DataFrame을 반환합니다. 이 DynamicFrame이 AWS Glue 스크립트 내에서 반환된 후에 사용자 지정 마스킹을 수행하거나 세분화된 작업이 포함된 detect() API를 대신 사용할 수 있습니다.

```
detect(frame: DynamicFrame,
       entityTypesToDetect: Seq[String],
       outputColumnName: String = "DetectedEntities",
       detectionSensitivity: String = "LOW"): DynamicFrame
```

파라미터:

- frame - (유형:`DynamicFrame`) 처리할 데이터가 포함된 입력 DynamicFrame입니다.
- entityTypesToDetect - (type:`[Seq[String]`) 탐지할 개체 유형 목록입니다. 관리 개체 유형 또는 사용자 지정 개체 유형일 수 있습니다.
- outputColumnName - (유형: `String`, 기본값: "DetectedEntities") 탐지된 개체를 저장할 열의 이름입니다. 지정하지 않은 경우 기본 열 이름은 "DetectedEntities"입니다.
- detectionSensitivity - (유형: `String`, 옵션: "LOW" or "HIGH", 기본: "LOW") 탐지 프로세스의 민감도를 지정합니다. 유효한 옵션은 "LOW" 또는 "HIGH"입니다. 제공되지 않은 경우 기본 민감도는 "LOW"으로 설정됩니다.

`outputColumnName` 설정:

탐지될 열 이름입니다. 지정하지 않은 경우 기본 열 이름은 "DetectedEntities"입니다. 출력 열의 각 행에 대해 보조 열에는 탐지된 개체 메타데이터에 대한 열 이름 맵이 다음과 같은 키-값 쌍과 함께 포함됩니다.

- entityType - 탐지된 엔티티 유형입니다.
- start - 원본 데이터에서 탐지된 개체의 시작 위치입니다.
- end - 원본 데이터에서 탐지된 개체의 끝 위치입니다.
- actionUsed - 탐지된 개체에 대해 수행된 작업 (예: "DETECT," "REDACT," "PARTIAL_REDACT," "SHA256_HASH").

예제:

```
{
   "DetectedEntities":{
      "SSN Col":[
         {
            "entityType":"USA_SSN",
            "actionUsed":"DETECT",
            "start":4,
            "end":15
         }
      ],
      "Random Data col":[
         {
            "entityType":"BANK_ACCOUNT",
            "actionUsed":"PARTIAL_REDACT",
            "start":4,
            "end":13
         },
         {
            "entityType":"IP_ADDRESS",
            "actionUsed":"REDACT",
            "start":4,
            "end":13
         }
      ]
   }
}
```

세분화된 동작이 포함된 **detect()**의 탐지 파라미터

이 방법은 지정된 매개 변수를 사용하여 DynamicFrame에서 개체를 감지하는 데 사용됩니다. 원래 값이 마스킹된 민감한 데이터로 대체되고 PII 탐지 메타데이터가 있는 추가 `outputColumnName` 열이 포함된 새 DataFrame을 반환합니다.

```
detect(frame: DynamicFrame,
       detectionParameters: JsonOptions,
       outputColumnName: String = "DetectedEntities",
       detectionSensitivity: String = "LOW"): DynamicFrame
```

파라미터:

- frame - (유형: DynamicFrame): 처리할 데이터가 포함된 입력 DynamicFrame입니다.
- detectionParameters - (유형: Json0ptions): 탐지 프로세스의 매개 변수를 지정하는 JSON 옵션입니다.
- outputColumnName - (유형: String, 기본값: "DetectedEntities"): 탐지된 개체를 저장할 열의 이름입니다. 지정하지 않은 경우 기본 열 이름은 "DetectedEntities"입니다.
- detectionSensitivity - (유형: String, 옵션: "LOW" or "HIGH", 기본: "LOW"): 탐지 프로세스의 민감도를 지정합니다. 유효한 옵션은 "LOW" 또는 "HIGH"입니다. 제공되지 않은 경우 기본 민감도는 "LOW"으로 설정됩니다.

detectionParameters 설정

설정이 포함되지 않은 경우 기본값이 사용됩니다.

- action - (유형: String, 옵션: "DETECT", "REDACT", "PARTIAL_REDACT", "SHA256_HASH") 엔티티에 대해 수행할 작업을 지정합니다. 필수 사항입니다. 마스킹을 수행하는 작업 ("DETECT"을 제외한 모든 작업) 은 열당 하나의 작업만 수행할 수 있다는 점에 유의하십시오. 이는 합쳐진 개체를 마스킹하기 위한 예방 조치입니다.
- sourceColumns - (유형: List[String], 기본값: [“*”]) 개체에 대해 탐지를 수행할 소스 열 이름 목록입니다. 없는 경우 기본값은 [“*”]입니다. 잘못된 열 이름을 사용하면 IllegalArgumentException이 발생합니다.
- sourceColumnsToExclude - (유형: List[String]) 엔티티에 대한 탐지를 수행할 소스 열 이름 목록입니다. sourceColumns 또는 sourceColumnsToExclude를 사용합니다. 잘못된 열 이름을 사용하면 IllegalArgumentException이 발생합니다.
- actionOptions - 지정된 작업에 따른 추가 옵션:
  - "DETECT"와 "SHA256_HASH"는 옵션이 불가합니다.
  - "REDACT"의 경우:
    - redactText - (유형: String, 기본값: "*****") 탐지된 개체를 대체할 텍스트입니다.
  - "PARTIAL_REDACT"의 경우:
    - redactChar -(유형: String, 기본값: "*") 개체에서 감지된 각 문자를 대체할 문자입니다.
    - matchPattern - (유형: String) 부분 수정을 위한 정규식 패턴입니다. numLeftCharsToExclude 또는 numRightCharsToExclude과 결합할 수 없습니다.
    - numLeftCharsToExclude - (유형: String, integer) 제외할 왼쪽 문자 수입니다. matchPattern과 함께 사용할 수는 없지만 numRightCharsToExclude와 함께 사용할 수는 있습니다.

- numRightCharsToExclude- (유형: `String, integer`) 제외할 오른쪽 문자 수입니다. matchPattern과 함께 사용할 수는 없지만 `numRightCharsToExclude`와 함께 사용할 수는 있습니다.

`outputColumnName` 설정

outputColumnName 설정을 참조하십시오

`classifyColumns()`의 탐지 파라미터

이 방법은 DynamicFrame에서 개체를 감지하는 데 사용됩니다. 키가 열 이름이고 값이 탐지된 개체 유형 목록인 맵을 반환합니다. 사용자 지정 마스킹은 AWS Glue 스크립트 내에서 반환된 후에 수행할 수 있습니다.

```
classifyColumns(frame: DynamicFrame,
                entityTypesToDetect: Seq[String],
                sampleFraction: Double = 0.1,
                thresholdFraction: Double = 0.1,
                detectionSensitivity: String = "LOW")
```

파라미터:

- frame - (유형:`DynamicFrame`) 처리할 데이터가 포함된 입력 DynamicFrame입니다.
- entityTypesToDetect - (type:`Seq[String]`) 탐지할 개체 유형 목록입니다. 관리 개체 유형 또는 사용자 지정 개체 유형일 수 있습니다.
- sampleFraction - (유형: `Double`, 기본값: 10%) PII 엔터티를 스캔할 때 샘플링할 데이터의 이름입니다.
- thresholdFraction- (유형: `Double`, 기본값: 10%): 열을 PII 데이터로 식별하기 위해 충족되어야 하는 데이터의 비율입니다.
- detectionSensitivity - (유형: `String`, 옵션: "LOW" or "HIGH", 기본: "LOW") 탐지 프로세스의 민감도를 지정합니다. 유효한 옵션은 "LOW" 또는 "HIGH"입니다. 제공되지 않은 경우 기본 민감도는 "LOW"으로 설정됩니다.

관리형 민감한 데이터 유형

글로벌 엔티티

데이터 형식	범주	설명
PERSON_NAME	일반	사람의 이름.
EMAIL	개인	이메일 주소.
IP_ADDRESS	컴퓨터	IP 주소
MAC_ADDRESS	개인	MAC 주소.

미국 데이터 유형

데이터 형식	설명
BANK_ACCOUNT	은행 계좌 번호. 특정 국가나 지역에 국한되지 않으나 미국 및 캐나다 계정 형식만 검색됩니다.
CREDIT_CARD	신용카드 번호.
PHONE_NUMBER	전화번호 특정 국가나 지역에 국한되지 않으나 현재 미국 및 캐나다 전화번호만 검색됩니다.
USA_ATIN	미국 국세청에서 발급한 미국 Adoption Taxpayer Identification Number(ATIN).
USA_CPT_CODE	CPT 코드(미국만 해당).
USA_DEA_NUMBER	DEA 번호(미국만 해당).
USA_DRIVING_LICENSE	운전면허증 번호(미국만 해당).
USA_HCPCS_CODE	HCPCS 코드(미국만 해당).
USA_HEALTH_INSURANCE_CLAIM_NUMBER	건강 보험 청구 번호(미국만 해당).
USA_ITIN	ITIN(미국 개인 또는 법인의 경우).
USA_MEDICARE_BENEFICIARY_IDENTIFIER	메디케어 수혜자 식별자(미국만 해당).

데이터 형식	설명
USA_NATIONAL_DRUG_CODE	NDC 코드(미국만 해당).
USA_NATIONAL_PROVIDER_IDENTIFIER	국가 공급자 식별 번호(미국만 해당).
USA_PASSPORT_NUMBER	여권 번호(미국인의 경우).
USA_PTIN	미국 국세청에서 발급한 미국 Preparer Tax Identification Number(PTIN).
USA_SSN	사회보장번호(미국인의 경우).

아르헨티나 데이터 형식

데이터 형식	설명
ARGENTINA_TAX_IDENTIFICATIO N_NUMBER	아르헨티나 세금 식별 번호. CUIT 또는 CUIL이라고도 합니다.

오스트레일리아 데이터 형식

데이터 형식	설명
AUSTRALIA_BUSINESS_NUMBER	호주 사업자 번호(ABN). 정부 및 지역 사회가 기업을 식별하기 위해 호주 사업자 등록부(ABN)에서 발급하는 고유 식별자.
AUSTRALIA_COMPANY_NUMBER	호주 회사 번호(ACN). 호주 증권투자위원회에서 발행한 고유 식별자.
AUSTRALIA_DRIVING_LICENSE	호주의 운전면허증 번호.
AUSTRALIA_MEDICARE_NUMBER	호주 메디케어 번호. 호주 건강보험위원회에서 발급한 개인 식별자.
AUSTRALIA_PASSPORT_NUMBER	호주 여권 번호.

데이터 형식	설명
AUSTRALIA_TAX_FILE_NUMBER	호주 납세자 번호(TFN). 세금 납부를 위해 호주 국세청(ATO)에서 납세자(개인, 회사 등)에게 발급하는 번호.

오스트리아 데이터 형식

데이터 형식	설명
AUSTRIA_DRIVING_LICENSE	운전면허증 번호(오스트리아만 해당).
AUSTRIA_PASSPORT_NUMBER	여권 번호(오스트리아만 해당).
AUSTRIA_SSN	사회보장번호(호주인의 경우).
AUSTRIA_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(오스트리아만 해당).
AUSTRIA_VALUE_ADDED_TAX	부가가치세(오스트리아만 해당).

발칸 반도 데이터 형식

데이터 형식	설명
BOSNIA_UNIQUE_MASTER_CITIZEN_NUMBER	보스니아-헤르체고비나 시민의 고유 마스터 시민 번호(JMBG).
KOSOVO_UNIQUE_MASTER_CITIZEN_NUMBER	코소보의 고유 마스터 시민 번호(JMBG).
MACEDONIA_UNIQUE_MASTER_CITIZEN_NUMBER	마케도니아의 고유 마스터 시민 번호.
MONTENEGRO_UNIQUE_MASTER_CITIZEN_NUMBER	몬테네그로의 고유 마스터 시민 번호(JMBG).

데이터 형식	설명
SERBIA_UNIQUE_MASTER_CITIZEN_NUMBER	세르비아의 고유 마스터 시민 번호(JMBG).
SERBIA_VALUE_ADDED_TAX	부가가치세(세르비아만 해당).
VOJVODINA_UNIQUE_MASTER_CITIZEN_NUMBER	보이보디나의 고유 마스터 시민 번호(JMBG).

벨기에 데이터 형식

데이터 형식	설명
BELGIUM_DRIVING_LICENSE	운전면허증 번호(벨기에만 해당).
BELGIUM_NATIONAL_IDENTIFICATION_NUMBER	벨기에 국가 번호(BNN).
BELGIUM_PASSPORT_NUMBER	여권 번호(벨기에만 해당).
BELGIUM_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(벨기에만 해당).
BELGIUM_VALUE_ADDED_TAX	부가가치세(벨기에만 해당).

브라질 데이터 형식

데이터 형식	설명
BRAZIL_BANK_ACCOUNT	은행 계좌 번호(브라질만 해당).
BRAZIL_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자(브라질만 해당).
BRAZIL_NATIONAL_REGISTRY_OF_LEGAL_ENTITIES_NUMBER	회사에 발급되는 식별 번호(CNPJ라고도 함, 브라질만 해당).

데이터 형식	설명
BRAZIL_NATURAL_PERSON_REGISTRY_NUMBER	자연인 등록 번호(CPF라고도 함).

불가리아 데이터 형식

데이터 형식	설명
BULGARIA_DRIVING_LICENSE	운전면허증 번호(불가리아만 해당).
BULGARIA_UNIFORM_CIVIL_NUMBER	국가 식별 번호 역할을 하는 통합 시민 번호(EGN).
BULGARIA_VALUE_ADDED_TAX	부가가치세(불가리아만 해당).

캐나다 데이터 형식

데이터 형식	설명
CANADA_DRIVING_LICENSE	운전면허증 번호(캐나다만 해당).
CANADA_GOVERNMENT_IDENTIFICATION_CARD_NUMBER	국가 식별자(캐나다만 해당).
CANADA_PASSPORT_NUMBER	여권 번호(캐나다만 해당).
CANADA_PERMANENT_RESIDENCE_NUMBER	영주권 번호(PR 카드 번호).
CANADA_PERSONAL_HEALTH_NUMBER	의료용 고유 식별자(PHN 번호).
CANADA_SOCIAL_INSURANCE_NUMBER	캐나다의 사회보험 번호(SIN).

칠레 데이터 형식

데이터 형식	설명
CHILE_DRIVING_LICENSE	운전면허증 번호(칠레만 해당).
CHILE_NATIONAL_IDENTIFICATION_NUMBER	칠레 국가 식별자(RUT 또는 RUN이라고도 함).

중국, 홍콩, 마카오, 대만 데이터 형식

데이터 형식	설명
CHINA_IDENTIFICATION	중국 식별자.
CHINA_LICENSE_PLATE_NUMBER	운전면허증 번호(중국만 해당).
CHINA_MAINLAND_TRAVEL_PERMIT_ID_HONG_KONG_MACAU	홍콩 및 마카오 거주자를 위한 본토 여행 허가증.
CHINA_MAINLAND_TRAVEL_PERMIT_ID_TAIWAN	중화인민공화국 정부(PRC)에서 발급한 대만 거주자를 위한 본토 여행 허가증.
CHINA_PASSPORT_NUMBER	여권 번호(중국만 해당).
CHINA_PHONE_NUMBER	전화번호(캐나다만 해당).
HONG_KONG_IDENTITY_CARD	홍콩 이민국에서 발급한 공식 신분증.
MACAU_RESIDENT_IDENTITY_CARD	마카오 거주자 신분증 또는 BIR은 마카오 신원 확인국에서 발급하는 공식 신분증.
TAIWAN_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자(대만만 해당).
TAIWAN_PASSPORT_NUMBER	여권 번호(대만만 해당).

콜롬비아 데이터 형식

데이터 형식	설명
COLOMBIA_PERSONAL_IDENTIFICATION_NUMBER	콜롬비아 출생 시 부여되는 고유 식별자.
COLOMBIA_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(콜롬비아만 해당).

크로아티아 데이터 형식

데이터 형식	설명
CROATIA_DRIVING_LICENSE	운전면허증 번호(크로아티아만 해당).
CROATIA_IDENTITY_NUMBER	국가 식별자(크로아티아에만 해당).
CROATIA_PASSPORT_NUMBER	여권 번호(크로아티아만 해당).
CROATIA_PERSONAL_IDENTIFICATION_NUMBER	개인 식별 번호(OIB).

키프로스 데이터 형식

데이터 형식	설명
CYPRUS_DRIVING_LICENSE	운전면허증 번호(키프로스만 해당).
CYPRUS_NATIONAL_IDENTIFICATION_NUMBER	키프로스 신분증.
CYPRUS_PASSPORT_NUMBER	여권 번호(키프로스만 해당).
CYPRUS_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(키프로스만 해당).
CYPRUS_VALUE_ADDED_TAX	부가가치세(키프로스만 해당).

체코 데이터 형식

데이터 형식	설명
CZECHIA_DRIVING_LICENSE	운전면허증 번호(체코만 해당).
CZECHIA_PERSONAL_IDENTIFICATION_NUMBER	개인 식별자 번호(체코만 해당).
CZECHIA_VALUE_ADDED_TAX	부가가치세(체코만 해당).

덴마크 데이터 형식

데이터 형식	설명
DENMARK_DRIVING_LICENSE	운전면허증 번호(덴마크만 해당).
DENMARK_PERSONAL_IDENTIFICATION_NUMBER	개인 식별자 번호(덴마크만 해당).
DENMARK_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(덴마크만 해당).
DENMARK_VALUE_ADDED_TAX	부가가치세(덴마크만 해당).

에스토니아 데이터 형식

데이터 형식	설명
ESTONIA_DRIVING_LICENSE	운전면허증 번호(에스토니아만 해당).
ESTONIA_PASSPORT_NUMBER	여권 번호(에스토니아만 해당).
ESTONIA_PERSONAL_IDENTIFICATION_CODE	개인 식별자 번호(에스토니아만 해당).
ESTONIA_VALUE_ADDED_TAX	부가가치세(에스토니아만 해당).

핀란드 데이터 형식

데이터 형식	설명
FINLAND_DRIVING_LICENSE	운전면허증 번호(핀란드만 해당).
FINLAND_HEALTH_INSURANCE_NUMBER	건강 보험 번호(핀란드만 해당).
FINLAND_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자 번호(핀란드만 해당).
FINLAND_PASSPORT_NUMBER	여권 번호(핀란드만 해당).
FINLAND_VALUE_ADDED_TAX	부가가치세(핀란드만 해당).

프랑스 데이터 형식

데이터 형식	설명
FRANCE_BANK_ACCOUNT	은행 계좌 번호(프랑스만 해당).
FRANCE_DRIVING_LICENSE	운전면허증 번호(프랑스만 해당).
FRANCE_HEALTH_INSURANCE_NUMBER	프랑스 건강보험 번호.
FRANCE_INSEE_CODE	프랑스 사회보장번호, SSN 번호 또는 NIR 번호.
FRANCE_NATIONAL_IDENTIFICATION_NUMBER	프랑스 국가 식별자 번호(CNI).
FRANCE_PASSPORT_NUMBER	여권 번호(프랑스만 해당).
FRANCE_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(프랑스만 해당).
FRANCE_VALUE_ADDED_TAX	부가가치세(프랑스만 해당).

독일 데이터 형식

데이터 형식	설명
GERMANY_BANK_ACCOUNT	은행 계좌 번호(독일만 해당).
GERMANY_DRIVING_LICENSE	운전면허증 번호(독일만 해당).
GERMANY_PASSPORT_NUMBER	여권 번호(독일만 해당).
GERMANY_PERSONAL_IDENTIFICATION_NUMBER	개인 식별 번호(독일만 해당).
GERMANY_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(독일만 해당).
GERMANY_VALUE_ADDED_TAX	부가가치세(독일만 해당).

그리스 데이터 형식

데이터 형식	설명
GREECE_DRIVING_LICENSE	운전면허증 번호(그리스만 해당).
GREECE_PASSPORT_NUMBER	여권 번호(그리스만 해당).
GREECE_SSN	사회보장번호(그리스인의 경우).
GREECE_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(그리스만 해당).
GREECE_VALUE_ADDED_TAX	부가가치세(그리스만 해당).

헝가리 데이터 형식

데이터 형식	설명
HUNGARY_DRIVING_LICENSE	운전면허증 번호(헝가리만 해당).
HUNGARY_PASSPORT_NUMBER	여권 번호(헝가리만 해당).
HUNGARY_SSN	사회보장번호(헝가리인의 경우).

데이터 형식	설명
HUNGARY_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(헝가리만 해당).
HUNGARY_VALUE_ADDED_TAX	부가가치세(헝가리만 해당).

아이슬란드 데이터 형식

데이터 형식	설명
ICELAND_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자(아이슬란드만 해당).
ICELAND_PASSPORT_NUMBER	여권 번호(아이슬란드만 해당).
ICELAND_VALUE_ADDED_TAX	부가가치세(아이슬란드만 해당).

인도 데이터 형식

데이터 형식	설명
INDIA_AADHAAR_NUMBER	인도 Unique Identification Authority에서 발급한 아드하르 식별 번호.
INDIA_PERMANENT_ACCOUNT_NUMBER	인도의 소득세 번호(PAN).

인도네시아 데이터 형식

데이터 형식	설명
INDONESIA_IDENTITY_CARD_NUMBER	국가 식별자(인도네시아만 해당).

아일랜드 데이터 형식

데이터 형식	설명
IRELAND_DRIVING_LICENSE	운전면허증 번호(아일랜드만 해당).
IRELAND_PASSPORT_NUMBER	여권 번호(아일랜드만 해당).
IRELAND_PERSONAL_PUBLIC_SERVICE_NUMBER	아일랜드 개인 공공 서비스 번호(PPS).
IRELAND_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(아일랜드만 해당).
IRELAND_VALUE_ADDED_TAX	부가가치세(아일랜드만 해당).

이스라엘 데이터 형식

데이터 형식	설명
ISRAEL_IDENTIFICATION_NUMBER	국가 식별자(이스라엘만 해당).

이탈리아 데이터 형식

데이터 형식	설명
ITALY_BANK_ACCOUNT	은행 계좌 번호(이탈리아만 해당).
ITALY_DRIVING_LICENSE	운전면허증 번호(이탈리아만 해당).
ITALY_FISCAL_CODE	식별자 번호(이탈리아 코디체 피스칼레(Codice Fiscale)라고도 함).
ITALY_PASSPORT_NUMBER	여권 번호(이탈리아만 해당).
ITALY_VALUE_ADDED_TAX	부가가치세(이탈리아만 해당).

일본 데이터 유형

데이터 형식	설명
JAPAN_BANK_ACCOUNT	일본 은행 계좌.
JAPAN_DRIVING_LICENSE	일본의 운전면허증 번호.
JAPAN_MY_NUMBER	조세 행정, 사회 보장 행정 및 재해 대응을 위해 사용하는 일본 시민 또는 법인의 고유 식별자
JAPAN_PASSPORT_NUMBER	일본 여권 번호.

한국 데이터 형식

데이터 형식	설명
KOREA_PASSPORT_NUMBER	여권 번호(한국만 해당).
KOREA_RESIDENCE_REGISTRATION_NUMBER_FOR_CITIZENS	한국 주민등록번호.
KOREA_RESIDENCE_REGISTRATION_NUMBER_FOR_FOREIGNERS	외국인을 위한 한국의 외국인 등록 번호.

라트비아 데이터 형식

데이터 형식	설명
LATVIA_DRIVING_LICENSE	운전면허증 번호(라트비아만 해당).
LATVIA_PASSPORT_NUMBER	여권 번호(라트비아만 해당).
LATVIA_PERSONAL_IDENTIFICATION_NUMBER	개인 식별자 번호(라트비아만 해당).
LATVIA_VALUE_ADDED_TAX	부가가치세(라트비아만 해당).

리히텐슈타인 데이터 형식

데이터 형식	설명
LIECHTENSTEIN_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자(리히텐슈타인만 해당).
LIECHTENSTEIN_PASSPORT_NUMBER	여권 번호(리히텐슈타인만 해당).
LIECHTENSTEIN_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(리히텐슈타인만 해당).

리투아니아 데이터 형식

데이터 형식	설명
LITHUANIA_DRIVING_LICENSE	운전면허증 번호(리투아니아만 해당).
LITHUANIA_PERSONAL_IDENTIFICATION_NUMBER	개인 식별자 번호(리투아니아만 해당).
LITHUANIA_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(리투아니아만 해당).
LITHUANIA_VALUE_ADDED_TAX	부가가치세(리투아니아만 해당).

룩셈부르크 데이터 형식

데이터 형식	설명
LUXEMBOURG_DRIVING_LICENSE	운전면허증 번호(룩셈부르크만 해당).
LUXEMBOURG_NATIONAL_INDIVIDUAL_NUMBER	국가 식별자(룩셈부르크만 해당).
LUXEMBOURG_PASSPORT_NUMBER	여권 번호(룩셈부르크만 해당).
LUXEMBOURG_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(룩셈부르크만 해당).
LUXEMBOURG_VALUE_ADDED_TAX	부가가치세(룩셈부르크만 해당).

말레이시아 데이터 형식

데이터 형식	설명
MALAYSIA_MYKAD_NUMBER	국가 식별자(말레이시아만 해당).
MALAYSIA_PASSPORT_NUMBER	여권 번호(말레이시아만 해당).

몰타 데이터 형식

데이터 형식	설명
MALTA_DRIVING_LICENSE	운전면허증 번호(몰타만 해당).
MALTA_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자(몰타만 해당).
MALTA_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(몰타만 해당).
MALTA_VALUE_ADDED_TAX	부가가치세(몰타만 해당).

멕시코 데이터 형식

데이터 형식	설명
MEXICO_CLABE_NUMBER	멕시코 Clave Bancaria Estandarizada(CLABE) 은행 번호.
MEXICO_DRIVING_LICENSE	운전면허증 번호(멕시코만 해당).
MEXICO_PASSPORT_NUMBER	여권 번호(멕시코만 해당).
MEXICO_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(멕시코만 해당).
MEXICO_UNIQUE_POPULATION_REGISTRY_CODE	멕시코의 고유 식별 코드(Clave Única de Registro de Población(CURP)).

네덜란드 데이터 형식

데이터 형식	설명
NETHERLANDS_CITIZEN_SERVICE_NUMBER	네덜란드의 시민 번호(BSN, burgerservicenummer).
NETHERLANDS_DRIVING_LICENSE	운전면허증 번호(네덜란드만 해당).
NETHERLANDS_PASSPORT_NUMBER	여권 번호(네덜란드만 해당).
NETHERLANDS_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(네덜란드만 해당).
NETHERLANDS_VALUE_ADDED_TAX	부가가치세(네덜란드만 해당).
NETHERLANDS_BANK_ACCOUNT	은행 계좌 번호(네덜란드만 해당).

뉴질랜드 데이터 형식

데이터 형식	설명
NEW_ZEALAND_DRIVING_LICENSE	운전면허증 번호(뉴질랜드만 해당).
NEW_ZEALAND_NATIONAL_HEALTH_INDEX_NUMBER	뉴질랜드 국가 건강 지수 번호.
NEW_ZEALAND_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(내국세 번호라고도 함, 뉴질랜드만 해당).

노르웨이 데이터 형식

데이터 형식	설명
NORWAY_BIRTH_NUMBER	노르웨이 국가 식별 번호.
NORWAY_DRIVING_LICENSE	운전면허증 번호(노르웨이만 해당).

데이터 형식	설명
NORWAY_HEALTH_INSURANCE_NUMBER	노르웨이 건강보험 번호.
NORWAY_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자 번호(노르웨이만 해당).
NORWAY_VALUE_ADDED_TAX	부가가치세(노르웨이만 해당).

필리핀 데이터 형식

데이터 형식	설명
PHILIPPINES_DRIVING_LICENSE	운전면허증 번호(필리핀만 해당).
PHILIPPINES_PASSPORT_NUMBER	여권 번호(필리핀만 해당).

폴란드 데이터 형식

데이터 형식	설명
POLAND_DRIVING_LICENSE	운전면허증 번호(폴란드만 해당).
POLAND_IDENTIFICATION_NUMBER	폴란드 식별자.
POLAND_PASSPORT_NUMBER	여권 번호(폴란드만 해당).
POLAND_REGON_NUMBER	REGON 식별자 번호(통계 식별 번호라고도 함).
POLAND_SSN	사회보장번호(폴란드인의 경우).
POLAND_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(폴란드만 해당).
POLAND_VALUE_ADDED_TAX	부가가치세(폴란드만 해당).

포르투갈 데이터 형식

데이터 형식	설명
PORTUGAL_DRIVING_LICENSE	운전면허증 번호(포르투갈만 해당).
PORTUGAL_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자 번호(포르투갈만 해당).
PORTUGAL_PASSPORT_NUMBER	여권 번호(포르투갈만 해당).
PORTUGAL_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(포르투갈만 해당).
PORTUGAL_VALUE_ADDED_TAX	부가가치세(포르투갈만 해당).

루마니아 데이터 형식

데이터 형식	설명
ROMANIA_DRIVING_LICENSE	운전면허증 번호(루마니아만 해당).
ROMANIA_NUMERICAL_PERSONAL_CODE	개인 식별자 번호(루마니아만 해당).
ROMANIA_PASSPORT_NUMBER	여권 번호(루마니아만 해당).
ROMANIA_VALUE_ADDED_TAX	부가가치세(루마니아만 해당).

싱가포르 데이터 형식

데이터 형식	설명
SINGAPORE_DRIVING_LICENSE	운전면허증 번호(싱가포르만 해당).
SINGAPORE_NATIONAL_REGISTRY_IDENTIFICATION_NUMBER	싱가포르 국가 등록 신분증.
SINGAPORE_PASSPORT_NUMBER	여권 번호(싱가포르만 해당).
SINGAPORE_UNIQUE_ENTITY_NUMBER	싱가포르 고유 법인 번호.

슬로바키아 데이터 형식

데이터 형식	설명
SLOVAKIA_DRIVING_LICENSE	운전면허증 번호(슬로바키아만 해당).
SLOVAKIA_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자 번호(슬로바키아만 해당).
SLOVAKIA_PASSPORT_NUMBER	여권 번호(슬로바키아만 해당).
SLOVAKIA_VALUE_ADDED_TAX	부가가치세(슬로바키아만 해당).

슬로베니아 데이터 형식

데이터 형식	설명
SLOVENIA_DRIVING_LICENSE	운전면허증 번호(슬로베니아만 해당).
SLOVENIA_PASSPORT_NUMBER	여권 번호(슬로베니아만 해당).
SLOVENIA_TAX_IDENTIFICATION_NUMBER	세금 식별 번호(슬로베니아만 해당).
SLOVENIA_UNIQUE_MASTER_CITIZEN_NUMBER	슬로베니아 시민의 고유 마스터 시민 번호(JMBG).
SLOVENIA_VALUE_ADDED_TAX	부가가치세(슬로베니아만 해당).

남아프리카 데이터 형식

데이터 형식	설명
SOUTH_AFRICA_PERSONAL_IDENTIFICATION_NUMBER	개인 식별자 번호(남아프리카 공화국만 해당).

일본 데이터 형식

데이터 형식	설명
SPAIN_BANK_ACCOUNT	은행 계좌 번호(스페인만 해당).
SPAIN_DNI	스페인의 국가 신분증(Documento Nacional de Identidad).
SPAIN_DRIVING_LICENSE	운전면허증 번호(스페인만 해당).
SPAIN_NIE	외국인 신분증 번호(NIE라고도 함, 스페인만 해당).
SPAIN_NIF	세금 식별 번호(NIF라고도 함, 스페인만 해당).
SPAIN_PASSPORT_NUMBER	여권 번호(스페인만 해당).
SPAIN_SSN	사회보장번호(스페인의 경우).
SPAIN_VALUE_ADDED_TAX	부가가치세(스페인만 해당).

스리랑카 데이터 형식

데이터 형식	설명
SRI_LANKA_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자(스리랑카만 해당).

스웨덴 데이터 형식

데이터 형식	설명
SWEDEN_DRIVING_LICENSE	운전면허증 번호(스웨덴만 해당).
SWEDEN_PASSPORT_NUMBER	여권 번호(스웨덴만 해당).
SWEDEN_PERSONAL_IDENTIFICATION_NUMBER	국가 식별자 번호(스웨덴만 해당).

데이터 형식	설명
SWEDEN_TAX_IDENTIFICATION_NUMBER	스웨덴 세금 식별 번호(personnummer).
SWEDEN_VALUE_ADDED_TAX	부가가치세(스웨덴만 해당).

스위스 데이터 형식

데이터 형식	설명
SWITZERLAND_AHV	스위스인의 사회보장번호(AHV).
SWITZERLAND_HEALTH_INSURANCE_NUMBER	스위스 건강보험 번호.
SWITZERLAND_PASSPORT_NUMBER	여권 번호(스위스만 해당).
SWITZERLAND_VALUE_ADDED_TAX	부가가치세(스위스만 해당).

태국 데이터 형식

데이터 형식	설명
THAILAND_PASSPORT_NUMBER	여권 번호(태국만 해당).
THAILAND_PERSONAL_IDENTIFICATION_NUMBER	개인 식별자 번호(태국만 해당).

터키 데이터 형식

데이터 형식	설명
TURKEY_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자 번호(튀르키예만 해당).
TURKEY_PASSPORT_NUMBER	여권 번호(튀르기예만 해당).

데이터 형식	설명
TURKEY_VALUE_ADDED_TAX	부가가치세(튀르키예만 해당).

우크라이나 데이터 형식

데이터 형식	설명
UKRAINE_INDIVIDUAL_IDENTIFICATION_NUMBER	고유 식별자(우크라이나만 해당).
UKRAINE_PASSPORT_NUMBER_DOMESTIC	국내 여권 번호(우크라이나만 해당).
UKRAINE_PASSPORT_NUMBER_INTERNATIONAL	국제 여권 번호(우크라이나만 해당).

아랍에미리트(UAE) 데이터 형식

데이터 형식	설명
UNITED_ARAB_EMIRATES_PERSONAL_NUMBER	개인 식별자 번호(UAE만 해당).

영국 데이터 유형

데이터 형식	설명
UK_BANK_ACCOUNT	영국 은행 계좌.
UK_BANK_SORT_CODE	영국 은행 지점 코드. 지점 코드(sort code)는 해당 통관 기관을 통해 각 국가 안의 은행 간에 송금을 라우팅하기 위해 사용하는 은행 코드입니다.
UK_DRIVING_LICENSE	영국 및 북아일랜드의 운전 면허증 번호(영국만 해당)

데이터 형식	설명
UK_ELECTORAL_ROLL_NUMBER	선거인단 번호(ERN)는 영국 선거 등록을 위해 개인에게 발급하는 식별 번호입니다. 이 번호의 형식은 영국 내각 사무처의 영국 정부 표준에 명시되어 있습니다.
UK_NATIONAL_HEALTH_SERVICE_NUMBER	국민 보건 서비스(NHS) 번호는 영국의 등록된 공중 보건 서비스 사용자에게 할당하는 고유 번호입니다.
UK_NATIONAL_INSURANCE_NUMBER	국민 보험 번호(NINO)는 영국(UK)에서 국민 보험 프로그램 또는 사회 보장 제도를 위해 개인을 식별하는 데 사용하는 번호입니다. 일부 경우에 NI No 또는 NINO라고도 합니다.
UK_PASSPORT_NUMBER	영국 여권 번호.
UK_UNIQUE_TAXPAYER_REFERENCE_NUMBER	영국(UK) 고유 납세자 참조(UTR) 번호. 영국 정부가 과세 제도를 관리하기 위해 사용하는 식별자.
UK_VALUE_ADDED_TAX	VAT는 최종 소비자가 부담하는 소비세입니다. VAT는 제조 및 유통 과정에서 이뤄지는 각 거래에 대해 지불합니다. 영국의 경우 VAT 번호는 사업체가 설립된 지역의 VAT 사무소에서 발급합니다.
UK_PHONE_NUMBER	영국 전화번호.

베네수엘라 데이터 형식

데이터 형식	설명
VENEZUELA_DRIVING_LICENSE	운전면허증 번호(베네수엘라만 해당).
VENEZUELA_NATIONAL_IDENTIFICATION_NUMBER	국가 식별자 번호(베네수엘라만 해당).

데이터 형식	설명
VENEZUELA_VALUE_ADDED_TAX	부가가치세(베네수엘라만 해당).

세분화된 데이터 탐지 사용하기

> **Note**
>
> 세분화된 동작은 AWS Glue 3.0 및 4.0에서만 사용할 수 있습니다. 여기에는 AWS Glue Studio 경험도 포함됩니다. 2.0에서는 영구 감사 로그 변경도 사용할 수 없습니다.
> 모든 AWS Glue Studio 3.0 및 4.0 비주얼 작업에는 세분화된 작업 API를 자동으로 사용하는 스크립트가 생성됩니다.

민감 데이터 탐지 변환은 사용자가 정의하거나 AWS Glue에서 사전 정의한 엔터티를 탐지, 마스크 또는 제거하는 기능을 제공합니다. 또한 세분화된 액션을 통해 엔터티별로 특정 액션을 적용할 수 있습니다. 추가 혜택은 다음과 같습니다.

- 데이터가 감지되는 즉시 조치가 적용되므로 성능이 개선되었습니다.
- 특정 열을 포함하거나 제외하는 옵션.
- 부분 마스킹을 사용할 수 있습니다. 이렇게 하면 전체 문자열을 마스킹하는 대신 감지된 민감한 데이터 엔터티를 부분적으로 마스킹할 수 있습니다. 오프셋과 정규식이 있는 단순 매개 변수가 모두 지원됩니다.

다음은 다음 섹션에서 참조할 샘플 작업에 사용되는 민감한 데이터 탐지 API와 세분화된 조치의 코드 스니펫입니다.

Detect API- 세분화된 작업에는 새 `detectionParameters` 파라미터가 사용됩니다.

```
def detect(
    frame: DynamicFrame,
    detectionParameters: JsonOptions,
    outputColumnName: String = "DetectedEntities",
    detectionSensitivity: String = "LOW"
): DynamicFrame = {}
```

민감한 데이터 탐지 API와 세밀한 조치 사용

detect를 사용하는 민감한 데이터 탐지 API는 제공된 데이터를 분석하고, 행 또는 열이 민감한 데이터 항목 유형인지 확인하고 각 엔터티 유형에 대해 사용자가 지정한 작업을 실행합니다.

탐지 API를 사용하여 세부적인 조치를 취하십시오

detect API를 사용하고 `outputColumnName` 및 `detectionParameters`를 지정합니다.

```
   object GlueApp {
     def main(sysArgs: Array[String]) {

       val spark: SparkContext = new SparkContext()
       val glueContext: GlueContext = new GlueContext(spark)

       // @params: [JOB_NAME]
       val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
       Job.init(args("JOB_NAME"), glueContext, args.asJava)

       // Script generated for node S3 bucket. Creates DataFrame from data stored in
S3.
       val S3bucket_node1 =
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar":
"\"", "withHeader": true, "separator": ",", "optimizePerformance": false}"""),
connectionType="s3", format="csv", options=JsonOptions("""{"paths":
["s3://189657479688-ddevansh-pii-test-bucket/tiny_pii.csv"], "recurse": true}"""),
transformationContext="S3bucket_node1").getDynamicFrame()

       // Script generated for node Detect Sensitive Data. Will run detect API for the
DataFrame
       // detectionParameter contains information on which EntityType are being
detected
       // and what actions are being applied to them when detected.
       val DetectSensitiveData_node2 = EntityDetector.detect(
           frame = S3bucket_node1,
           detectionParameters = JsonOptions(
            """
               {
                   "PHONE_NUMBER": [
                       {
                           "action": "PARTIAL_REDACT",
                           "actionOptions": {
                               "numLeftCharsToExclude": "3",
```

```
                                "numRightCharsToExclude": "4",
                                "redactChar": "#"
                            },
                            "sourceColumnsToExclude": [ "Passport No", "DL NO#" ]
                        }
                    ],
                    "USA_PASSPORT_NUMBER": [
                        {
                            "action": "SHA256_HASH",
                            "sourceColumns": [ "Passport No" ]
                        }
                    ],
                    "USA_DRIVING_LICENSE": [
                        {
                            "action": "REDACT",
                            "actionOptions": {
                                "redactText": "USA_DL"
                            },
                            "sourceColumns": [ "DL NO#" ]
                        }
                    ]

                }
            """
            ),
            outputColumnName = "DetectedEntities"
        )

        // Script generated for node S3 bucket. Store Results of detect to S3 location
        val S3bucket_node3 = glueContext.getSinkWithFormat(connectionType="s3",
options=JsonOptions("""{"path": "s3://amzn-s3-demo-bucket/test-output/",
"partitionKeys": []}"""), transformationContext="S3bucket_node3",
format="json").writeDynamicFrame(DetectSensitiveData_node2)

        Job.commit()
      }
```

위 스크립트는 Amazon S3의 위치에서 DataFrame을 생성한 다음 `detect` API를 실행합니다. `detect` API는 Glue 객체로 표시되는 필드 `detectionParameters`(해당 엔터티에 사용할 모든 작업 설정을 나열하는 엔터티 이름 맵)를 AWS Glue의 `JsonOptions` 객체로 표시해야 하므로 API의 기능을 확장할 수도 있습니다.

엔터티별로 지정된 각 작업에 대해 엔터티/액션 조합을 적용할 모든 열 이름 목록을 입력합니다. 이렇게 하면 데이터 세트의 모든 열을 감지하도록 항목을 사용자 지정하고 특정 열에 없는 것으로 알려진 항목은 건너뛸 수 있습니다. 또한 이러한 엔터티에 대해 불필요한 탐지 호출을 수행하지 않아도 되므로 작업 성능이 향상되고 각 열과 엔터티 조합에 고유한 작업을 수행할 수 있습니다.

detectionParameters를 자세히 살펴보면 샘플 작업에는 세 가지 엔터티 유형이 있습니다. Phone Number, USA_PASSPORT_NUMBER, USA_DRIVING_LICENSE입니다. 이 같은 각 개체 유형에 대해 AWS Glue는 PARTIAL_REDACT, SHA256_HASH, REDACT, DETECT 같은 서로 다른 작업을 실행합니다. 각 개체 유형도 sourceColumns에 적용하거나 탐지된 경우 sourceColumnsToExclude에 적용해야 합니다.

> **Note**
>
> 열당 내부 편집 작업(PARTIAL_REDACT, SHA256_HASH, 또는REDACT) 하나만 사용할 수 있지만 해당 DETECT 작업은 이러한 모든 작업에 사용할 수 있습니다.

detectionParameters 필드 레이아웃은 다음과 같습니다.

```
ENTITY_NAME -> List[Actions]
{
 "ENTITY_NAME": [{
  Action, // required
  ColumnSpecs,
  ActionOptionsMap
    }],
    "ENTITY_NAME2": [{
  ...
    }]
}
```

actions 및 actionOptions의 유형은 다음과 같습니다.

```
DETECT
{
    # Required
    "action": "DETECT",
    # Optional, depending on action chosen
    "actionOptions": {
```

```
        // There are no actionOptions for DETECT
    },
    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

SHA256_HASH
{
    # Required
    "action": "SHA256_HASH",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // There are no actionOptions for SHA256_HASH
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

REDACT
{
    # Required
    "action": "REDACT",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // The text that is being replaced
        "redactText": "USA_DL"
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
```

```
        "COL_5"
    ]
}

PARTIAL_REDACT
{
    # Required
    "action": "PARTIAL_REDACT",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // number of characters to not redact from the left side
        "numLeftCharsToExclude": "3",
        // number of characters to not redact from the right side
        "numRightCharsToExclude": "4",
        // the partial redact will be made with this redacted character
        "redactChar": "#",
        // regex pattern for partial redaction
        "matchPattern": "[0-9]"
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}
```

스크립트가 실행되면 결과가 지정된 Amazon S3 위치에 출력됩니다. Amazon S3에서 데이터를 볼 수 있지만 선택한 엔터티 유형은 선택한 작업에 따라 구분됩니다. 이 경우 다음과 같은 행이 있을 것입니다.

```
{
    "Name": "Colby Schuster",
    "Address": "39041 Antonietta Vista, South Rodgerside, Nebraska 24151",
    "Car Owned": "Fiat",
    "Email": "Kitty46@gmail.com",
    "Company": "O'Reilly Group",
    "Job Title": "Dynamic Functionality Facilitator",
    "ITIN": "991-22-2906",
    "Username": "Cassandre.Kub43",
```

```
    "SSN": "914-22-2906",
    "DOB": "2020-08-27",
    "Phone Number": "1-2#######1718",
    "Bank Account No": "69741187",
    "Credit Card Number": "6441-6289-6867-2162-2711",
    "Passport No": "94f311e93a623c72ccb6fc46cf5f5b0265ccb42c517498a0f27fd4c43b47111e",
    "DL NO#": "USA_DL"
}
```

위 스크립트에서는 Phone Number를 사용하여 부분적으로 #으로 수정되었습니다. Passport No가 SHA256 해시로 변경되었습니다. DL NO# 는 미국 운전면허증 번호로 인식되어 detectionParameters에 명시된 대로 “USA_DL”로 수정되었습니다.

> **Note**
>
> classifyColumns API는 API의 특성상 세분화된 작업에는 사용할 수 없습니다. 이 API는 열 샘플링(사용자가 조정 가능하지만 기본값 사용)을 수행하여 탐지를 더 빠르게 수행합니다. 이러한 이유로 세밀한 작업을 수행하려면 모든 값을 반복해야 합니다.

영구 감사 로그

세분화된 작업과 함께 도입된 새로운 기능(일반 API를 사용할 때도 사용 가능)은 영구 감사 로그의 존재입니다. 현재 detect API를 실행하면 PII 탐지 메타데이터가 포함된 추가 열(DetectedEntities가 기본값이지만 outputColumnName을 통해 사용자 지정 가능) 매개변수가 추가됩니다. 이제 여기에는,DETECT, PARTIAL_REDACT, SHA256_HASH, REDACT 중 하나인 “actionUsed” 메타데이터 키가 있습니다.

```
"DetectedEntities": {
    "Credit Card Number": [
        {
            "entityType": "CREDIT_CARD",
            "actionUsed": "DETECT",
            "start": 0,
            "end": 19
        }
    ],
    "Phone Number": [
        {
            "entityType": "PHONE_NUMBER",
```

```
            "actionUsed": "REDACT",
            "start": 0,
            "end": 14
        }
    ]
}
```

예를 들어 detect(entityTypesToDetect, outputColumnName)와 같이 세밀한 조치 없이 API를 사용하는 고객도 결과 데이터 프레임에서 이 영구 감사 로그를 볼 수 있습니다.

세분화된 조치가 포함된 API를 사용하는 고객은 수정 여부에 관계없이 모든 작업을 볼 수 있습니다. 예시:

```
+--------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
| Credit Card Number  |  Phone Number  |
                                              DetectedEntities
                                                                    |
+--------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
| 622126741306XXXX    | +12#####7890   | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":16}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":12}]}} |
| 6221 2674 1306 XXXX | +12#######7890 | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":19}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":14}]}} |
| 6221-2674-1306-XXXX | 22#######7890  | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":19}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":14}]}} |
+--------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
```

DetectedEntities 열을 보지 않으려면 사용자 지정 스크립트에 추가 열을 놓기만 하면 됩니다.

AWS Glue Visual Job API

AWS Glue는 고객이 시각적 단계 워크플로를 나타내는 JSON 객체의 AWS Glue API를 사용하여 데이터 통합 작업을 생성할 수 있게 해주는 API를 제공합니다. 그러면 고객은 AWS Glue Studio의 시각적 편집기를 사용하여 이러한 작업을 수행할 수 있습니다.

시각적 작업 API 데이터 형식에 대한 자세한 내용은 Visual Job API를 참조하세요.

주제

- API 설계 및 CRUD API
- 시작하기
- 시각적 작업 제한 사항

API 설계 및 CRUD API

이제 CreateJob 및 UpdateJob API 에서 선택적 추가 파라미터인 codeGenConfigurationNodes를 지원합니다. 이 필드에 비어 있지 않은 JSON 구조를 제공하면 생성된 작업에 대한 DAG가 AWS Glue Studio에 등록되고 관련 코드가 생성됩니다. 작업 생성 시 이 필드의 Null 값이나 빈 문자열은 무시됩니다.

codeGenConfigurationNodes 필드에 대한 업데이트는 CreateJob과 비슷한 방식으로 UpdateJob AWS Glue API를 통해 수행됩니다. DAG가 원하는 대로 변경된 UpdateJob에 전체 필드를 지정해야 합니다. 제공된 Null 값은 무시되며 DAG에 대한 업데이트가 수행되지 않습니다. 빈 구조나 문자열을 제공하면 codeGenConfigurationNodes가 비어 있는 것으로 설정되고 이전 DAG는 모두 제거됩니다. GetJob API는 DAG를 반환합니다(존재하는 경우). DeleteJob API는 연결된 모든 DAG도 삭제합니다.

시작하기

작업을 생성하려면 CreateJob 작업을 사용합니다. `CreateJob` 요청 입력에는 JSON에서 DAG 객체를 지정할 수 있는 추가 필드인 'codeGenConfigurationNodes'가 있습니다.

다음과 같은 사항에 유의하세요.

- 'codeGenConfigurationNodes' 필드는 nodeId를 노드에 매핑한 것입니다.
- 각 노드는 노드 종류를 식별하는 키로 시작합니다.
- 노드는 단일 유형만 가능하므로 하나의 키만 지정할 수 있습니다.
- 입력 필드에는 현재 노드의 상위 노드가 포함됩니다.

다음은 CreateJob 입력의 JSON 표현입니다.

```
{
  "node-1": {
    "S3CatalogSource": {
      "Table": "csvFormattedTable",
      "PartitionPredicate": "",
      "Name": "S3 bucket",
      "AdditionalOptions": {},
      "Database": "myDatabase"
    }
  },
  "node-3": {
    "S3DirectTarget": {
      "Inputs": ["node-2"],
      "PartitionKeys": [],
      "Compression": "none",
      "Format": "json",
      "SchemaChangePolicy": { "EnableUpdateCatalog": false },
      "Path": "",
      "Name": "S3 bucket"
    }
  },
  "node-2": {
    "ApplyMapping": {
      "Inputs": ["node-1"],
      "Name": "ApplyMapping",
      "Mapping": [
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader1",
          "FromPath": ["myheader1"]
        },
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader2",
          "FromPath": ["myheader2"]
        },
        {
          "FromType": "long",
```

```
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader3",
          "FromPath": ["myheader3"]
        }
      ]
    }
  }
}
```

작업 업데이트 및 가져오기

UpdateJob에도 ‘codeGenConfigurationNodes’ 필드가 있으므로 입력 형식은 동일합니다. UpdateJob 작업을 참조하세요.

GetJob 작업도 동일한 형식으로 ‘codeGenConfigurationNodes’ 필드를 반환합니다. GetJob 작업을 참조하세요.

시각적 작업 제한 사항

'codeGenConfigurationNodes' 파라미터가 기존 API에 추가되었으므로 해당 API의 모든 제한 사항이 상속됩니다. 또한 codeGenConfigurationNodes와 일부 노드는 크기가 제한됩니다. 자세한 내용은 작업 구조 를 참조하세요.

Ray 스크립트 프로그래밍

AWS Glue를 사용하면 Ray 스크립트를 쉽게 쓰고 실행할 수 있습니다. 이 섹션에서는 AWS Glue for Ray에서 사용할 수 있는 지원되는 Ray 기능을 설명합니다. Python에서 Ray 스크립트를 프로그래밍합니다.

사용자 정의 스크립트는 작업 정의의 `Runtime` 필드에 정의된 Ray 버전과 호환되어야 합니다. 작업 API에서 `Runtime`에 대한 자세한 내용은 the section called “작업” 섹션을 참조하세요. 각 런타임 환경에 대한 자세한 내용은 the section called “지원되는 Ray 런타임 환경” 섹션을 참조하세요.

주제

- 자습서: AWS Glue for Ray에서 ETL 스크립트 작성
- AWS Glue for Ray에서 Ray Core 및 Ray Data 사용
- Ray 작업에 파일 및 Python 라이브러리 제공
- Ray 작업의 데이터에 연결

자습서: AWS Glue for Ray에서 ETL 스크립트 작성

Ray는 기본적으로 Python에서 배포된 작업을 작성하고 규모를 조정할 수 있는 기능을 제공합니다. AWS Glue for Ray는 작업과 대화형 세션 모두에서 액세스할 수 있는 서버리스 Ray 환경을 제공합니다(Ray 대화형 세션은 평가판 단계임). AWS Glue 작업 시스템은 일정에 따라, 트리거 또는 AWS Glue 콘솔을 통해 작업을 관리하고 실행할 수 있는 일관된 방법을 제공합니다.

이러한 AWS Glue 도구를 결합하면 추출, 전환, 적재(ETL) 워크로드에 사용할 수 있는 강력한 도구 모음이 생성되며, AWS Glue의 사용 사례에 많이 사용됩니다. 이 자습서에서는 이 솔루션을 구성하는 기본 사항을 학습합니다.

또한 ETL 워크로드에서 AWS Glue for Spark를 사용할 수 있도록 지원합니다. AWS Glue for Spark 스크립트 작성에 대한 자습서는 the section called “자습서: Spark 스크립트 작성” 섹션을 참조하세요. 사용 가능한 엔진에 대한 자세한 내용은 the section called “AWS Glue for Spark 및 AWS Glue for Ray” 섹션을 참조하세요. Ray는 분석, 기계 학습(ML) 및 애플리케이션 개발 분야에서 다양한 종류의 작업을 처리할 수 있습니다.

이 자습서에서는 Amazon Simple Storage Service(S3)에 호스팅된 CSV 데이터 세트를 추출, 전환 및 적재합니다. 먼저 퍼블릭 Amazon S3 버킷에 저장된 New York City Taxi and Limousine Commission (TLC) Trip Record Data 데이터 세트로 시작합니다. 이 데이터 세트에 대한 자세한 내용은 AWS의 오픈 데이터 레지스트리를 참조하세요.

Ray Data 라이브러리에서 사용할 수 있는 미리 정의된 변환을 사용하여 데이터를 변환합니다. Ray Data는 Ray에서 설계한 데이터 세트 준비 라이브러리이며 AWS Glue for Ray 환경에 기본적으로 포함되어 있습니다. 기본적으로 포함된 라이브러리에 대한 자세한 내용은 the section called “Ray 작업과 함께 제공되는 모듈” 섹션을 참조하세요. 그런 다음 사용자가 제어하는 Amazon S3 버킷에 변환된 데이터를 작성합니다.

사전 조건 - 이 자습서에서는 AWS Glue 및 Amazon S3에 액세스할 수 있는 AWS 계정이 필요합니다.

1단계: Amazon S3에 출력 데이터를 보관할 버킷 생성

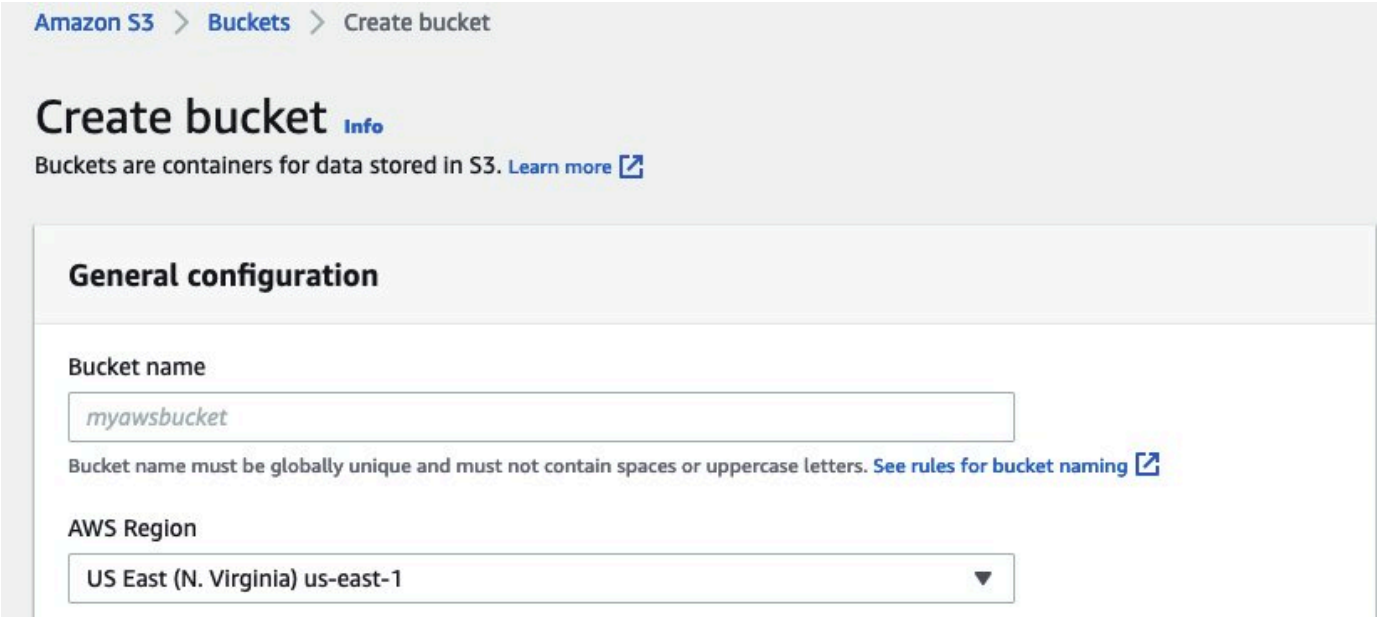
이 자습서에서 생성한 데이터의 싱크 역할을 하기 위해 사용자가 제어하는 Amazon S3 버킷이 필요합니다. 다음 절차에 따라 이 버킷을 생성할 수 있습니다.

> **Note**
>
> 사용자가 제어하는 기존 버킷에 데이터를 쓰려면 이 단계를 건너뛰면 됩니다. 이후 단계에서 사용하기 위해 기존 버킷 이름(*yourBucketName*)을 기록해둡니다.

Ray 작업 출력을 위한 버킷을 생성하려면

- Amazon S3 사용 설명서의 버킷 생성에 나와 있는 단계에 따라 버킷을 생성합니다.
  - 버킷 이름을 선택할 때는 *yourBucketName*을 기록해둡니다. 이 이름은 이후 단계에서 참조합니다.
  - 다른 구성의 경우 Amazon S3 콘솔에서 제공하는 제안된 설정도 이 자습서에서 작동할 수 있습니다.

  예를 들어 Amazon S3 콘솔의 버킷 생성 대화 상자는 다음과 비슷할 수 있습니다.



2단계: Ray 작업에 대한 IAM 역할 생성

작업에는 다음을 포함하는 AWS Identity and Access Management(IAM) 역할이 필요합니다.

- `AWSGlueServiceRole` 관리형 정책에서 부여한 권한. 이는 AWS Glue 작업을 실행하는 데 필요한 기본 권한입니다.
- `nyc-tlc/*` Amazon S3 리소스에 대한 `Read` 액세스 수준 권한.
- *`yourBucketName`*`/*` Amazon S3 리소스에 대한 `Write` 액세스 수준 권한.
- `glue.amazonaws.com` 보안 주체가 역할을 수임할 수 있도록 하는 신뢰 관계.

다음 절차에 따라 이 역할을 생성할 수 있습니다.

AWS Glue for Ray 작업에 대한 IAM 역할을 생성하려면

> **Note**
>
> 다양한 절차에 따라 IAM 역할을 생성할 수 있습니다. IAM 리소스를 프로비저닝하는 방법에 대한 자세한 내용이나 옵션은 AWS Identity and Access Management 설명서를 참조하세요.

1. IAM 사용 설명서의 시각적 편집기를 사용하여 IAM 정책(콘솔) 생성에 나온 단계를 수행하여 이전에 설명한 Amazon S3 권한을 정의하는 정책을 생성합니다.
   - 서비스를 선택할 때 Amazon S3를 선택합니다.
   - 정책에 대한 권한을 선택할 때 앞서 언급한 다음 리소스에 대해 다음 작업 세트를 연결합니다.
     - `nyc-tlc/*` Amazon S3 리소스에 대한 읽기 액세스 수준 권한.
     - *`yourBucketName`*`/*` Amazon S3 리소스에 대한 쓰기 액세스 수준 권한.
   - 정책 이름을 선택할 때는 *`YourPolicyName`*을 기록해둡니다. 이 이름은 이후 단계에서 참조합니다.
2. IAM 사용 설명서의 AWS 서비스에 대한 역할 생성(콘솔)에 나온 단계에 따라 AWS Glue for Ray 작업의 역할을 생성합니다.
   - 신뢰할 수 있는 AWS 서비스 엔터티를 선택할 때 `Glue`를 선택합니다. 그러면 작업에 필요한 신뢰 관계가 자동으로 채워집니다.
   - 권한 정책에 대한 정책을 선택할 때 다음 정책을 연결합니다.
     - `AWSGlueServiceRole`
     - *`YourPolicyName`*
   - 역할 이름을 선택할 때는 *`YourRoleName`*을 기록해둡니다. 이 이름은 이후 단계에서 참조합니다.

3단계: AWS Glue for Ray 작업 생성 및 실행

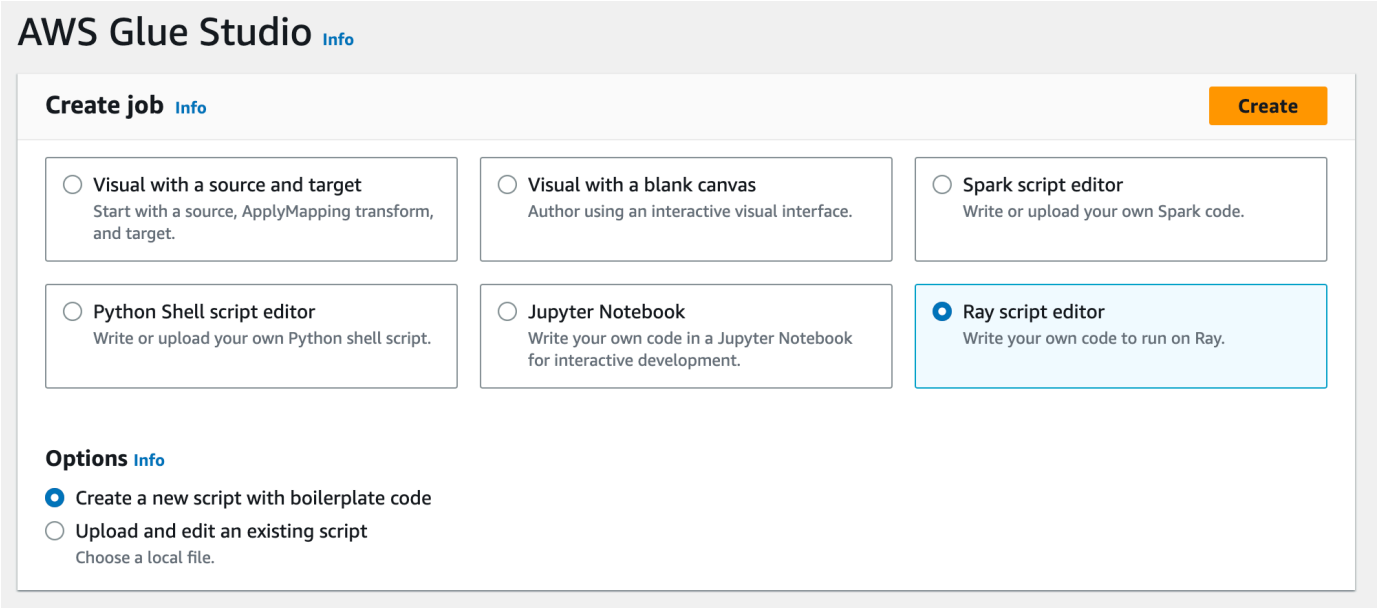
이 단계에서는 AWS Management Console을 사용하여 AWS Glue 작업을 생성하고 샘플 스크립트를 함께 제공하며 작업을 실행합니다. 작업을 생성하면 콘솔에 Ray 스크립트를 저장, 구성 및 편집할 수 있는 위치가 생성됩니다. 작업 생성에 대한 자세한 내용은 the section called “콘솔로 로그인합니다” 섹션을 참조하세요.

이 자습서에서는 다음과 같은 ETL 시나리오를 다룹니다. New York City TLC Trip Record 데이터 세트에서 2022년 1월 기록을 읽고, 기존 열의 데이터를 결합하여 데이터 세트에 새 열(tip_rate)을 추가한 다음, 현재 분석과 관련이 없는 열을 여러 개 제거하고 결과를 *yourBucketName*에 쓰려고 합니다. 다음 Ray 스크립트에서 다음 단계를 수행합니다.

```
import ray
import pandas
from ray import data

ray.init('auto')

ds = ray.data.read_csv("s3://nyc-tlc/opendata_repo/opendata_webconvert/yellow/
yellow_tripdata_2022-01.csv")

# Add the given new column to the dataset and show the sample record after adding a new
 column
ds = ds.add_column( "tip_rate", lambda df: df["tip_amount"] / df["total_amount"])

# Dropping few columns from the underlying Dataset
ds = ds.drop_columns(["payment_type", "fare_amount", "extra", "tolls_amount",
 "improvement_surcharge"])

ds.write_parquet("s3://yourBucketName/ray/tutorial/output/")
```

AWS Glue for Ray 작업을 생성하고 실행하려면

1. AWS Management Console에서 AWS Glue 랜딩 페이지로 이동합니다.
2. 측면 탐색 창에서 ETL 작업을 선택합니다.
3. 다음 그림과 같이 작업 생성에서 Ray 스크립트 편집기를 선택한 다음 생성을 선택합니다.



4. 스크립트의 전체 텍스트를 스크립트 창에 붙여넣고 기존 텍스트를 모두 바꿉니다.
5. 작업 세부 정보로 이동하여 IAM 역할 속성을 *YourRoleName*으로 설정합니다.
6. 저장을 선택한 다음 실행을 선택합니다.

4단계: 출력 검사

AWS Glue 작업을 실행한 후에는 출력이 이 시나리오의 예상과 일치하는지 확인해야 합니다. 다음 구문을 사용하면 이 작업을 수행할 수 있습니다.

Ray 작업이 성공적으로 실행되었는지 확인하려면

1. 작업 세부 정보 페이지에서 실행으로 이동합니다.
2. 몇 분 후 실행 상태가 성공인 실행이 표시됩니다.
3. Amazon S3 콘솔(https://console.aws.amazon.com/s3/)로 이동하여 *yourBucketName*을 검사합니다. 출력 버킷에 쓴 파일이 표시됩니다.
4. Parquet 파일을 읽고 해당 콘텐츠를 확인합니다. 기존 도구를 사용하여 이 작업을 수행할 수 있습니다. Parquet 파일을 검증하는 프로세스가 없는 경우 Spark 또는 Ray(평가판)를 사용하여 AWS Glue 대화형 세션이 있는 AWS Glue 콘솔에서 이 작업을 수행할 수 있습니다.

   대화형 세션에서는 선택한 엔진에 따라 기본적으로 제공되는 Ray Data, Spark 또는 pandas 라이브러리에 액세스할 수 있습니다. 파일 콘텐츠를 확인하려면 해당 라이브러리에서 사용할 수 있는 일반적인 검사 방법(예: `count`, `schema` 및 `show`)을 사용할 수 있습니다. 콘솔에서의 대화형 세션에 대한 자세한 내용은 AWS Glue Studio 및 AWS Glue에서 노트북 사용을 참조하세요.

파일이 버킷에 기록된 것을 확인했으므로 출력에 문제가 있는 경우 IAM 구성과 관련이 없다고 비교적 확신할 수 있습니다. 관련 파일에 액세스할 수 있도록 *yourRoleName*으로 세션을 구성합니다.

예상된 결과를 얻지 못하면 이 안내서의 문제 해결 콘텐츠를 살펴보고 오류의 원인을 파악하고 해결합니다. 작업 실행 오류 상태를 해석하려면 the section called “작업 실행 상태” 섹션을 참조하세요. AWS Glue 문제 해결 장에서 문제 해결 콘텐츠를 찾을 수 있습니다. Ray 작업과 관련된 특정 오류에 대해서는 문제 해결 장의 the section called “Ray 오류 해결” 섹션을 참조하세요.

다음 단계

지금까지 AWS Glue for Ray를 사용하여 ETL 프로세스를 처음부터 끝까지 확인하고 수행했습니다. 다음 리소스를 통해 AWS Glue for Ray가 대규모로 데이터를 변환하고 해석하기 위해 제공하는 도구를 이해할 수 있습니다.

- Ray의 작업 모델에 대한 자세한 내용은 the section called “AWS Glue for Ray에서 Ray Core 및 Ray Data 사용” 섹션을 참조하세요. Ray 작업 사용에 대한 더 많은 경험을 쌓으려면 Ray Core 설명서의 예제를 따르세요. Ray 설명서의 Ray Core: Ray Tutorials and Examples (2.4.0)를 참조하세요.
- AWS Glue for Ray에서 사용 가능한 데이터 관리 라이브러리에 대한 지침은 the section called “데이터에 연결” 섹션을 참조하세요. Ray Data를 사용하여 데이터 세트를 변환하고 작성하는 데 더 많은 경험을 쌓으려면 Ray Data 설명서의 예제를 따르세요. Ray Data: Examples (2.4.0)를 참조하세요.
- AWS Glue for Ray 작업 구성에 대한 자세한 내용은 the section called “Ray 작업 사용” 섹션을 참조하세요.
- AWS Glue for Ray 스크립트 작성에 대한 자세한 내용은 이 섹션의 설명서를 계속 읽어보세요.

AWS Glue for Ray에서 Ray Core 및 Ray Data 사용

Ray는 클러스터 전체에 작업을 분산하여 Python 스크립트 규모를 스케일 업하는 프레임워크입니다. Ray는 다양한 문제에 대한 솔루션으로 사용할 수 있으므로 Ray는 특정 작업을 최적화하는 라이브러리를 제공합니다. AWS Glue에서는 Ray를 사용하여 대규모 데이터 세트를 변환하는 데 중점을 둡니다. AWS Glue에서는 이 작업을 용이하게 하기 위해 Ray Data 및 Ray Core의 일부에 대한 지원을 제공합니다.

Ray Core란 무엇인가요?

분산 애플리케이션을 구축하는 첫 번째 단계는 동시에 수행할 수 있는 작업을 식별하고 정의하는 것입니다. Ray Core에는 동시에 수행할 수 있는 작업을 정의하는 데 사용하는 Ray의 일부 기능이 포함되어

있습니다. Ray는 Ray에서 제공하는 도구를 익히는 데 사용할 수 있는 참조 정보 및 빠른 시작 정보를 제공합니다. 자세한 내용은 What is Ray Core? 및 Ray Core Quick Start를 참조하세요. Ray에서 동시 작업을 효과적으로 정의하는 방법에 대한 자세한 내용은 Tips for first-time users를 참조하세요.

> **Ray 작업 및 액터**
>
> AWS Glue for Ray 설명서에서는 Ray의 핵심 개념인 작업 및 액터를 언급할 수 있습니다. Ray는 Python 함수와 클래스를 분산 컴퓨팅 시스템의 구성 요소로 사용합니다. Python 함수와 변수가 클래스에서 사용될 때 '메서드'와 '속성' 역할을 하는 것과 마찬가지로, Ray에서 작업자에게 코드를 보낼 때 사용되는 함수는 '작업'이 되고 클래스는 '액터'가 됩니다. `@ray.remote` 주석을 통해 Ray에서 사용할 수 있는 함수와 클래스를 식별할 수 있습니다.
>
> 작업과 액터는 구성이 가능하고, 수명 주기가 있으며, 수명 주기 동안 컴퓨팅 리소스를 차지합니다. 문제의 근본 원인을 찾을 때 작업이나 액터 수준까지 오류를 발생시키는 코드를 추적할 수 있습니다. 따라서 이러한 용어는 AWS Glue for Ray 작업의 구성, 모니터링 또는 디버깅 방법을 배울 때 언급될 수 있습니다.
>
> 작업과 액터를 효과적으로 사용하여 분산 애플리케이션을 구축하는 방법을 배우려면 Ray 설명서의 Key Concepts를 참조하세요.

AWS Glue for Ray의 Ray Core

AWS Glue for Ray 환경은 클러스터 형성 및 규모 조정뿐만 아니라 로그 수집 및 시각화를 관리합니다. AWS에서도 이러한 항목을 관리하기 때문에 오픈 소스 클러스터에서 이러한 항목을 해결하는 데 사용되는 Ray Core의 API에 대한 액세스 및 지원이 제한됩니다.

관리형 `Ray2.4` 런타임 환경에서 다음을 지원하지 않습니다.

- Ray Core CLI
- Ray State CLI
- `ray.util.metrics` Prometheus 지표 유틸리티 메서드:
  - Counter
  - Gauge
  - Histogram
- 기타 디버깅 도구:
  - ray.util.pdb.set_trace
  - ray.util.inspect_serializability

- ray.timeline

Ray Data란 무엇인가요?

데이터 소스 및 대상에 연결하고, 데이터 세트를 처리하며, 일반적인 변환을 시작할 때 Ray Data는 Ray를 사용하여 Ray 데이터 세트 변환 관련 문제를 해결할 수 있는 간단한 방법입니다. Ray Data 사용에 대한 자세한 내용은 Ray Datasets: Distributed Data Preprocessing을 참조하세요.

Ray Data 또는 기타 도구를 사용하여 데이터에 액세스할 수 있습니다. Ray에서 데이터에 액세스하는 방법에 대한 자세한 내용은 the section called “데이터에 연결” 섹션을 참조하세요.

AWS Glue for Ray의 Ray Data

Ray Data는 관리형 `Ray2.4` 런타임 환경에서 기본적으로 지원되고 제공됩니다. 제공된 모듈에 대한 자세한 내용은 the section called “Ray 작업과 함께 제공되는 모듈” 섹션을 참조하세요.

Ray 작업에 파일 및 Python 라이브러리 제공

이 섹션에서는 AWS Glue Ray 작업에서 Python 라이브러리를 사용하는 데 필요한 정보를 제공합니다. 모든 Ray 작업에 기본적으로 포함된 특정 공용 라이브러리를 사용할 수 있습니다. Ray 작업에 자체 Python 라이브러리를 제공할 수도 있습니다.

Ray 작업과 함께 제공되는 모듈

다음과 같은 제공된 패키지를 사용하여 Ray 작업에서 데이터 통합 워크플로를 수행할 수 있습니다. 이러한 패키지는 Ray 작업에서 기본적으로 사용할 수 있습니다.

AWS Glue version 4.0

AWS Glue 4.0에서 Ray(`Ray2.4` 런타임) 환경은 다음 패키지를 제공합니다.

- boto3 == 1.26.133
- ray == 2.4.0
- pyarrow == 11.0.0
- pandas == 1.5.3
- numpy == 1.24.3
- fsspec == 2023.4.0

이 목록에는 `ray[data] == 2.4.0`과 함께 설치할 모든 패키지가 포함되어 있습니다. Ray Data는 기본적으로 지원됩니다.

Ray 작업에 파일 제공

`--working-dir` 파라미터를 사용하여 Ray 작업에 파일을 제공할 수 있습니다. Amazon S3에 호스팅된 .zip 파일의 경로를 이 파라미터에 제공합니다. .zip 파일 내에서 파일은 단일 최상위 디렉터리에 포함되어야 합니다. 최상위 수준에 다른 파일은 있을 수 없습니다.

스크립트를 실행하기 전에 파일이 각 Ray 노드에 배포됩니다. 이때 각 Ray 노드에서 사용할 수 있는 디스크 공간에 미치는 영향을 고려합니다. 사용 가능한 디스크 공간은 작업 구성에 설정된 WorkerType에 따라 결정됩니다. 작업 데이터를 대규모로 제공하려는 경우 이 메커니즘은 올바른 솔루션이 아닙니다. 작업에 데이터를 제공하는 방법에 대한 자세한 내용은 the section called “데이터에 연결” 섹션을 참조하세요.

`working_dir` 파라미터를 통해 Ray에 디렉터리를 제공한 것처럼 파일에도 액세스할 수 있습니다. 예를 들어 .zip 파일의 최상위 디렉터리에서 이름이 `sample.txt`인 파일을 읽으려면 다음을 직접 호출할 수 있습니다.

```
@ray.remote
def do_work():
    f = open("sample.txt", "r")
    print(f.read())
```

`working_dir`에 대한 자세한 내용은 Ray 설명서를 참조하세요. 이 기능은 Ray의 기본 기능과 유사하게 작동합니다.

Ray 작업을 위한 추가 Python 모듈

PyPI의 추가 모듈

Ray 작업에서는 Python 패키지 설치 프로그램(pip3)을 사용하여 Ray 스크립트에서 사용할 추가 모듈을 설치합니다. `--pip-install` 파라미터를 쉼표로 구분된 Python 모듈 목록과 함께 사용하여 새 모듈을 추가하거나 기존 모듈의 버전을 변경할 수 있습니다.

예를 들어 새 `scikit-learn` 모듈을 업데이트하거나 추가하려면 다음 키 값 페어를 사용합니다.

```
"--pip-install", "scikit-learn==0.21.3"
```

사용자 지정 모듈이나 사용자 지정 패치가 있는 경우, Amazon S3에서 `--s3-py-modules` 파라미터를 사용하여 자체 라이브러리를 배포할 수 있습니다. 배포를 업로드하기 전에 다시 패키징하고 다시 구축해야 할 수도 있습니다. the section called “Ray 작업에 Python 코드 포함”의 지침을 따릅니다.

Amazon S3의 사용자 지정 배포

사용자 지정 배포는 종속성에 대한 Ray 패키징 지침을 준수해야 합니다. 다음 섹션에서 이러한 배포를 빌드하는 방법을 확인할 수 있습니다. Ray에서 종속성을 설정하는 방법에 대한 자세한 내용은 Ray 설명서의 환경 종속성을 참조하세요.

콘텐츠를 평가한 후 사용자 지정 배포 파일을 포함하려면 작업의 IAM 역할에서 사용할 수 있는 버킷에 배포 가능 파일을 업로드합니다. 파라미터 구성에서 Python zip 아카이브에 대한 Amazon S3 경로를 지정합니다. 배포 파일을 여러 개 제공하는 경우 쉼표로 구분합니다. 예:

```
"--s3-py-modules", "s3://s3bucket/pythonPackage.zip"
```

제한 사항

Ray 작업에서는 작업 환경의 기본 코드 컴파일을 지원하지 않습니다. Python 종속성이 컴파일된 기본 코드에 전이적으로 종속되는 경우 이로 인해 제한될 수 있습니다. Ray 작업에서는 제공된 바이너리를 실행할 수 있지만 ARM64 기반 Linux용으로 컴파일해야 합니다. 즉, `aarch64manylinux` Wheel의 콘텐츠를 사용할 수 있습니다. Wheel을 Ray 표준에 따라 다시 패키징하여 컴파일된 형식으로 기본 종속성을 제공할 수 있습니다. 일반적으로 이는 아카이브의 루트에 단 하나의 폴더만 남도록 `dist-info` 폴더를 제거하는 것을 의미합니다.

이 파라미터를 사용하여 `ray` 또는 `ray[data]` 버전을 업그레이드할 수 없습니다. 새 버전의 Ray를 사용하려면 지원이 출시된 후 작업의 런타임 필드를 변경해야 합니다. 지원되는 Ray 버전에 대한 자세한 내용은 the section called “AWS Glue 버전” 섹션을 참조하세요.

Ray 작업에 Python 코드 포함

Python Software Foundation은 다양한 런타임에서 사용할 Python 파일을 패키징하기 위한 표준화된 동작을 제공합니다. Ray는 사용자가 숙지해야 하는 패키징 표준에 대한 제한 사항을 도입했습니다. AWS Glue에서는 Ray에 지정된 표준 이외의 패키징 표준을 지정하지 않습니다. 다음 지침은 간단한 Python 패키지 패키징 관련 표준 지침을 제공합니다.

파일을 `.zip` 아카이브에 패키징합니다. 디렉터리는 아카이브의 루트에 있어야 합니다. 아카이브의 루트 수준에 다른 파일이 없어야 합니다. 그렇지 않으면 예기치 않은 동작이 발생합니다. 루트 디렉터리는 패키지이며 패키지 이름은 패키지를 가져올 때 Python 코드를 참조하는 데 사용됩니다.

`--s3-py-modules`를 사용하여 Ray 작업에 이 양식의 배포를 제공하는 경우 Ray 스크립트의 패키지 파일에서 Python 코드를 가져올 수 있습니다.

패키지는 몇 가지 Python 파일을 포함하는 단일 Python 모듈을 제공하거나 여러 모듈을 함께 패키징할 수 있습니다. PyPI의 라이브러리와 같은 종속성을 다시 패키징할 때 해당 패키지 내에 숨겨진 파일 및 메타데이터 디렉터리가 있는지 확인합니다.

> **Warning**
>
> 특정 OS 동작으로 인해 이러한 패키징 지침을 제대로 수행하지 못할 수 있습니다.
>
> - OSX는 최상위 수준의 zip 파일에 숨겨진 파일(예: `__MACOSX`)을 추가할 수 있습니다.
> - Windows에서는 파일을 zip 내의 폴더에 자동으로 추가하여 의도치 않게 중첩된 폴더가 생성될 수 있습니다.

다음 절차에서는 Amazon Linux 2 또는 Info-ZIP `zip` 및 `zipinfo` 유틸리티 배포를 제공하는 유사한 OS에서 파일과 상호 작용하는 경우를 가정합니다. 예상치 못한 동작을 방지하려면 이러한 도구를 사용하는 것이 좋습니다.

Ray에서 사용할 Python 파일을 패키징하려면

1. 패키지 이름으로 임시 디렉터리를 생성하고 작업 디렉터리가 상위 디렉터리인지 확인합니다. 다음 명령으로 이를 수행할 수 있습니다.

   ```
   cd parent_directory
   mkdir temp_dir
   ```

2. 파일을 임시 디렉터리에 복사한 다음 디렉터리 구조를 확인합니다. 이 디렉터리의 콘텐츠는 Python 모듈로 직접 액세스됩니다. 이러한 권한 부여는 다음 명령을 사용하여 가능합니다.

   ```
   ls -AR temp_dir
   # my_file_1.py
   # my_file_2.py
   ```

3. zip을 사용하여 임시 폴더를 압축합니다. 다음 명령으로 이를 수행할 수 있습니다.

   ```
   zip -r zip_file.zip temp_dir
   ```

4. 파일이 제대로 패키징되었는지 확인합니다. 이제 작업 디렉터리에서 *zip_file*.zip을 찾을 수 있습니다. 다음 명령을 사용하여 검사할 수 있습니다.

```
zipinfo -1 zip_file.zip
# temp_dir/
# temp_dir/my_file_1.py
# temp_dir/my_file_2.py
```

Ray에서 사용할 Python 패키지를 다시 패키징하려면

1. 패키지 이름으로 임시 디렉터리를 생성하고 작업 디렉터리가 상위 디렉터리인지 확인합니다. 다음 명령으로 이를 수행할 수 있습니다.

```
cd parent_directory
mkdir temp_dir
```

2. 패키지의 압축을 풀고 콘텐츠를 임시 디렉터리에 복사합니다. 모듈의 콘텐츠만 남기고 이전 패키징 표준과 관련된 파일을 제거합니다. 다음 명령을 사용하여 파일 구조가 올바른지 확인합니다.

```
ls -AR temp_dir
# my_module
# my_module/__init__.py
# my_module/my_file_1.py
# my_module/my_submodule/__init__.py
# my_module/my_submodule/my_file_2.py
# my_module/my_submodule/my_file_3.py
```

3. zip을 사용하여 임시 폴더를 압축합니다. 다음 명령으로 이를 수행할 수 있습니다.

```
zip -r zip_file.zip temp_dir
```

4. 파일이 제대로 패키징되었는지 확인합니다. 이제 작업 디렉터리에서 *zip_file*.zip을 찾을 수 있습니다. 다음 명령을 사용하여 검사할 수 있습니다.

```
zipinfo -1 zip_file.zip
# temp_dir/my_module/
# temp_dir/my_module/__init__.py
# temp_dir/my_module/my_file_1.py
# temp_dir/my_module/my_submodule/
```

```
# temp_dir/my_module/my_submodule/__init__.py
# temp_dir/my_module/my_submodule/my_file_2.py
# temp_dir/my_module/my_submodule/my_file_3.py
```

Ray 작업의 데이터에 연결

AWS Glue Ray 작업에서는 데이터를 빠르게 통합하도록 설계된 다양한 Python 패키지를 사용할 수 있습니다. 사용자 환경의 혼란이 가중되지 않도록 최소한의 종속성 세트가 제공됩니다. 기본적으로 포함되는 항목에 대한 자세한 정보는 the section called “Ray 작업과 함께 제공되는 모듈” 섹션을 참조하세요.

> **Note**
>
> AWS Glue 추출, 전환, 적재(ETL)에서는 DynamicFrame 추상화를 제공하여 데이터 세트의 행 간 스키마 차이를 해결하는 ETL 워크플로를 간소화합니다. AWS Glue ETL은 추가 기능(예: 작업 북마크 및 입력 파일 그룹화)을 제공합니다. 현재 Ray 작업에서는 해당 기능을 제공하지 않습니다.
>
> AWS Glue for Spark는 특정 데이터 형식, 소스 및 싱크와의 연결을 직접 지원합니다. Ray에서 AWS SDK for pandas 및 최신 서드 파티 라이브러리가 실질적으로 이러한 요구 사항을 충족합니다. 사용 가능한 기능을 알아보려면 해당 라이브러리를 참조해야 합니다.

AWS Glue for Ray 및 Amazon VPC의 통합은 현재 사용할 수 없습니다. Amazon VPC 내 리소스는 퍼블릭 경로로만 액세스할 수 있습니다. Amazon VPC에서 AWS Glue를 사용하는 방법에 대한 자세한 내용은 the section called “AWS Glue에 대한 인터페이스 VPC 엔드포인트(AWS PrivateLink) 구성 ” 섹션을 참조하세요.

Ray의 데이터 작업을 위한 공통 라이브러리

Ray Data - Ray Data는 일반적인 데이터 형식, 소스 및 싱크를 처리하는 메서드를 제공합니다. Ray Data에서 지원되는 형식 및 소스에 대한 자세한 내용은 Ray Data 설명서의 Input/Output을 참조하세요. Ray Data는 데이터 세트를 처리하기 위한 범용 라이브러리가 아닌, 독자적인 라이브러리입니다.

Ray는 Ray Data가 작업에 가장 적합한 솔루션일 수 있는 사용 사례에 대한 특정 지침을 제공합니다. 자세한 내용은 Ray 설명서에서 Ray 사용 사례를 참조하세요.

AWS SDK for pandas (awswrangler) – AWS SDK for pandas는 변환을 통해 pandas DataFrames로 데이터를 관리할 때 AWS 서비스에서 읽고 쓸 수 있는 테스트된 깔끔한 솔루션을 제공하는 AWS 제품입

니다. AWS SDK for pandas에서 지원되는 형식 및 소스에 대한 자세한 내용은 AWS SDK for pandas 설명서의 API Reference를 참조하세요.

AWS SDK for pandas를 사용하여 데이터를 읽고 쓰는 방법에 대한 예제는 AWS SDK for pandas 설명서의 Quick Start를 참조하세요. AWS SDK for pandas에서는 데이터에 대한 변환을 제공하지 않습니다. 소스에서의 읽기 및 쓰기만 지원합니다.

Modin – Modin은 일반적인 pandas 연산을 배포 가능한 방식으로 구현하는 Python 라이브러리입니다. Modin에 대한 자세한 내용은 Modin 설명서를 참조하세요. Modin 자체는 소스에서의 읽기 및 쓰기를 지원하지 않습니다. 공통 변환의 분산 구현을 제공합니다. Modin은 AWS SDK for pandas에서 지원됩니다.

Ray 환경에서 Modin 및 AWS SDK for pandas를 함께 실행하면 공통 ETL 작업을 수행하여 더 뛰어난 결과를 얻을 수 있습니다. AWS SDK for pandas와 함께 Modin을 사용하는 방법에 대한 자세한 내용은 AWS SDK for pandas 설명서의 At scale을 참조하세요.

기타 프레임워크 - Ray가 지원하는 프레임워크에 대한 자세한 내용은 Ray 설명서의 Ray 에코시스템을 참조하세요. AWS Glue for Ray에서는 다른 프레임워크에 대한 지원을 제공하지 않습니다.

데이터 카탈로그를 통해 데이터에 연결

Ray 작업과 함께 데이터 카탈로그를 통해 데이터를 관리하는 기능은 AWS SDK for pandas에서 지원됩니다. 자세한 내용은 AWS SDK for pandas 웹사이트의 Glue 카탈로그를 참조하세요.

AWS SDK와 함께 이 서비스 사용

다양한 프로그래밍 언어에 대해 AWS 소프트웨어 개발 키트(SDK)을 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예시
AWS SDK for C++	AWS SDK for C++ 코드 예시
AWS CLI	AWS CLI 코드 예시
AWS SDK for Go	AWS SDK for Go 코드 예시
AWS SDK for Java	AWS SDK for Java 코드 예시
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예시
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예시
AWS SDK for .NET	AWS SDK for .NET 코드 예시
AWS SDK for PHP	AWS SDK for PHP 코드 예시
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예시
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예시
AWS SDK for Ruby	AWS SDK for Ruby 코드 예시
AWS SDK for Rust	AWS SDK for Rust 코드 예시
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예시
AWS SDK for Swift	AWS SDK for Swift 코드 예시

이 서비스 관련 예시는 AWS SDK를 사용한 AWS Glue API 코드 예제를 참조하세요.

예제 사용 가능 여부

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

AWS Glue API

이 섹션에서는 AWS Glue SDK 및 도구에서 사용하는 데이터 유형 및 기본 요소에 대해 설명합니다. 각각 자체 문서가 있는 AWS Management Console 외부에서 프로그래밍 방식으로 AWS Glue와 상호 작용하는 세 가지 일반적인 방법이 있습니다.

- 언어 SDK 라이브러리를 사용하면 일반적인 프로그래밍 언어의 AWS 리소스에 액세스할 수 있습니다. AWS를 기반으로 빌드할 도구에서 자세한 정보를 찾으십시오.
- AWS CLI를 사용하면 명령줄에서 AWS 리소스에 액세스할 수 있습니다. AWS CLI명령 참조에서 자세한 정보를 찾으십시오.
- AWS CloudFormation을 사용하면 일관되게 함께 프로비저닝할 AWS 리소스 집합을 정의할 수 있습니다. AWS CloudFormation:AWS Glue리소스 유형 참조에서 자세한 정보를 찾으십시오.

이 섹션에서는 이러한 SDK 및 도구와는 별도로 공유 기본 요소에 대해 설명합니다. 도구는 AWS Glue 웹 API 참조를 사용하여 AWS와 통신합니다.

목차

- AWS Glue의 보안 API
  - 데이터 타입
  - DataCatalogEncryptionSettings 구조
  - EncryptionAtRest 구조
  - ConnectionPasswordEncryption 구조
  - EncryptionConfiguration 구조
  - S3Encryption 구조
  - CloudWatchEncryption 구조
  - JobBookmarksEncryption 구조
  - SecurityConfiguration 구조
  - GluePolicy 구조
  - DataQualityEncryption 구조
  - 운영
  - GetDataCatalogEncryptionSettings 작업(Python: get_data_catalog_encryption_settings)
  - PutDataCatalogEncryptionSettings 작업(Python: put_data_catalog_encryption_settings)
  - PutResourcePolicy 작업(Python: put_resource_policy)

- GetResourcePolicy 작업(Python: get_resource_policy)
- DeleteResourcePolicy 작업(Python: delete_resource_policy)
- CreateSecurityConfiguration 작업(Python: create_security_configuration)
- DeleteSecurityConfiguration 작업(Python: delete_security_configuration)
- GetSecurityConfiguration 작업(Python: get_security_configuration)
- GetSecurityConfigurations 작업(Python: get_security_configurations)
- GetResourcePolicies 작업(Python: get_resource_policies)
- 카탈로그 객체 API
- 카탈로그 API
    - 데이터 타입
    - 카탈로그 구조
    - CatalogInput 구조
    - TargetRedshiftCatalog 구조
    - CatalogProperties 구조
    - CatalogPropertiesOutput 구조
    - DataLakeAccessProperties 구조
    - DataLakeAccessPropertiesOutput 구조
    - FederatedCatalog 구조
    - 운영
    - CreateCatalog 작업(Python: create_catalog)
    - UpdateCatalog 작업(Python: update_catalog)
    - DeleteCatalog 작업(Python: delete_catalog)
    - GetCatalog 작업(Python: get_catalog)
    - GetCatalogs 작업(Python: get_catalogs)
- 데이터베이스 API
    - 데이터 타입
    - 데이터베이스 구조
    - DatabaseInput 구조
    - PrincipalPermissions 구조
    - DataLakePrincipal 구조

- DatabaseIdentifier 구조
- FederatedDatabase 구조
- 운영
- CreateDatabase 작업(Python: create_database)
- UpdateDatabase 작업(Python: update_database)
- DeleteDatabase 작업(Python: delete_database)
- GetDatabase 작업(Python: get_database)
- GetDatabases 작업(Python: get_database)

- 표 API
  - 데이터 타입
  - 테이블 구조
  - TableInput 구조
  - FederatedTable 구조
  - 열 구조
  - StorageDescriptor 구조
  - SchemaReference 구조
  - SerDeInfo 구조
  - Order 구조
  - SkewedInfo 구조
  - TableVersion 구조
  - TableError 구조
  - TableVersionError 구조
  - SortCriterion 구조
  - TableIdentifier 구조
  - KeySchemaElement 구조
  - PartitionIndex 구조
  - PartitionIndexDescriptor 구조
  - BackfillError 구조
  - IcebergInput 구조
  - OpenTableFormatInput 구조

- ViewDefinition 구조
- ViewDefinitionInput 구조
- ViewRepresentation 구조
- ViewRepresentationInput 구조
- UpdateOpenTableFormatInput 구조
- UpdateIcebergInput 구조
- CreateIcebergTableInput 구조
- UpdateIcebergTableInput 구조
- IcebergSortOrder 구조
- IcebergSortField 구조
- IcebergPartitionSpec 구조
- IcebergPartitionField 구조
- IcebergSchema 구조
- IcebergStructField 구조
- IcebergTableUpdate 구조
- 운영
- CreateTable 작업(Python: create_table)
- UpdateTable 작업(Python: update_table)
- DeleteTable 작업(Python: delete_table)
- BatchDeleteTable 작업(Python: batch_delete_table)
- GetTable 작업(Python: get_table)
- GetTables 작업(Python: get_tables)
- GetTableVersion 작업(Python: get_table_version)
- GetTableVersions 작업(Python: get_table_versions)
- DeleteTableVersion 작업(Python: delete_table_version)
- BatchDeleteTableVersion 작업(Python: batch_delete_table_version)
- SearchTables 작업(Python: search_tables)
- GetPartitionIndexes 작업(Python: get_partition_indexes)
- CreatePartitionIndex 작업(Python: create_partition_index)
- DeletePartitionIndex 작업(Python: delete_partition_index)

- GetColumnStatisticsForTable 작업(Python: get_column_statistics_for_table)
- UpdateColumnStatisticsForTable 작업(Python: update_column_statistics_for_table)
- DeleteColumnStatisticsForTable 작업(Python: delete_column_statistics_for_table)
- 파티션 API
  - 데이터 타입
  - 파티션 구조
  - PartitionInput 구조
  - PartitionSpecWithSharedStorageDescriptor 구조
  - PartitionListComposingSpec 구조
  - PartitionSpecProxy 구조
  - PartitionValueList 구조
  - 세그먼트 구조
  - PartitionError 구조
  - BatchUpdatePartitionFailureEntry 구조
  - BatchUpdatePartitionRequestEntry 구조
  - StorageDescriptor 구조
  - SchemaReference 구조
  - SerDeInfo 구조
  - SkewedInfo 구조
  - 운영
  - CreatePartition 작업(Python: create_partition)
  - BatchCreatePartition 작업(Python: batch_create_partition)
  - UpdatePartition 작업(Python: update_partition)
  - DeletePartition 작업(Python: delete_partition)
  - BatchDeletePartition 작업(Python: batch_delete_partition)
  - GetPartition 작업(Python: get_partition)
  - GetPartitions 작업(Python: get_partitions)
  - BatchGetPartition 작업(Python: batch_get_partition)
  - BatchUpdatePartition 작업(Python: batch_update_partition)
  - GetColumnStatisticsForPartition 작업(Python: get_column_statistics_for_partition)

- UpdateColumnStatisticsForPartition 작업(Python: update_column_statistics_for_partition)
- DeleteColumnStatisticsForPartition 작업(Python: delete_column_statistics_for_partition)

- 연결 API
  - 연결 API
    - 데이터 타입
    - 연결 구조
    - ConnectionInput 구조
    - TestConnectionInput 구조
    - PhysicalConnectionRequirements 구조
    - GetConnectionsFilter 구조
    - AuthenticationConfiguration 구조
    - AuthenticationConfigurationInput 구조
    - OAuth2Properties 구조
    - OAuth2PropertiesInput 구조
    - OAuth2ClientApplication 구조
    - AuthorizationCodeProperties 구조
    - BasicAuthenticationCredentials 구조
    - OAuth2Credentials 구조
    - 운영
    - CreateConnection 작업(Python: create_connection)
    - DeleteConnection 작업(Python: delete_connection)
    - GetConnection 작업(Python: get_connection)
    - GetConnections 작업(Python: get_connections)
    - UpdateConnection 작업(Python: update_connection)
    - TestConnection 작업(Python: test_connection)
    - BatchDeleteConnection 작업(Python: batch_delete_connection)
  - 연결 유형 API
    - 연결 관리 API
    - DescribeConnectionType 작업(Python: describe_connection_type)
    - ListConnectionTypes 작업(Python: list_connection_types)

        - ConnectionTypeBrief 구조
        - ConnectionTypeVariant 구조
        - 데이터 유형
        - Validation 구조
        - AuthConfiguration 구조
        - Capabilities 구조
        - Property 구조
        - AllowedValue 구조
        - ComputeEnvironmentConfiguration 구조
    - 연결 메타데이터 및 미리 보기 API
        - 데이터 타입
        - Entity 구조
        - Field 구조
        - 운영
        - ListEntities 작업(Python: list_entities)
        - DescribeEntity 작업(Python: describe_entity)
        - GetEntityRecords 작업(Python: get_entity_records)
- 사용자 정의 함수 API
    - 데이터 타입
    - UserDefinedFunction 구조
    - UserDefinedFunctionInput 구조
    - 운영
    - CreateUserDefinedFunction 작업(Python: create_user_defined_function)
    - UpdateUserDefinedFunction 작업 (Python: update_user_defined_function)
    - DeleteUserDefinedFunction 작업 (Python: delete_user_defined_function)
    - GetUserDefinedFunction 작업 (Python: get_user_defined_function)
    - GetUserDefinedFunctions 작업 (Python: get_user_defined_functions)
- Athena 카탈로그를 AWS Glue로 가져오기
    - 데이터 타입
    - CatalogImportStatus 구조

    - 운영
    - ImportCatalogToGlue 작업(Python: import_catalog_to_glue)
    - GetCatalogImportStatus 작업(Python: get_catalog_import_status)
- 테이블 옵티마이저 API
    - 데이터 타입
    - TableOptimizer 구조
    - TableOptimizerConfiguration 구조
    - TableOptimizerVpcConfiguration 구조
    - CompactionConfiguration 구조
    - IcebergCompactionConfiguration 구조
    - TableOptimizerRun 구조
    - BatchGetTableOptimizerEntry 구조
    - BatchTableOptimizer 구조
    - BatchGetTableOptimizerError 구조
    - RetentionConfiguration 구조
    - IcebergRetentionConfiguration 구조
    - OrphanFileDeletionConfiguration 구조
    - IcebergOrphanFileDeletionConfiguration 구조
    - CompactionMetrics 구조
    - RetentionMetrics 구조
    - OrphanFileDeletionMetrics 구조
    - IcebergCompactionMetrics 구조
    - IcebergRetentionMetrics 구조
    - IcebergOrphanFileDeletionMetrics 구조
    - RunMetrics 구조
    - 운영
    - GetTableOptimizer 작업 (Python: get_table_optimizer)
    - BatchGetTableOptimizer 작업 (Python: batch_get_table_optimizer)
    - ListTableOptimizerRuns 작업 (Python: list_table_optimizer_runs)
    - CreateTableOptimizer 작업 (Python: create_table_optimizer)

- DeleteTableOptimizer 작업 (Python: delete_table_optimizer)
- UpdateTableOptimizer 작업 (Python: update_table_optimizer)
- 크롤러 및 분류자 API
  - 분류자 API
    - 데이터 타입
    - 분류자 구조
    - GrokClassifier 구조
    - XMLClassifier 구조
    - JsonClassifier 구조
    - CsvClassifier 구조
    - CreateGrokClassifierRequest 구조
    - UpdateGrokClassifierRequest 구조
    - CreateXMLClassifierRequest 구조
    - UpdateXMLClassifierRequest 구조
    - CreateJsonClassifierRequest 구조
    - UpdateJsonClassifierRequest 구조
    - CreateCsvClassifierRequest 구조
    - UpdateCsvClassifierRequest 구조
    - 운영
    - CreateClassifier 작업(Python: create_classifier)
    - DeleteClassifier 작업(Python: delete_classifier)
    - GetClassifier 작업(Python: get_classifier)
    - GetClassifiers 작업(Python: get_classifiers)
    - UpdateClassifier 작업(Python: update_classifier)
  - 크롤러 API
    - 데이터 타입
    - 크롤러 구조
    - 일정 구조
    - CrawlerTargets 구조
    - S3Target 구조

- S3DeltaCatalogTarget 구조
- S3DeltaDirectTarget 구조
- JdbcTarget 구조
- MongoDBTarget 구조
- DynamoDBTarget 구조
- DeltaTarget 구조
- IcebergTarget 구조
- HudiTarget 구조
- CatalogTarget 구조
- CrawlerMetrics 구조
- CrawlerHistory 구조
- CrawlsFilter 구조
- SchemaChangePolicy 구조
- LastCrawlInfo 구조
- RecrawlPolicy 구조
- LineageConfiguration 구조
- LakeFormationConfiguration 구조
- 운영
- CreateCrawler 작업(Python: create_crawler)
- DeleteCrawler 작업(Python: delete_crawler)
- GetCrawler 작업(Python: get_crawler)
- GetCrawlers 작업(Python: get_crawler)
- GetCrawlerMetrics Action(Python: get_crawler_metrics)
- UpdateCrawler 작업(Python: update_crawler)
- StartCrawler 작업(Python: start_crawler)
- StopCrawler 작업(Python: stop_crawler)
- BatchGetCrawlers 작업(Python: batch_get_crawlers)
- ListCrawlers 작업(Python: list_crawlers)
- ListCrawls 작업(Python: list_crawls)

- 열 통계의 API

- 데이터 타입
- ColumnStatisticsTaskRun 구조
- ColumnStatisticsTaskSettings 구조
- ExecutionAttempt 구조
- 운영
- StartColumnStatisticsTaskRun 작업 (Python: start_column_statistics_task_run)
- GetColumnStatisticsTaskRun 작업 (Python: get_column_statistics_task_run)
- GetColumnStatisticsTaskRuns 작업 (Python: get_column_statistics_task_runs)
- ListColumnStatisticsTaskRuns 작업 (Python: list_column_statistics_task_runs)
- StopColumnStatisticsTaskRun 작업 (Python: stop_column_statistics_task_run)
- CreateColumnStatisticsTaskSettings 작업(Python: create_column_statistics_task_settings)
- UpdateColumnStatisticsTaskSettings 작업(Python: update_column_statistics_task_settings)
- GetColumnStatisticsTaskSettings 작업(Python: get_column_statistics_task_settings)
- DeleteColumnStatisticsTaskSettings 작업(Python: delete_column_statistics_task_settings)
- StartColumnStatisticsTaskRunSchedule 작업(Python: start_column_statistics_task_run_schedule)
- StopColumnStatisticsTaskRunSchedule 작업(Python: stop_column_statistics_task_run_schedule)
- 예외
- ColumnStatisticsTaskRunningException 구조
- ColumnStatisticsTaskNotRunningException 구조
- ColumnStatisticsTaskStoppingException 구조
- ColumnStatisticsTaskAutoConcurrencyLimitException 구조
- InvalidCatalogSettingException 구조

- 크롤러 스케줄러 API
  - 데이터 타입
  - 일정 구조
  - 운영
  - UpdateCrawlerSchedule 작업(Python: start_crawler_schedule)
  - StartCrawlerSchedule 작업(Python: start_crawler_schedule)
  - StopCrawlerSchedule 작업(Python: stop_crawler_schedule)

- ETL 스크립트 API 자동 생성
  - 데이터 타입
  - CodeGenNode 구조
  - CodeGenNodeArg 구조
  - CodeGenEdge 구조
  - 위치 구조
  - CatalogEntry 구조
  - MappingEntry 구조
  - 운영
  - CreateScript 작업(Python: create_script)
  - GetDataflowGraph 작업(Python: get_dataflow_graph)
  - GetMapping 작업(Python: get_mapping)
  - GetPlan 작업(Python: get_plan)
- 시각적 작업 API
  - 데이터 타입
  - CodeGenConfigurationNode 구조
  - JDBCConnectorOptions 구조
  - StreamingDataPreviewOptions 구조
  - AthenaConnectorSource 구조
  - JDBCConnectorSource 구조
  - SparkConnectorSource 구조
  - CatalogSource 구조
  - MySQLCatalogSource 구조
  - PostgreSQLCatalogSource 구조
  - OracleSQLCatalogSource 구조
  - MicrosoftSQLServerCatalogSource 구조
  - CatalogKinesisSource 구조
  - DirectKinesisSource 구조
  - KinesisStreamingSourceOptions 구조
  - CatalogKafkaSource 구조

- DirectKafkaSource 구조
- KafkaStreamingSourceOptions 구조
- RedshiftSource 구조
- AmazonRedshiftSource 구조
- AmazonRedshiftNodeData 구조
- AmazonRedshiftAdvancedOption 구조
- 옵션 구조
- S3CatalogSource 구조
- S3SourceAdditionalOptions 구조
- S3CsvSource 구조
- DirectJDBCSource 구조
- S3DirectSourceAdditionalOptions 구조
- S3JsonSource 구조
- S3ParquetSource 구조
- S3DeltaSource 구조
- S3CatalogDeltaSource 구조
- CatalogDeltaSource 구조
- S3HudiSource 구조
- S3CatalogHudiSource 구조
- S3ExcelSource 구조
- CatalogHudiSource 구조
- DynamoDBCatalogSource 구조
- RelationalCatalogSource 구조
- JDBCConnectorTarget 구조
- SparkConnectorTarget 구조
- BasicCatalogTarget 구조
- MySQLCatalogTarget 구조
- PostgreSQLCatalogTarget 구조
- OracleSQLCatalogTarget 구조
- MicrosoftSQLServerCatalogTarget 구조

- RedshiftTarget 구조
- AmazonRedshiftTarget 구조
- UpsertRedshiftTargetOptions 구조
- S3CatalogTarget 구조
- S3GlueParquetTarget 구조
- CatalogSchemaChangePolicy 구조
- S3DirectTarget 구조
- S3HudiCatalogTarget 구조
- S3HudiDirectTarget 구조
- S3DeltaCatalogTarget 구조
- S3DeltaDirectTarget 구조
- S3HyperDirectTarget 구조
- S3IcebergDirectTarget 구조
- DirectSchemaChangePolicy 구조
- ApplyMapping 구조
- Mapping 구조
- SelectFields 구조
- DropFields 구조
- RenameField 구조
- Spigot 구조
- 조인 구조
- JoinColumn 구조
- SplitFields 구조
- SelectFromCollection 구조
- FillMissingValues 구조
- Filter 구조
- FilterExpression 구조
- FilterValue 구조
- CustomCode 구조
- SparkSQL 구조

- SqlAlias 구조
- DropNullFields 구조
- NullCheckBoxList 구조
- NullValueField 구조
- 데이터 형식 구조
- 병합 구조
- 결합 구조
- PIIDetection 구조
- 집계 구조
- DropDuplicates 구조
- GovernedCatalogTarget 구조
- GovernedCatalogSource 구조
- AggregateOperation 구조
- GlueSchema 구조
- GlueStudioSchemaColumn 구조
- GlueStudioColumn 구조
- DynamicTransform 구조
- TransformConfigParameter 구조
- EvaluateDataQuality 구조
- DQResultsPublishingOptions 구조
- DQStopJobOnFailureOptions 구조
- EvaluateDataQualityMultiFrame 구조
- 레시피 구조
- RecipeReference 구조
- SnowflakeNodeData 구조
- SnowflakeSource 구조
- SnowflakeTarget 구조
- ConnectorDataSource 구조
- ConnectorDataTarget 구조
- RecipeStep 구조

- RecipeAction 구조
- ConditionExpression 구조
- 작업 API
  - 작업
    - 데이터 타입
    - 작업 구조
    - ExecutionProperty 구조
    - NotificationProperty 구조
    - JobCommand 구조
    - ConnectionsList 구조
    - JobUpdate 구조
    - SourceControlDetails 구조
    - 운영
    - CreateJob 작업(Python: create_job)
    - UpdateJob 작업(Python: update_job)
    - GetJob 작업(Python: get_job)
    - GetJobs 작업(Python: get_jobs)
    - DeleteJob 작업(Python: delete_job)
    - ListJobs 작업(Python: list_jobs)
    - BatchGetJobs 작업(Python: batch_get_jobs)
  - 작업 실행
    - 데이터 타입
    - JobRun 구조
    - 이전 구조
    - JobBookmarkEntry 구조
    - BatchStopJobRunSuccessfulSubmission 구조
    - BatchStopJobRunError 구조
    - NotificationProperty 구조
    - 운영
    - StartJobRun 작업(Python: start_job_run)

    - BatchStopJobRun 작업(Python: batch_stop_job_run)
    - GetJobRun 작업(Python: get_job_run)
    - GetJobRuns 작업(Python: get_job_runs)
    - GetJobBookmark 작업(Python: get_job_bookmark)
    - GetJobBookmarks 작업(Python: get_job_bookmarks)
    - ResetJobBookmark 작업(Python: reset_job_bookmark)
  - 트리거
    - 데이터 타입
    - 트리거 구조
    - TriggerUpdate 구조
    - 조건자 구조
    - 조건 구조
    - 작업 구조
    - EventBatchingCondition 구조
    - 운영
    - CreateTrigger 작업(Python: create_trigger)
    - StartTrigger 작업(Python: start_trigger)
    - GetTrigger 작업(Python: get_trigger)
    - GetTrigger 작업(Python: get_triggers)
    - UpdateTrigger 작업(Python: update_trigger)
    - StopTrigger 작업(Python: stop_trigger)
    - DeleteTrigger 작업(Python: delete_trigger)
    - ListTriggers 작업(Python: list_triggers)
    - BatchGetTriggers 작업(Python: batch_get_triggers)
- AWS Glue의 통합 API
  - 데이터 타입
  - 통합 구조
  - IntegrationConfig 구조
  - IntegrationPartition 구조
  - IntegrationError 구조

- IntegrationFilter 구조
- InboundIntegration 구조
- SourceProcessingProperties 구조
- TargetProcessingProperties 구조
- SourceTableConfig 구조
- TargetTableConfig 구조
- 운영
- CreateIntegration 작업(Python: create_integration)
- ModifyIntegration 작업(Python: modify_integration)
- DescribeIntegrations 작업(Python: describe_integrations)
- DeleteIntegration 작업(Python: delete_integration)
- DescribeInboundIntegrations 작업(Python: describe_inbound_integrations)
- CreateIntegrationTableProperties 작업(Python: create_integration_table_properties)
- UpdateIntegrationTableProperties 작업(Python: update_integration_table_properties)
- GetIntegrationTableProperties 작업(Python: get_integration_table_properties)
- DeleteIntegrationTableProperties 작업(Python: delete_integration_table_properties)
- CreateIntegrationResourceProperty 작업(Python: create_integration_resource_property)
- UpdateIntegrationResourceProperty 작업(Python: update_integration_resource_property)
- GetIntegrationResourceProperty 작업(Python: get_integration_resource_property)
- UntagResource 작업(Python: untag_resource)
- ListTagsForResource 작업(Python: list_tags_for_resource)
- 예외
- ResourceNotFoundException 구조
- InternalServerException 구조
- IntegrationAlreadyExistsFault 구조
- IntegrationConflictOperationFault 구조
- IntegrationQuotaExceededFault 구조
- KMSKeyNotAccessibleFault 구조
- IntegrationNotFoundFault 구조
- TargetResourceNotFound 구조

- InvalidIntegrationStateFault 구조
- 대화형 세션 API
  - 데이터 타입
  - 세션 구조
  - SessionCommand 구조
  - 명령문 구조
  - StatementOutput 구조
  - StatementOutputData 구조
  - ConnectionsList 구조
  - 운영
  - CreateSession 작업(Python: create_session)
  - StopSession 작업(Python: stop_session)
  - DeleteSession 작업(Python: delete_session)
  - GetSession 작업(Python: get_session)
  - ListSessions 작업(Python: list_sessions)
  - RunStatement 작업(Python: run_statement)
  - CancelStatement 작업(Python: cancel_statement)
  - GetStatement 작업(Python: get_statement)
  - ListStatements 작업(Python: list_statements)
- 개발 엔드포인트 API
  - 데이터 타입
  - DevEndpoint 구조
  - DevEndpointCustomLibraries 구조
  - 운영
  - CreateDevEndpoint 작업(Python: create_dev_endpoint)
  - UpdateDevEndpoint 작업(Python: update_dev_endpoint)
  - DeleteDevEndpoint 작업(Python: delete_dev_endpoint)
  - GetDevEndpoint 작업(Python: get_dev_endpoint)
  - GetDevEndpoints 작업(Python: get_dev_endpoints)
  - BatchGetDevEndpoints 작업(Python: batch_get_dev_endpoints)

- ListDevEndpoints 작업(Python: list_dev_endpoints)
- Schema Registry
  - 데이터 타입
  - RegistryId 구조
  - RegistryListItem 구조
  - MetadataInfo 구조
  - OtherMetadataValueListItem 구조
  - SchemaListItem 구조
  - SchemaVersionListItem 구조
  - MetadataKeyValuePair 구조
  - SchemaVersionErrorItem 구조
  - ErrorDetails 구조
  - SchemaVersionNumber 구조
  - SchemaId 구조
  - 운영
  - CreateRegistry 작업(Python: create_registry)
  - CreateSchema 작업(Python: create_schema)
  - GetSchema 작업(Python: get_schema)
  - ListSchemaVersions 작업(Python: list_schema_versions)
  - GetSchemaVersion 작업(Python: get_schema_version)
  - GetSchemaVersionsDiff 작업(Python: get_schema_versions_diff)
  - ListRegistries 작업(Python: list_registries)
  - ListSchemas 작업(Python: list_schemas)
  - RegisterSchemaVersion 작업(Python: register_schema_version)
  - UpdateSchema 작업(Python: update_schema)
  - CheckSchemaVersionValidity 작업(Python: check_schema_version_validity)
  - UpdateRegistry 작업(Python: update_registry)
  - GetSchemaByDefinition 작업(Python: get_schema_by_definition)
  - GetRegistry 작업(Python: get_registry)
  - PutSchemaVersionMetadata 작업(Python: put_schema_version_metadata)

- QuerySchemaVersionMetadata 작업(Python: query_schema_version_metadata)
- RemoveSchemaVersionMetadata 작업(Python: remove_schema_version_metadata)
- DeleteRegistry 작업(Python: delete_registry)
- DeleteSchema 작업(Python: delete_schema)
- DeleteSchemaVersions 작업(Python: delete_schema_versions)
- 워크플로
  - 데이터 타입
  - JobNodeDetails 구조
  - CrawlerNodeDetails 구조
  - TriggerNodeDetails 구조
  - 크롤 구조
  - 노드 구조
  - 엣지 구조
  - 워크플루 구조
  - WorkflowGraph 구조
  - WorkflowRun 구조
  - WorkflowRunStatistics 구조
  - StartingEventBatchCondition 구조
  - 블루프린트 구조
  - BlueprintDetails 구조
  - LastActiveDefinition 구조
  - BlueprintRun 구조
  - 운영
  - CreateWorkflow 작업(Python: create_workflow)
  - UpdateWorkflow 작업(Python: update_workflow)
  - DeleteWorkflow 작업(Python: delete_workflow)
  - GetWorkflow 작업(Python: get_workflow)
  - ListWorkflows 작업(Python: list_workflows)
  - BatchGetWorkflows 작업(Python: batch_get_workflows)
  - GetWorkflowRun 작업(Python: get_workflow_run)

- GetWorkflowRuns 작업(Python: get_workflow_runs)
- GetWorkflowRunProperties 작업(Python: get_workflow_run_properties)
- PutWorkflowRunProperties 작업(Python: put_workflow_run_properties)
- CreateBlueprint 작업(Python: create_blueprint)
- UpdateBlueprint 작업(Python: update_blueprint)
- DeleteBlueprint 작업(Python: delete_blueprint)
- ListBlueprints 작업(Python: list_blueprints)
- BatchGetBlueprints 작업(Python: batch_get_blueprints)
- StartBlueprintRun 작업(Python: start_blueprint_run)
- GetBlueprintRun 작업(Python: get_blueprint_run)
- GetBlueprintRuns 작업(Python: get_blueprint_runs)
- StartWorkflowRun 작업(Python: start_workflow_run)
- StopWorkflowRun 작업(Python: stop_workflow_run)
- ResumeWorkflowRun 작업(Python: resume_workflow_run)
- 사용 프로필
  - 데이터 타입
  - ProfileConfiguration 구조
  - ConfigurationObject 구조
  - UsageProfileDefinition 구조
  - 운영
  - CreateUsageProfile 작업(Python: create_usage_profile)
  - GetUsageProfile 작업(Python: get_usage_profile)
  - UpdateUsageProfile 작업(Python: update_usage_profile)
  - DeleteUsageProfile 작업(Python: delete_usage_profile)
  - ListUsageProfiles 작업(Python: list_usage_profiles)
- 기계 학습 API
  - 데이터 타입
  - TransformParameters 구조
  - EvaluationMetrics 구조
  - MLTransform 구조

- FindMatchesParameters 구조
- FindMatchesMetrics 구조
- ConfusionMatrix 구조
- GlueTable 구조
- TaskRun 구조
- TransformFilterCriteria 구조
- TransformSortCriteria 구조
- TaskRunFilterCriteria 구조
- TaskRunSortCriteria 구조
- TaskRunProperties 구조
- FindMatchesTaskRunProperties 구조
- ImportLabelsTaskRunProperties 구조
- ExportLabelsTaskRunProperties 구조
- LabelingSetGenerationTaskRunProperties 구조
- SchemaColumn 구조
- TransformEncryption 구조
- MLUserDataEncryption 구조
- ColumnImportance 구조
- 운영
- CreateMLTransform 작업(Python: create_ml_transform)
- UpdateMLTransform 작업(Python: update_ml_transform)
- DeleteMLTransform 작업(Python: delete_ml_transform)
- GetMLTransform 작업(Python: get_ml_transform)
- GetMLTransforms 작업(Python: get_ml_transforms)
- ListMLTransforms 작업(Python: list_ml_transforms)
- StartMLEvaluationTaskRun 작업(Python: start_ml_evaluation_task_run)
- StartMLLabelingSetGenerationTaskRun 작업(Python: start_ml_labeling_set_generation_task_run)
- GetMLTaskRun 작업(Python: get_ml_task_run)
- GetMLTaskRuns 작업(Python: get_ml_task_runs)

- CancelMLTaskRun 작업(Python: cancel_ml_task_run)
- StartExportLabelsTaskRun 작업(Python: start_export_labels_task_run)
- StartImportLabelsTaskRun 작업(Python: start_import_labels_task_run)

- 데이터 품질 API
  - 데이터 타입
  - DataSource 구조
  - DataQualityRulesetListDetails 구조
  - DataQualityTargetTable 구조
  - DataQualityRulesetEvaluationRunDescription 구조
  - DataQualityRulesetEvaluationRunFilter 구조
  - DataQualityEvaluationRunAdditionalRunOptions 구조
  - DataQualityRuleRecommendationRunDescription 구조
  - DataQualityRuleRecommendationRunFilter 구조
  - DataQualityResult 구조
  - DataQualityAnalyzerResult 구조
  - DataQualityObservation 구조
  - MetricBasedObservation 구조
  - DataQualityMetricValues 구조
  - DataQualityRuleResult 구조
  - DataQualityResultDescription 구조
  - DataQualityResultFilterCriteria 구조
  - DataQualityRulesetFilterCriteria 구조
  - DataQualityAggregatedMetrics 구조
  - StatisticAnnotation 구조
  - TimestampedInclusionAnnotation 구조
  - AnnotationError 구조
  - DatapointInclusionAnnotation 구조
  - StatisticSummaryList 목록
  - StatisticSummary 구조
  - RunIdentifier 구조

- StatisticModelResult 구조
- 운영
- StartDataQualityRulesetEvaluationRun 작업(Python: start_data_quality_ruleset_evaluation_run)
- CancelDataQualityRulesetEvaluationRun 작업(Python: cancel_data_quality_ruleset_evaluation_run)
- GetDataQualityRulesetEvaluationRun 작업(Python: get_data_quality_ruleset_evaluation_run)
- ListDataQualityRulesetEvaluationRuns 작업(Python: list_data_quality_ruleset_evaluation_runs)
- StartDataQualityRuleRecommendationRun 작업(Python: start_data_quality_rule_recommendation_run)
- CancelDataQualityRuleRecommendationRun 작업(Python: cancel_data_quality_rule_recommendation_run)
- GetDataQualityRuleRecommendationRun 작업(Python: get_data_quality_rule_recommendation_run)
- ListDataQualityRuleRecommendationRuns 작업(Python: list_data_quality_rule_recommendation_runs)
- GetDataQualityResult 작업(Python: get_data_quality_result)
- BatchGetDataQualityResult 작업(Python: batch_get_data_quality_result)
- ListDataQualityResults 작업(Python: list_data_quality_results)
- CreateDataQualityRuleset 작업(Python: create_data_quality_ruleset)
- DeleteDataQualityRuleset 작업(Python: delete_data_quality_ruleset)
- GetDataQualityRuleset 작업(Python: get_data_quality_ruleset)
- ListDataQualityRulesets 작업(Python: list_data_quality_rulesets)
- UpdateDataQualityRuleset 작업(Python: update_data_quality_ruleset)
- ListDataQualityStatistics 작업(Python: list_data_quality_statistics)
- TimestampFilter 구조
- CreateDataQualityRulesetRequest 구조
- GetDataQualityRulesetResponse 구조
- GetDataQualityResultResponse 구조
- StartDataQualityRuleRecommendationRunRequest 구조
- GetDataQualityRuleRecommendationRunResponse 구조

- BatchPutDataQualityStatisticAnnotation 작업(Python: batch_put_data_quality_statistic_annotation)
- GetDataQualityModel 작업(Python: get_data_quality_model)
- GetDataQualityModelResult 작업(Python: get_data_quality_model_result)
- ListDataQualityStatisticAnnotations 작업(Python: list_data_quality_statistic_annotations)
- PutDataQualityProfileAnnotation 작업(Python: put_data_quality_profile_annotation)
- 민감한 데이터 감지 API
  - 데이터 타입
  - CustomEntityType 구조
  - 운영
  - CreateCustomEntityType 작업(Python: create_custom_entity_type)
  - DeleteCustomEntityType 작업(Python: delete_custom_entity_type)
  - GetCustomEntityType 작업(Python: get_custom_entity_type)
  - BatchGetCustomEntityTypes 작업(Python: batch_get_custom_entity_types)
  - ListCustomEntityTypes 작업(Python: list_custom_entity_types)
- AWS Glue에서 API 태그 지정
  - 데이터 타입
  - 태그 구조
  - 운영
  - TagResource 작업(Python: tag_resource)
  - UntagResource 작업(Python: untag_resource)
  - GetTags 작업(Python: get_tags)
- 공통 데이터 형식
  - 태그 구조
  - DecimalNumber 구조
  - ErrorDetail 구조
  - PropertyPredicate 구조
  - ResourceUri 구조
  - ColumnStatistics 구조
  - ColumnStatisticsError 구조

- ColumnError 구조
- ColumnStatisticsData 구조
- BooleanColumnStatisticsData 구조
- DateColumnStatisticsData 구조
- DecimalColumnStatisticsData 구조
- DoubleColumnStatisticsData 구조
- LongColumnStatisticsData 구조
- StringColumnStatisticsData 구조
- BinaryColumnStatisticsData 구조
- 문자열 패턴

- 예외
  - AccessDeniedException 구조
  - AlreadyExistsException 구조
  - ConcurrentModificationException 구조
  - ConcurrentRunsExceededException 구조
  - CrawlerNotRunningException 구조
  - CrawlerRunningException 구조
  - CrawlerStoppingException 구조
  - EntityNotFoundException 구조
  - FederationSourceException 구조
  - FederationSourceRetryableException 구조
  - GlueEncryptionException 구조
  - IdempotentParameterMismatchException 구조
  - IllegalWorkflowStateException 구조
  - InternalServiceException 구조
  - InvalidExecutionEngineException 구조
  - InvalidInputException 구조
  - InvalidStateException 구조
  - InvalidTaskStatusTransitionException 구조
  - JobDefinitionErrorException 구조

- JobRunInTerminalStateException 구조
- JobRunInvalidStateTransitionException 구조
- JobRunNotInTerminalStateException 구조
- LateRunnerException 구조
- NoScheduleException 구조
- OperationTimeoutException 구조
- ResourceNotReadyException 구조
- ResourceNumberLimitExceededException 구조
- SchedulerNotRunningException 구조
- SchedulerRunningException 구조
- SchedulerTransitioningException 구조
- UnrecognizedRunnerException 구조
- ValidationException 구조
- VersionMismatchException 구조

AWS Glue의 보안 API

보안 API는 보안 데이터 유형 및 AWS Glue의 보안과 관련된 API에 대해 설명합니다.

데이터 타입

- DataCatalogEncryptionSettings 구조
- EncryptionAtRest 구조
- ConnectionPasswordEncryption 구조
- EncryptionConfiguration 구조
- S3Encryption 구조
- CloudWatchEncryption 구조
- JobBookmarksEncryption 구조
- SecurityConfiguration 구조
- GluePolicy 구조
- DataQualityEncryption 구조

DataCatalogEncryptionSettings 구조

데이터 카탈로그 보안을 유지 관리하기 위한 구성 정보가 포함되어 있습니다.

필드

- `EncryptionAtRest` – EncryptionAtRest 객체입니다.

  데이터 카탈로그에 대한 저장 데이터 암호화 구성을 지정합니다.
- `ConnectionPasswordEncryption` – ConnectionPasswordEncryption 객체입니다.

  연결 암호 보호가 활성화된 경우 데이터 카탈로그는 고객 제공 키를 사용하여 `CreateConnection` 또는 `UpdateConnection`의 일부로 암호를 암호화하고 이를 연결 속성의 `ENCRYPTED_PASSWORD` 필드에 저장합니다. 카탈로그 암호화를 활성화하거나 암호 암호화만 활성화할 수 있습니다.

EncryptionAtRest 구조

데이터 카탈로그에 대한 저장 데이터 암호화 구성을 지정합니다.

필드

- `CatalogEncryptionMode` – 필수: UTF-8 문자열입니다(유효한 값: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-KMS-WITH-SERVICE-ROLE="SSEKMSWITHSERVICEROLE"`).

  데이터 카탈로그 데이터를 암호화하기 위한 저장 데이터 암호화 모드입니다.
- `SseAwsKmsKeyId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  저장된 데이터 암호화에 사용할 AWS KMS 키의 ID입니다.
- `CatalogEncryptionServiceRole` – Custom string pattern #51과(와) 일치하는 UTF-8 문자열입니다.

  AWS Glue가 호출자를 대신하여 데이터 카탈로그 객체를 암호화 및 해독하는 역할을 맡습니다.

ConnectionPasswordEncryption 구조

데이터 카탈로그가 `CreateConnection` 또는 `UpdateConnection`의 일부로 암호를 암호화하고 이를 연결 속성의 `ENCRYPTED_PASSWORD` 필드에 저장하기 위해 사용하는 데이터 구조입니다. 카탈로그 암호화를 활성화하거나 암호 암호화만 활성화할 수 있습니다.

암호를 포함하는 CreationConnection 요청이 도착하면 Data Catalog는 먼저 AWS KMS 키를 사용하여 암호를 암호화합니다. 그런 다음 카탈로그 암호화도 활성화되어 있으면 전체 연결 객체를 다시 암호화합니다.

이 암호화를 사용하려면 보안 요구 사항에 따라 암호 키에 대한 액세스를 사용하거나 제한하도록 AWS KMS 키 권한을 설정해야 합니다. 예를 들어 관리자만 암호 키에 대한 암호화 해제 권한을 갖도록 하고자 할 수 있습니다.

필드

- ReturnConnectionPasswordEncrypted – 필수(Required): 부울.

  ReturnConnectionPasswordEncrypted 플래그가 "true"로 설정된 경우, 암호는 GetConnection 및 GetConnections의 응답에서 암호화된 상태로 유지됩니다. 이 암호화는 카탈로그 암호화와는 독립적으로 적용됩니다.
- AwsKmsKeyId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결 암호를 암호화하는 데 사용되는 AWS KMS 키입니다.

  연결 암호 보호가 사용되는 경우 CreateConnection 및 UpdateConnection의 호출자는 Data Catalog에 암호를 저장하기 전에 암호화하려면 지정된 AWS KMS 키에 대한 최소한 kms:Encrypt 권한이 필요합니다.

  보안 요구 사항에 따라 암호 키에 대한 액세스를 허용하거나 제한하도록 암호화 해제 권한을 설정할 수 있습니다.

EncryptionConfiguration 구조

암호화 구성을 지정합니다.

필드

- S3Encryption – S3Encryption 객체의 배열입니다.

  Amazon Simple Storage Service(Amazon S3) 데이터에 대한 암호화 구성.
- CloudWatchEncryption – CloudWatchEncryption 객체입니다.

  Amazon CloudWatch에 대한 암호화 구성.
- JobBookmarksEncryption – JobBookmarksEncryption 객체입니다.

작업 북마크에 대한 암호화 구성.

- `DataQualityEncryption` – DataQualityEncryption 객체입니다.

  AWS Glue Data Quality 자산에 대한 암호화 구성.

S3Encryption 구조

Amazon Simple Storage Service(Amazon S3) 데이터를 암호화하는 방법을 지정합니다.

필드

- `S3EncryptionMode` – UTF-8 문자열입니다(유효한 값: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-S3="SSES3"`).

  Amazon S3 데이터에 사용할 암호화 모드입니다.

- `KmsKeyArn` – Custom string pattern #29과(와) 일치하는 UTF-8 문자열입니다.

  데이터를 암호화하는 데 사용되는 KMS 키의 Amazon 리소스 이름(ARN).

CloudWatchEncryption 구조

Amazon CloudWatch 데이터를 암호화하는 방법을 지정합니다.

필드

- `CloudWatchEncryptionMode` – UTF-8 문자열입니다(유효한 값: `DISABLED` | `SSE-KMS="SSEKMS"`).

  CloudWatch 데이터에 사용할 암호화 모드입니다.

- `KmsKeyArn` – Custom string pattern #29과(와) 일치하는 UTF-8 문자열입니다.

  데이터를 암호화하는 데 사용되는 KMS 키의 Amazon 리소스 이름(ARN).

JobBookmarksEncryption 구조

작업 북마크 데이터를 암호화하는 방법을 지정합니다.

필드

- `JobBookmarksEncryptionMode` – UTF-8 문자열입니다(유효한 값: `DISABLED` | `CSE-KMS="CSEKMS"`).

  작업 북마크 데이터에 사용할 암호화 모드입니다.
- `KmsKeyArn` – Custom string pattern #29과(와) 일치하는 UTF-8 문자열입니다.

  데이터를 암호화하는 데 사용되는 KMS 키의 Amazon 리소스 이름(ARN).

SecurityConfiguration 구조

보안 구성을 지정합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  보안 구성의 이름입니다.
- `CreatedTimeStamp` – 타임스탬프입니다.

  이 보안 구성이 생성된 시간입니다.
- `EncryptionConfiguration` – EncryptionConfiguration 객체입니다.

  이 보안 구성과 관련된 암호화 구성입니다.

GluePolicy 구조

리소스 정책을 반환하기 위한 구조입니다.

필드

- `PolicyInJson` – UTF-8 문자열입니다(최소 2바이트).

  요청된 정책 문서(JSON 형식)가 포함되어 있습니다.
- `PolicyHash` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 정책과 연관된 해시 값을 포함하고 있습니다.
- `CreateTime` – 타임스탬프입니다.

정책이 생성된 날짜와 시간입니다.

- `UpdateTime` – 타임스탬프입니다.

  정책이 업데이트된 날짜와 시간입니다.

DataQualityEncryption 구조

계정의 Data Quality 자산을 암호화하는 방법을 지정합니다.

필드

- `DataQualityEncryptionMode` – UTF-8 문자열입니다(유효한 값: `DISABLED` | `SSE-KMS="SSEKMS"`).

  Data Quality 자산을 암호화하는 데 사용할 암호화 모드. 이러한 자산에는 데이터 품질 규칙 세트, 결과, 통계, 이상 탐지 모델 및 관찰이 포함됩니다.

  유효한 값은 고객 관리형 KMS 키를 사용한 암호화의 경우 `SSEKMS`, 또는 `DISABLED`입니다.

- `KmsKeyArn` – Custom string pattern #29과(와) 일치하는 UTF-8 문자열입니다.

  데이터를 암호화하는 데 사용되는 KMS 키의 Amazon 리소스 이름(ARN).

운영

- GetDataCatalogEncryptionSettings 작업(Python: get_data_catalog_encryption_settings)
- PutDataCatalogEncryptionSettings 작업(Python: put_data_catalog_encryption_settings)
- PutResourcePolicy 작업(Python: put_resource_policy)
- GetResourcePolicy 작업(Python: get_resource_policy)
- DeleteResourcePolicy 작업(Python: delete_resource_policy)
- CreateSecurityConfiguration 작업(Python: create_security_configuration)
- DeleteSecurityConfiguration 작업(Python: delete_security_configuration)
- GetSecurityConfiguration 작업(Python: get_security_configuration)
- GetSecurityConfigurations 작업(Python: get_security_configurations)
- GetResourcePolicies 작업(Python: get_resource_policies)

GetDataCatalogEncryptionSettings 작업(Python: get_data_catalog_encryption_settings)

지정된 카탈로그에 대한 보안 구성을 검색합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  보안 구성을 검색할 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

응답

- `DataCatalogEncryptionSettings` – DataCatalogEncryptionSettings 객체입니다.

  요청된 보안 구성입니다.

오류

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

PutDataCatalogEncryptionSettings 작업(Python: put_data_catalog_encryption_settings)

지정된 카탈로그에 대한 보안 구성을 설정합니다. 구성이 설정되면 지정된 암호화가 이후의 모든 카탈로그 쓰기에 적용됩니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  보안 구성을 설정할 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DataCatalogEncryptionSettings` – 필수(Required): DataCatalogEncryptionSettings 객체입니다.

  설정할 보안 구성입니다.

응답

- 무응답 파라미터.

오류

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

PutResourcePolicy 작업(Python: put_resource_policy)

액세스 제어를 위한 데이터 카탈로그 리소스 정책을 설정합니다.

요청

- `PolicyInJson` – 필수: UTF-8 문자열입니다(최소 2바이트).

  설정할 정책 문서(JSON 형식)가 포함되어 있습니다.

- `ResourceArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  사용하지 않습니다. 내부용입니다.

- `PolicyHashCondition` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이전 정책이 `PutResourcePolicy`를 사용하여 설정된 경우 반환되는 해시 값입니다. 정책의 동시 수정을 방지하기 위해 사용됩니다. 설정된 이전 정책이 없는 경우에는 이 파라미터를 사용하지 마십시오.

- `PolicyExistsCondition` – UTF-8 문자열입니다(유효 값: `MUST_EXIST` | `NOT_EXIST` | `NONE`).

MUST_EXIST의 값은 정책을 업데이트하는 데 사용됩니다. NOT_EXIST의 값은 새 정책을 생성하는 데 사용됩니다. NONE의 값 또는 null 값이 사용되는 경우 호출은 정책의 존재 여부에 종속되지 않습니다.

- EnableHybrid – UTF-8 문자열입니다(유효한 값: TRUE | FALSE).

  'TRUE'이면 Data Catalog 리소스에 대한 교차 계정 액세스 권한을 부여하기 위해 두 가지 방법을 모두 사용하고 있음을 나타냅니다.

  - PutResourePolicy로 리소스 정책을 직접 업데이트하여
  - AWS Management Console에서 권한 부여 명령을 사용합니다.

  이미 관리 콘솔을 사용하여 교차 계정 액세스 권한을 부여한 경우 'TRUE'로 설정해야 합니다. 그렇지 않으면 호출이 실패합니다. 기본값은 'FALSE'입니다.

응답

- PolicyHash – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  방금 설정한 정책의 해시입니다. 이 정책을 덮어쓰거나 업데이트하는 후속 호출에 반드시 포함되어 있어야 합니다.

오류

- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- InvalidInputException
- ConditionCheckFailureException

GetResourcePolicy 작업(Python: get_resource_policy)

지정된 리소스 정책을 검색합니다.

요청

- ResourceArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

리소스 정책을 검색할 AWS Glue 리소스의 ARN입니다. 제공하지 않으면 Data Catalog 리소스 정책이 반환됩니다. `GetResourcePolicies`를 사용하여 기존 리소스 정책을 모두 봅니다. 자세한 내용은 AWS Glue 리소스 ARN 지정을 참조하세요.

응답

- `PolicyInJson` – UTF-8 문자열입니다(최소 2바이트).

  요청된 정책 문서(JSON 형식)가 포함되어 있습니다.

- `PolicyHash` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 정책과 연관된 해시 값을 포함하고 있습니다.

- `CreateTime` – 타임스탬프입니다.

  정책이 생성된 날짜와 시간입니다.

- `UpdateTime` – 타임스탬프입니다.

  정책이 업데이트된 날짜와 시간입니다.

오류

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

DeleteResourcePolicy 작업(Python: delete_resource_policy)

지정된 정책을 삭제합니다.

요청

- `PolicyHashCondition` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 정책이 설정되었을 때 반환된 해시 값입니다.

- `ResourceArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  삭제할 리소스 정책에 대한 AWS Glue 리소스의 ARN입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ConditionCheckFailureException`

CreateSecurityConfiguration 작업(Python: create_security_configuration)

새로운 보안 구성을 생성합니다. 보안 구성은 AWS Glue에서 사용할 수 있는 보안 속성 세트입니다. 보안 구성을 사용하여 미사용 데이터를 암호화할 수 있습니다. AWS Glue에서 보안 구성 사용에 대한 자세한 내용은 크롤러, 작업 및 개발 엔드포인트로 기록된 데이터 암호화를 참조하세요.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  새 보안 구성의 이름입니다.

- `EncryptionConfiguration` – 필수(Required): EncryptionConfiguration 객체입니다.

  새 보안 구성을 위한 암호화 구성입니다.

응답

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

새 보안 구성에 할당된 이름입니다.

- `CreatedTimestamp` – 타임스탬프입니다.

  새 보안 구성이 생성된 시간입니다.

오류

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

DeleteSecurityConfiguration 작업(Python: delete_security_configuration)

지정된 보안 구성을 삭제합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 보안 구성의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetSecurityConfiguration 작업(Python: get_security_configuration)

지정된 보안 구성을 검색합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색할 보안 구성의 이름입니다.

응답

- `SecurityConfiguration` – SecurityConfiguration 객체입니다.

  요청된 보안 구성입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetSecurityConfigurations 작업(Python: get_security_configurations)

모든 보안 구성 목록을 검색합니다.

요청

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `SecurityConfigurations` – SecurityConfiguration 객체의 배열입니다.

  보안 구성의 목록입니다.
- `NextToken` – UTF-8 문자열입니다.

  반환할 보안 구성이 더 있는 경우 연속 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetResourcePolicies 작업(Python: get_resource_policies)

교차 계정 권한 부여 중에 AWS Resource Access Manager가 개별 리소스에 설정한 리소스 정책을 검색합니다. 또한 Data Catalog 리소스 정책을 검색합니다.

Data Catalog 설정에서 메타데이터 암호화를 사용하고 AWS KMS 키에 대한 권한이 없으면 작업에서 Data Catalog 리소스 정책을 반환할 수 없습니다.

요청

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 최대 크기.

응답

- `GetResourcePoliciesResponseList` – GluePolicy 객체의 배열입니다.

  개별 리소스 정책 및 계정 수준 리소스 정책 목록입니다.

- NextToken – UTF-8 문자열입니다.

  반환된 목록이 사용 가능한 마지막 리소스 정책을 포함하지 경우의 연속 토큰입니다.

오류

- InternalServiceException
- OperationTimeoutException
- InvalidInputException
- GlueEncryptionException

카탈로그 객체 API

카탈로그 객체 API는 AWS Glue에서의 카탈로그 작업과 관련된 API 및 데이터 유형에 대해 설명합니다.

주제

- 카탈로그 API
- 데이터베이스 API
- 표 API
- 파티션 API
- 연결 API
- 사용자 정의 함수 API
- Athena 카탈로그를 AWS Glue로 가져오기

카탈로그 API

카탈로그 API는 카탈로그 생성, 삭제, 찾기, 업데이트 및 나열을 위한 API를 설명합니다.

데이터 타입

- 카탈로그 구조
- CatalogInput 구조
- TargetRedshiftCatalog 구조

- CatalogProperties 구조
- CatalogPropertiesOutput 구조
- DataLakeAccessProperties 구조
- DataLakeAccessPropertiesOutput 구조
- FederatedCatalog 구조

카탈로그 구조

카탈로그 객체는 AWS Glue Data Catalog 또는 페더레이션 소스에 있는 데이터베이스의 논리적 그룹을 나타냅니다. 이제 Redshift 페더레이션 카탈로그를 생성하거나, 다른 계정 또는 리전의 Redshift 데이터베이스에 대한 리소스 링크가 포함된 카탈로그를 생성할 수 있습니다.

필드

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  카탈로그의 ID입니다. 기본 카탈로그에 대한 액세스 권한을 부여하려면 이 필드를 제공해서는 안 됩니다.
- `Name` – 필수: Custom string pattern #25과(와) 일치하는 1~64바이트 길이의 UTF-8 문자열입니다.

  카탈로그의 이름입니다. 계정 ID와 같을 수 없습니다.
- `ResourceArn` – UTF-8 문자열입니다.

  카탈로그 리소스에 할당된 Amazon 리소스 이름(ARN)입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  여러 줄로 이루어진 URI 주소 문자열 패턴과 일치하는 2,048바이트 이하 길이의 설명 문자열입니다. 카탈로그에 대한 설명입니다.
- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  카탈로그의 파라미터와 속성을 정의하는 키-값 페어의 맵 배열입니다.
- `CreateTime` – 타임스탬프입니다.

카탈로그가 생성된 시간입니다.

- `UpdateTime` – 타임스탬프입니다.

  카탈로그가 마지막으로 업데이트된 시간입니다.

- `TargetRedshiftCatalog` – TargetRedshiftCatalog 객체입니다.

  데이터베이스 리소스 링크에 대한 대상 카탈로그를 설명하는 `TargetRedshiftCatalog` 객체입니다.

- `FederatedCatalog` – FederatedCatalog 객체입니다.

  AWS Glue Data Catalog 외부의 엔터티를 가리키는 `FederatedCatalog` 객체입니다.

- `CatalogProperties` – CatalogPropertiesOutput 객체입니다.

  데이터 레이크 액세스 속성 및 기타 사용자 지정 속성을 지정하는 `CatalogProperties` 객체입니다.

- `CreateTableDefaultPermissions` – PrincipalPermissions 객체의 배열입니다.

  `PrincipalPermissions` 객체 어레이. 위탁자에 대한 테이블에서 기본 권한 세트를 생성합니다. AWS Lake Formation에서 사용됩니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다.

- `CreateDatabaseDefaultPermissions` – PrincipalPermissions 객체의 배열입니다.

  `PrincipalPermissions` 객체 어레이. 위탁자에 대한 데이터베이스에서 기본 권한 세트를 생성합니다. AWS Lake Formation에서 사용됩니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다.

- `AllowFullTableExternalDataAccess` – UTF-8 문자열입니다(유효한 값: `True` | `False`).

  타사 엔진이 Lake Formation에 등록된 Amazon S3 위치의 데이터에 액세스할 수 있도록 허용합니다.

CatalogInput 구조

카탈로그 속성을 설명하는 구조입니다.

필드

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

여러 줄로 이루어진 URI 주소 문자열 패턴과 일치하는 2,048바이트 이하 길이의 설명 문자열입니다. 카탈로그에 대한 설명입니다.

- `FederatedCatalog` – FederatedCatalog 객체입니다.

  `FederatedCatalog` 객체입니다. AWS Glue Data Catalog 외부의 객체(예: Redshift 데이터베이스)를 참조하는 `FederatedCatalog` 구조입니다.

- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  카탈로그의 파라미터와 속성을 정의하는 키-값 페어의 맵 배열입니다.

- `TargetRedshiftCatalog` – TargetRedshiftCatalog 객체입니다.

  리소스 링크에 대한 대상 카탈로그를 설명하는 `TargetRedshiftCatalog` 객체입니다.

- `CatalogProperties` – CatalogProperties 객체입니다.

  데이터 레이크 액세스 속성 및 기타 사용자 지정 속성을 지정하는 `CatalogProperties` 객체입니다.

- `CreateTableDefaultPermissions` – PrincipalPermissions 객체의 배열입니다.

  `PrincipalPermissions` 객체 어레이. 위탁자에 대한 테이블에서 기본 권한 세트를 생성합니다. AWS Lake Formation에서 사용됩니다. 일반적으로, 빈 목록으로 명시적으로 설정해야 합니다.

- `CreateDatabaseDefaultPermissions` – PrincipalPermissions 객체의 배열입니다.

  `PrincipalPermissions` 객체 어레이. 위탁자에 대한 데이터베이스에서 기본 권한 세트를 생성합니다. AWS Lake Formation에서 사용됩니다. 일반적으로, 빈 목록으로 명시적으로 설정해야 합니다.

- `AllowFullTableExternalDataAccess` – UTF-8 문자열입니다(유효한 값: `True` | `False`).

  타사 엔진이 Lake Formation에 등록된 Amazon S3 위치의 데이터에 액세스할 수 있도록 허용합니다.

TargetRedshiftCatalog 구조

리소스 링크에 대한 대상 카탈로그를 설명하는 구조입니다.

필드

- `CatalogArn` – 필수: UTF-8 문자열입니다.

  카탈로그 리소스의 Amazon 리소스 이름(ARN)입니다.

CatalogProperties 구조

데이터 레이크 액세스 속성 및 기타 사용자 지정 속성을 지정하는 구조입니다.

필드

- `DataLakeAccessProperties` – DataLakeAccessProperties 객체입니다.

  AWS Glue Data Catalog에서 카탈로그 리소스의 데이터 레이크 액세스를 구성하기 위한 속성을 지정하는 `DataLakeAccessProperties` 객체입니다.
- `CustomProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  열 통계 최적화와 같은, 카탈로그의 추가 키-값 속성입니다.

CatalogPropertiesOutput 구조

카탈로그 리소스에 대한 구성 속성을 포함하는 속성 특성입니다.

필드

- `DataLakeAccessProperties` – DataLakeAccessPropertiesOutput 객체입니다.

  AWS Glue Data Catalog에서 카탈로그 리소스의 데이터 레이크 액세스를 구성하기 위한 입력 속성이 있는 `DataLakeAccessProperties` 객체입니다.
- `CustomProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  열 통계 최적화와 같은, 카탈로그의 추가 키-값 속성입니다.

DataLakeAccessProperties 구조

AWS Glue Data Catalog에서 카탈로그 리소스의 데이터 레이크 액세스를 구성하기 위한 입력 속성입니다.

필드

- `DataLakeAccess` – 부울입니다.

  Amazon Athena, Amazon EMR, AWS Glue ETL 등 Redshift 이외의 엔진에서 Data Catalog의 Amazon Redshift 데이터베이스에 액세스하는 Apache Spark 애플리케이션의 데이터 레이크 액세스를 활성화하거나 비활성화합니다.
- `DataTransferRole` – Custom string pattern #51과(와) 일치하는 UTF-8 문자열입니다.

  쿼리 중에 스테이징 안팎으로 데이터를 전송하기 위해 AWS Glue가 수임할 역할입니다.
- `KmsKey` – UTF-8 문자열입니다.

  카탈로그와 함께 생성되는 스테이징 버킷에 사용할 암호화 키입니다.
- `CatalogType` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  네이티브 카탈로그 리소스의 페더레이션 카탈로그 유형을 지정합니다. 현재 지원되는 유형은 `aws:redshift`입니다.

DataLakeAccessPropertiesOutput 구조

AWS Glue Data Catalog의 카탈로그 리소스에 대한 데이터 레이크 액세스 구성의 출력 속성입니다.

필드

- `DataLakeAccess` – 부울입니다.

  Data Catalog에서 Amazon Redshift 데이터베이스에 액세스하는 Apache Spark 애플리케이션의 데이터 레이크 액세스를 활성화하거나 비활성화합니다.
- `DataTransferRole` – Custom string pattern #51과(와) 일치하는 UTF-8 문자열입니다.

  쿼리 중에 스테이징 안팎으로 데이터를 전송하기 위해 AWS Glue가 수임할 역할입니다.
- `KmsKey` – UTF-8 문자열입니다.

  카탈로그와 함께 생성되는 스테이징 버킷에 사용할 암호화 키입니다.

- `ManagedWorkgroupName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  카탈로그 리소스에 대해 생성된 관리형 Redshift Serverless 컴퓨팅 이름입니다.

- `ManagedWorkgroupStatus` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  관리형 Redshift Serverless 컴퓨팅 상태입니다.

- `RedshiftDatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  관리형 컴퓨팅의 기본 Redshift 데이터베이스 리소스 이름입니다.

- `StatusMessage` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  관리형 작업 그룹 상태에 대한 세부 정보를 제공하는 메시지입니다.

- `CatalogType` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  네이티브 카탈로그 리소스의 페더레이션 카탈로그 유형을 지정합니다. 현재 지원되는 유형은 `aws:redshift`입니다.

FederatedCatalog 구조

AWS Glue Data Catalog 외부의 엔터티를 가리키는 카탈로그입니다.

필드

- `Identifier` – Single-line string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  페더레이션 카탈로그의 고유 식별자입니다.

- `ConnectionName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Redshift 페더레이션 카탈로그와 같은 외부 데이터 소스에 대한 연결의 이름입니다.

- `ConnectionType` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

페더레이션 카탈로그에 액세스하는 데 사용되는 연결 유형으로, 외부 데이터 소스에 연결하기 위한 프로토콜 또는 방법을 지정합니다.

운영

- CreateCatalog 작업(Python: create_catalog)
- UpdateCatalog 작업(Python: update_catalog)
- DeleteCatalog 작업(Python: delete_catalog)
- GetCatalog 작업(Python: get_catalog)
- GetCatalogs 작업(Python: get_catalogs)

CreateCatalog 작업(Python: create_catalog)

AWS Glue Data Catalog에 새 카탈로그를 생성합니다.

요청

- `Name` – 필수: Custom string pattern #25과(와) 일치하는 1~64바이트 길이의 UTF-8 문자열입니다.

  생성할 카탈로그의 이름입니다.

- `CatalogInput` – 필수(Required): CatalogInput 객체입니다.

  카탈로그의 메타데이터를 정의하는 `CatalogInput` 객체입니다.

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다. 각 키는 길이가 1~128바이트인 UTF-8 문자열입니다. 각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다. 카탈로그에 할당하는 태그입니다.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `FederatedResourceAlreadyExistsException`
- `FederationSourceException`

UpdateCatalog 작업(Python: update_catalog)

AWS Glue Data Catalog에서 기존 카탈로그의 속성을 업데이트합니다.

요청

- `CatalogId` – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  카탈로그의 ID입니다.
- `CatalogInput` – 필수(Required): CatalogInput 객체입니다.

  기존 카탈로그의 새 속성을 지정하는 `CatalogInput` 객체입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`

- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- ConcurrentModificationException
- AccessDeniedException
- FederationSourceException

DeleteCatalog 작업(Python: delete_catalog)

AWS Glue Data Catalog에서 지정된 카탈로그를 제거합니다.

이 작업을 완료한 후에는 데이터베이스, 테이블(및 테이블에 속하는 모든 테이블 버전과 파티션)은 물론 삭제된 카탈로그에 있는 사용자 정의 함수에 더 이상 액세스하지 못합니다. AWS Glue는 이러한 "분리된" 리소스를 서비스 재량에 따라 적시에 비동기로 삭제합니다

관련된 모든 리소스가 즉시 삭제되도록 DeleteCatalog 작업을 직접적으로 호출하기 전에 DeleteTableVersion(또는 BatchDeleteTableVersion), DeletePartition(또는 BatchDeletePartition), DeleteTable(또는 BatchDeleteTable), DeleteUserDefinedFunction 및 DeleteDatabase를 사용하여 카탈로그에 속하는 모든 리소스를 삭제하세요.

요청

- CatalogId – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  카탈로그의 ID입니다.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException

- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- ConcurrentModificationException
- AccessDeniedException
- FederationSourceException

GetCatalog 작업(Python: get_catalog)

가져올 카탈로그의 이름입니다. 모두 소문자여야 합니다.

요청

- CatalogId – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  카탈로그가 있는 상위 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 번호가 사용됩니다.

응답

- Catalog – 카탈로그 객체입니다.

  Catalog 객체입니다. AWS Glue Data Catalog에서 지정된 카탈로그의 정의입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- AccessDeniedException
- EntityNotFoundException
- FederationSourceException

- FederationSourceRetryableException

GetCatalogs 작업(Python: get_catalogs)

AWS Glue Data Catalog의 카탈로그에 정의된 모든 카탈로그를 검색합니다. Redshift 페더레이션 카탈로그 사용 사례의 경우, 이 작업은 Redshift 네임스페이스 카탈로그의 Redshift 데이터베이스에 매핑된 카탈로그 목록을 반환합니다.

요청

- ParentCatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  카탈로그가 있는 상위 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 번호가 사용됩니다.

- NextToken – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

- MaxResults – 1~1,000의 숫자(정수)입니다.

  한 번의 응답으로 반환될 최대 카탈로그 수입니다.

- Recursive – 부울입니다.

  ParentCatalogId부터 시작하여 카탈로그 계층 구조 전반의 모든 카탈로그를 나열할지 여부를 나타냅니다. 기본값은 false입니다. true인 경우 ParentCatalogID 계층 구조의 모든 카탈로그 객체가 응답에 열거됩니다.

- IncludeRoot – 부울입니다.

  응답에서 계정 및 리전의 기본 카탈로그를 나열할지 여부를 나타냅니다. 기본값은 false입니다. true 및 ParentCatalogId = NULL | AWS Account ID인 경우 모든 카탈로그와 기본 카탈로그가 응답에 열거됩니다.

  ParentCatalogId가 null이 아니고, 이 속성이 false 또는 true로 전달되면 InvalidInputException이 발생합니다.

응답

- CatalogList – 필수(Required): 카탈로그 객체의 배열입니다.

Catalog 객체 어레이. 지정된 상위 카탈로그의 Catalog 목록입니다.

- NextToken – UTF-8 문자열입니다.

  목록의 현재 세그먼트가 마지막이 아니면 반환된 토큰 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

오류

- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- AccessDeniedException
- EntityNotFoundException
- FederationSourceException
- FederationSourceRetryableException

데이터베이스 API

데이터베이스 API는 데이터베이스 데이터 형식을 설명하며, 데이터베이스를 생성, 삭제, 위치 지정, 업데이트 및 목록화하는 API를 포함합니다.

데이터 타입

- 데이터베이스 구조
- DatabaseInput 구조
- PrincipalPermissions 구조
- DataLakePrincipal 구조
- DatabaseIdentifier 구조
- FederatedDatabase 구조

데이터베이스 구조

Database 객체는 Hive 메타스토어 혹은 RDBMS에 있는 테이블의 논리적 그룹을 나타냅니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터베이스의 이름입니다. 저장될 때 소문자로 저장되어 Hive 호환성을 유지합니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  데이터베이스에 대한 설명입니다.

- LocationUri – URI address multi-line string pattern과(와) 일치하는 1~1,024바이트 길이의 URI(Uniform Resource Identifier)입니다.

  데이터베이스의 위치(예: HDFS 경로)입니다.

- Parameters – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 키-값 쌍은 데이터베이스의 파라미터와 속성을 정의합니다.

- CreateTime – 타임스탬프입니다.

  메타 데이터베이스가 카탈로그에 생성된 시간입니다.

- CreateTableDefaultPermissions – PrincipalPermissions 객체의 배열입니다.

  보안 주체에 대한 테이블에서 기본 권한 세트를 생성합니다. AWS Lake Formation에서 사용됩니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다.

- TargetDatabase – DatabaseIdentifier 객체입니다.

  리소스 링크에 대한 대상 데이터베이스를 설명하는 DatabaseIdentifier 구조입니다.

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  데이터베이스가 있는 데이터 카탈로그의 ID입니다.

- FederatedDatabase – FederatedDatabase 객체입니다.

  AWS Glue Data Catalog 외부의 엔티티를 참조하는 FederatedDatabase 구조입니다.

DatabaseInput 구조

데이터베이스를 생성 및 업데이트할 때 사용되는 구조입니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터베이스의 이름입니다. 저장될 때 소문자로 저장되어 Hive 호환성을 유지합니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  데이터베이스에 대한 설명입니다.

- LocationUri – URI address multi-line string pattern과(와) 일치하는 1~1,024바이트 길이의 URI(Uniform Resource Identifier)입니다.

  데이터베이스의 위치(예: HDFS 경로)입니다.

- Parameters – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 키-값 쌍은 데이터베이스의 파라미터와 속성을 정의합니다.

  이러한 키-값 쌍은 데이터베이스의 파라미터와 속성을 정의합니다.

- CreateTableDefaultPermissions – PrincipalPermissions 객체의 배열입니다.

  보안 주체에 대한 테이블에서 기본 권한 세트를 생성합니다. AWS Lake Formation에서 사용됩니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다.

- TargetDatabase – DatabaseIdentifier 객체입니다.

  리소스 링크에 대한 대상 데이터베이스를 설명하는 DatabaseIdentifier 구조입니다.

- FederatedDatabase – FederatedDatabase 객체입니다.

  AWS Glue Data Catalog 외부의 엔터티를 참조하는 FederatedDatabase 구조입니다.

PrincipalPermissions 구조

보안 주체에게 부여된 권한입니다.

필드

- `Principal` – DataLakePrincipal 객체입니다.

  권한을 부여받는 보안 주체입니다.
- `Permissions` – UTF-8 문자열의 배열입니다.

  보안 주체에게 부여되는 권한입니다.

DataLakePrincipal 구조

AWS Lake Formation 보안 주체입니다.

필드

- `DataLakePrincipalIdentifier` – 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Lake Formation 보안 주체의 식별자입니다.

DatabaseIdentifier 구조

리소스 링크에 대한 대상 데이터베이스를 설명하는 구조입니다.

필드

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  데이터베이스가 있는 데이터 카탈로그의 ID입니다.
- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  카탈로그 데이터베이스의 이름입니다.
- `Region` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  대상 데이터베이스의 리전입니다.

FederatedDatabase 구조

AWS Glue Data Catalog 외부의 엔터티를 가리키는 데이터베이스입니다.

필드

- `Identifier` – Single-line string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  페더레이션된 데이터베이스의 고유 식별자입니다.
- `ConnectionName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  외부 메타스토어에 대한 연결 이름입니다.
- `ConnectionType` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  JDBC, ODBC 또는 기타 지원되는 연결 프로토콜과 같은 페더레이션 데이터베이스 액세스에 사용되는 연결 유형입니다.

운영

- CreateDatabase 작업(Python: create_database)
- UpdateDatabase 작업(Python: update_database)
- DeleteDatabase 작업(Python: delete_database)
- GetDatabase 작업(Python: get_database)
- GetDatabases 작업(Python: get_database)

CreateDatabase 작업(Python: create_database)

데이터 카탈로그에 새로운 데이터베이스를 생성합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  데이터베이스를 생성할 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseInput` – 필수(Required): DatabaseInput 객체입니다.

  데이터베이스의 메타데이터입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  데이터베이스에 할당하는 태그입니다.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `FederatedResourceAlreadyExistsException`
- `FederationSourceException`
- `FederationSourceRetryableException`

UpdateDatabase 작업(Python: update_database)

데이터 카탈로그에서 기존 데이터베이스를 업데이트합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

메타데이터 데이터베이스가 있는 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  카탈로그에서 업데이트할 데이터베이스의 이름입니다. 저장될 때 소문자로 저장되어 Hive 호환성을 유지합니다.

- DatabaseInput – 필수(Required): DatabaseInput 객체입니다.

  DatabaseInput 객체는 카탈로그에서 메타데이터 데이터베이스의 새로운 정의를 지정합니다.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- ConcurrentModificationException
- FederationSourceException
- FederationSourceRetryableException
- AlreadyExistsException

DeleteDatabase 작업(Python: delete_database)

데이터 카탈로그에서 지정된 데이터베이스를 제거합니다.

> **Note**
>
> 이 작업을 완료한 후에는 테이블(및 테이블에 속하는 모든 테이블 버전과 파티션)은 물론 삭제된 데이터베이스에 있는 사용자 정의 함수에 더 이상 액세스하지 못합니다. AWS Glue는 이러한 "분리된" 리소스를 서비스 재량에 따라 적시에 비동기로 삭제합니다
> 관련된 모든 리소스가 즉시 삭제되도록 `DeleteDatabase` 호출 전에 `DeleteTableVersion` 또는 `BatchDeleteTableVersion`, `DeletePartition` 또는 `BatchDeletePartition`, `DeleteUserDefinedFunction` 및 `DeleteTable` 또는 `BatchDeleteTable`를 사용하여 데이터베이스에 속하는 모든 리소스를 삭제하십시오.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  데이터베이스가 있는 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 데이터베이스의 이름입니다. 반드시 모두 소문자로 저장하여 Hive 호환성을 유지하도록 합니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `FederationSourceException`

- FederationSourceRetryableException

GetDatabase 작업(Python: get_database)

지정된 데이터베이스의 정의를 검색합니다.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  데이터베이스가 있는 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색할 데이터베이스의 이름입니다. 반드시 모두 소문자로 저장하여 Hive 호환성을 유지하도록 합니다.

응답

- Database – 데이터베이스 객체입니다.

  데이터 카탈로그에서 지정된 데이터베이스의 정의입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- FederationSourceException
- FederationSourceRetryableException

GetDatabases 작업(Python: get_database)

주어진 데이터 카탈로그에 정의된 모든 데이터베이스를 검색합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  `Databases`를 검색할 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

- `MaxResults` – 1~100의 숫자(정수)입니다.

  한 번의 응답으로 반환될 최대 데이터베이스 수입니다.

- `ResourceShareType` – UTF-8 문자열입니다(유효한 값: `FOREIGN` | `ALL` | `FEDERATED`).

  계정과 공유된 데이터베이스를 나열하도록 지정할 수 있습니다. 허용 가능한 값은 `FEDERATED`, `FOREIGN` 또는 `ALL`입니다.

  - `FEDERATED`로 설정하면 계정과 공유된 페더레이션된 데이터베이스(외부 엔터티라고 함)가 나열됩니다.
  - `FOREIGN`으로 설정하면 계정과 공유된 데이터베이스가 나열됩니다.
  - `ALL`로 설정하면 계정과 공유된 데이터베이스와 로컬 계정의 데이터베이스가 나열됩니다.

- `AttributesToGet` – UTF-8 문자열의 배열입니다.

  `GetDatabases` 호출에서 반환된 데이터베이스 필드를 지정합니다. 이 파라미터는 빈 목록을 허용하지 않습니다. 요청에는 `NAME`이 포함되어야 합니다.

응답

- `DatabaseList` – 필수(Required): 데이터베이스 객체의 배열입니다.

  지정된 카탈로그의 `Database` 목록입니다.

- `NextToken` – UTF-8 문자열입니다.

목록의 현재 세그먼트가 마지막이 아니면 반환된 토큰 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

오류

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `EntityNotFoundException`
- `FederationSourceException`
- `FederationSourceRetryableException`

표 API

테이블 API는 테이블과 관련된 데이터 형식 및 작업에 대해 설명합니다.

데이터 타입

- 테이블 구조
- TableInput 구조
- FederatedTable 구조
- 열 구조
- StorageDescriptor 구조
- SchemaReference 구조
- SerDeInfo 구조
- Order 구조
- SkewedInfo 구조
- TableVersion 구조
- TableError 구조
- TableVersionError 구조
- SortCriterion 구조

- TableIdentifier 구조
- KeySchemaElement 구조
- PartitionIndex 구조
- PartitionIndexDescriptor 구조
- BackfillError 구조
- IcebergInput 구조
- OpenTableFormatInput 구조
- ViewDefinition 구조
- ViewDefinitionInput 구조
- ViewRepresentation 구조
- ViewRepresentationInput 구조
- UpdateOpenTableFormatInput 구조
- UpdateIcebergInput 구조
- CreateIcebergTableInput 구조
- UpdateIcebergTableInput 구조
- IcebergSortOrder 구조
- IcebergSortField 구조
- IcebergPartitionSpec 구조
- IcebergPartitionField 구조
- IcebergSchema 구조
- IcebergStructField 구조
- IcebergTableUpdate 구조

테이블 구조

행과 열에 조직된 관련 데이터 모음을 보여줍니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 이름. 반드시 모두 소문자로 저장하여 Hive 호환성을 유지하도록 합니다.

- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 메타데이터가 있는 데이터베이스의 이름입니다. 반드시 모두 소문자로 저장하여 Hive 호환성을 유지하도록 합니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  테이블에 대한 설명입니다.

- `Owner` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 소유자입니다.

- `CreateTime` – 타임스탬프입니다.

  테이블 정의가 데이터 카탈로그에 생성된 시간입니다.

- `UpdateTime` – 타임스탬프입니다.

  테이블이 업데이트된 마지막 시간입니다.

- `LastAccessTime` – 타임스탬프입니다.

  테이블이 액세스된 마지막 시간입니다. 보통 HDFS에서 가져왔고 대다수는 신뢰할 만하지 않습니다.

- `LastAnalyzedTime` – 타임스탬프입니다.

  이 테이블에 대한 열 통계가 계산된 마지막 시간입니다.

- `Retention` – None 이하의 숫자(정수)입니다.

  이 테이블의 보관 기간입니다.

- `StorageDescriptor` – StorageDescriptor 객체입니다.

  이 테이블의 물리적 스토리지에 대한 정보를 포함하는 스토리지 서술자입니다.

- `PartitionKeys` – 열 객체의 배열입니다.

  테이블을 분할할 열의 목록입니다. 초기 유형만 파티션 키로써 지원됩니다.

  Amazon Athena에서 사용하는 테이블을 생성할 때 `partitionKeys`를 지정하지 않은 경우, 최소한 값 `partitionKeys`를 빈 목록으로 설정해야 합니다. 예:

  ```
  "PartitionKeys": []
  ```

- `ViewOriginalText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  Apache Hive 호환성을 위해 포함되었습니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다. 테이블이 `VIRTUAL_VIEW`인 경우 특정 Athena 구성이 base64로 인코딩됩니다.

- `ViewExpandedText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  Apache Hive 호환성을 위해 포함되었습니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다.

- `TableType` – 255바이트 이하 길이의 UTF-8 문자열입니다.

  이 테이블의 유형입니다. AWS Glue에서 `EXTERNAL_TABLE` 유형으로 테이블을 생성합니다. Athena 등의 다른 서비스에서는 추가 테이블 유형으로 테이블을 생성할 수 있습니다.

  AWS Glue 관련 테이블 유형:

  EXTERNAL_TABLE

  Hive 호환 속성 - Hive 관리형이 아닌 테이블을 나타냅니다.

  GOVERNED

  AWS Lake Formation에서 사용됩니다. AWS Glue 데이터 카탈로그는 `GOVERNED`를 이해합니다.

- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 키-값 쌍은 테이블과 관련된 속성을 정의합니다.

- `CreatedBy` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블을 만든 사용자 혹은 개체.

- `IsRegisteredWithLakeFormation` – 부울입니다.

  테이블이 AWS Lake Formation에 등록되었는지 여부를 나타냅니다.

- `TargetTable` – TableIdentifier 객체입니다.

  리소스 링크에 대한 대상 테이블을 설명하는 `TableIdentifier` 구조입니다.

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

테이블이 위치한 데이터 카탈로그의 ID입니다.

- `VersionId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 버전의 ID입니다.

- `FederatedTable` – FederatedTable 객체입니다.

  AWS Glue Data Catalog 외부의 엔터티를 참조하는 `FederatedTable` 구조입니다.

- `ViewDefinition` – ViewDefinition 객체입니다.

  뷰에 대한 하나 이상의 언어와 쿼리 등 뷰를 정의하는 모든 정보를 포함하는 구조입니다.

- `IsMultiDialectView` – 부울입니다.

  뷰가 여러 다양한 쿼리 엔진의 SQL 언어의 지원하며 해당 엔진에서 읽을 수 있는지 여부를 지정합니다.

TableInput 구조

테이블을 정의하는 데 사용된 구조입니다.

필드

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 이름. 저장될 때 소문자로 저장되어 Hive 호환성을 유지합니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  테이블에 대한 설명입니다.

- `Owner` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 소유자입니다. Apache Hive 호환성을 위해 포함되었습니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다.

- `LastAccessTime` – 타임스탬프입니다.

  테이블이 액세스된 마지막 시간입니다.

- `LastAnalyzedTime` – 타임스탬프입니다.

이 테이블에 대한 열 통계가 계산된 마지막 시간입니다.

- `Retention` – None 이하의 숫자(정수)입니다.

  이 테이블의 보관 기간입니다.

- `StorageDescriptor` – StorageDescriptor 객체입니다.

  이 테이블의 물리적 스토리지에 대한 정보를 포함하는 스토리지 서술자입니다.

- `PartitionKeys` – 열 객체의 배열입니다.

  테이블을 분할할 열의 목록입니다. 초기 유형만 파티션 키로써 지원됩니다.

  Amazon Athena에서 사용하는 테이블을 생성할 때 `partitionKeys`를 지정하지 않은 경우, 최소한 값 `partitionKeys`를 빈 목록으로 설정해야 합니다. 예:

  `"PartitionKeys": []`

- `ViewOriginalText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  Apache Hive 호환성을 위해 포함되었습니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다. 테이블이 `VIRTUAL_VIEW`인 경우 특정 Athena 구성이 base64로 인코딩됩니다.

- `ViewExpandedText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  Apache Hive 호환성을 위해 포함되었습니다. AWS Glue 작업의 일반적인 과정에서는 사용되지 않습니다.

- `TableType` – 255바이트 이하 길이의 UTF-8 문자열입니다.

  이 테이블의 유형입니다. AWS Glue에서 `EXTERNAL_TABLE` 유형으로 테이블을 생성합니다. Athena 등의 다른 서비스에서는 추가 테이블 유형으로 테이블을 생성할 수 있습니다.

  AWS Glue 관련 테이블 유형:

  EXTERNAL_TABLE

  Hive 호환 속성 - Hive 관리형이 아닌 테이블을 나타냅니다.

  GOVERNED

  AWS Lake Formation에서 사용됩니다. AWS Glue 데이터 카탈로그는 `GOVERNED`를 이해합니다.

- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

이러한 키-값 쌍은 테이블과 관련된 속성을 정의합니다.

- `TargetTable` – TableIdentifier 객체입니다.

  리소스 링크에 대한 대상 테이블을 설명하는 `TableIdentifier` 구조입니다.

- `ViewDefinition` – ViewDefinitionInput 객체입니다.

  뷰에 대한 하나 이상의 언어와 쿼리 등 뷰를 정의하는 모든 정보를 포함하는 구조입니다.

FederatedTable 구조

AWS Glue Data Catalog 외부의 엔터티를 가리키는 테이블입니다.

필드

- `Identifier` – Single-line string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  페더레이션된 테이블의 고유 식별자입니다.

- `DatabaseIdentifier` – Single-line string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  페더레이션된 데이터베이스의 고유 식별자입니다.

- `ConnectionName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  외부 메타스토어에 대한 연결 이름입니다.

- `ConnectionType` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  페더레이션 테이블에 액세스하는 데 사용되는 연결 유형으로, 외부 데이터 소스에 연결하기 위한 프로토콜 또는 방법을 지정합니다.

열 구조

`Table`의 열.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Column의 이름입니다.

- Type – Single-line string pattern과(와) 일치하는 131,072바이트 이하 길이의 UTF-8 문자열입니다.

  Column의 데이터 형식입니다.

- Comment – Single-line string pattern과(와) 일치하는 255바이트 이하 길이의 주석 문자열입니다.

  자유 형식의 텍스트 설명.

- Parameters – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 키-값 페어는 열과 관련된 속성을 정의합니다.

StorageDescriptor 구조

테이블 데이터의 물리적 스토리지를 설명합니다.

필드

- Columns – 열 객체의 배열입니다.

  테이블의 Columns 목록입니다.

- Location – URI address multi-line string pattern과(와) 일치하는 2,056바이트 이하 길이의 위치 문자열입니다.

  테이블의 물리적 위치입니다. 기본적으로 웨어하우스 위치, 웨어하우스의 데이터베이스 위치, 테이블 이름 순으로 나타납니다.

- AdditionalLocations – UTF-8 문자열의 배열입니다.

  Delta 테이블이 위치한 경로를 가리키는 위치 목록입니다.

- InputFormat – Single-line string pattern과(와) 일치하는 128바이트 이하 길이의 포맷 문자열입니다.

입력 형식: SequenceFileInputFormat(이진), TextInputFormat 또는 사용자 지정 형식입니다.

- OutputFormat – Single-line string pattern과(와) 일치하는 128바이트 이하 길이의 포맷 문자열입니다.

  출력 형식: SequenceFileOutputFormat(이진), IgnoreKeyTextOutputFormat 또는 사용자 지정 형식입니다.
- Compressed – 부울입니다.

  테이블의 데이터가 압축되면 True이고 그렇지 않으면 False입니다.
- NumberOfBuckets - 숫자(정수)입니다.

  테이블에 차원 열이 포함되어 있는 경우 이 속성을 지정해야 합니다.
- SerdeInfo – SerDeInfo 객체입니다.

  직렬화/역직렬화(SerDe) 정보입니다.
- BucketColumns – UTF-8 문자열의 배열입니다.

  테이블의 열, 클러스터링 열 및 버킷 열을 지정하는 그룹화하는 reducer 목록입니다.
- SortColumns – Order 객체의 배열입니다.

  테이블에 있는 각 버킷의 정렬 순서를 지정하는 목록입니다.
- Parameters – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  키 값 형식의 사용자 제공 속성입니다.
- SkewedInfo – SkewedInfo 객체입니다.

  열에 자주 표시되는 값에 대한 정보입니다(편향된 값).
- StoredAsSubDirectories – 부울입니다.

  테이블 데이터가 하위 디렉터리에 저장되면 True이고 그렇지 않으면 False입니다.
- SchemaReference – SchemaReference 객체입니다.

  AWS Glue Schema Registry에 저장된 스키마를 참조하는 객체입니다.

테이블을 생성할 때 스키마에 대한 빈 열 목록을 전달하고, 대신 스키마 참조를 사용할 수 있습니다.

SchemaReference 구조

AWS Glue Schema Registry에 저장된 스키마를 참조하는 객체입니다.

필드

- `SchemaId` – SchemaId 객체입니다.

  스키마 ID 필드를 포함하는 구조입니다. 이것 또는 `SchemaVersionId`가 제공되어야 합니다.
- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전에 할당된 고유 ID입니다. 이것 또는 `SchemaId`가 제공되어야 합니다.
- `SchemaVersionNumber` - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.

SerDeInfo 구조

추출기 및 로더 역할을 하는 직렬화/역직렬화 프로그램(SerDe)에 대한 정보입니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  SerDe의 이름입니다.
- `SerializationLibrary` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  일반적으로 SerDe를 구현하는 클래스입니다. 예를 들면, `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`입니다.
- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

이러한 키-값 쌍은 SerDe의 초기화 파라미터를 정의합니다.

Order 구조

분류된 열의 정렬 순서를 지정합니다.

필드

- `Column` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열의 이름입니다.
- `SortOrder` – 필수(Required): 1 이하의 숫자(정수)입니다.

  열이 오름차순(== 1) 또는 내림차순(==0)으로 정렬된 것을 나타냅니다.

SkewedInfo 구조

테이블에 왜곡된 값을 지정합니다. 왜곡된 값은 매우 높은 빈도를 통해 발생한 값입니다.

필드

- `SkewedColumnNames` – UTF-8 문자열의 배열입니다.

  왜곡된 값이 포함된 열의 이름 목록입니다.
- `SkewedColumnValues` – UTF-8 문자열의 배열입니다.

  너무 자주 나타나서 왜곡된 것으로 간주되는 값의 목록입니다.
- `SkewedColumnValueLocationMaps` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  왜곡된 값을 이 값을 포함하는 열에 매핑.

TableVersion 구조

테이블 버전을 지정합니다.

필드

- `Table` – 표 객체입니다.

  문제의 테이블입니다.

- `VersionId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 테이블 버전을 식별하는 ID 값. `VersionId`는 정수의 문자열 표현입니다. 각 버전은 1씩 증가합니다.

TableError 구조

테이블 작업의 오류 기록.

필드

- `TableName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름 반드시 모두 소문자로 저장하여 Hive 호환성을 유지하도록 합니다.

- `ErrorDetail` – ErrorDetail 객체입니다.

  오류에 대한 세부 정보입니다.

TableVersionError 구조

테이블 버전 작업의 오류 기록.

필드

- `TableName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 테이블 이름입니다.

- `VersionId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 버전 ID 값입니다. `VersionID`는 정수의 문자열 표현입니다. 각 버전은 1씩 증가합니다.

- `ErrorDetail` – ErrorDetail 객체입니다.

  오류에 대한 세부 정보입니다.

SortCriterion 구조

정렬 기준 필드와 정렬 순서를 지정합니다.

필드

- `FieldName` – 값 문자열입니다(1~1,024바이트).

  정렬할 필드의 이름입니다.

- `Sort` – UTF-8 문자열입니다(유효 값: `ASC="ASCENDING"` | `DESC="DESCENDING"`).

  오름차순 또는 내림차순 정렬.

TableIdentifier 구조

리소스 링크에 대한 대상 테이블을 설명하는 구조입니다.

필드

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 위치한 데이터 카탈로그의 ID입니다.

- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  대상 테이블을 포함하는 카탈로그 데이터베이스의 이름입니다.

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  대상 테이블의 이름입니다.

- `Region` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  대상 테이블의 리전입니다.

KeySchemaElement 구조

이름과 유형으로 구성된 파티션 키 페어입니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 키의 이름입니다.

- Type – 필수(Required): Single-line string pattern과(와) 일치하는 131,072바이트 이하 길이의 UTF-8 문자열입니다.

  파티션 키의 유형입니다.

PartitionIndex 구조

파티션 인덱스의 구조입니다.

필드

- Keys – 필수(Required): UTF-8 문자열의 배열이며 문자열은 1개 이상입니다.

  파티션 인덱스의 키입니다.

- IndexName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스의 이름입니다.

PartitionIndexDescriptor 구조

테이블의 파티션 인덱스에 대한 설명자입니다.

필드

- IndexName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스의 이름입니다.

- Keys – 필수(Required): KeySchemaElement 객체의 배열로 구조가 1개 이상입니다.

  파티션 인덱스에 대한 하나 이상의 키 목록(KeySchemaElement 구조)입니다.

- `IndexStatus` – 필수(Required): UTF-8 문자열입니다(유효 값: `CREATING` | `ACTIVE` | `DELETING` | `FAILED`).

  파티션 인덱스의 상태입니다.

  가능한 상태는 다음과 같습니다.
  - [생성 중(CREATING)]: 인덱스를 생성 중입니다. 인덱스가 [생성 중(CREATING)] 상태인 경우 인덱스 또는 해당 테이블을 삭제할 수 없습니다.
  - [활성(ACTIVE)]: 인덱스 생성에 성공했습니다.
  - [실패(FAILED)]: 인덱스 생성에 실패했습니다.
  - [삭제 중(DELETING)]: 인덱스 목록에서 인덱스를 삭제합니다.
- `BackfillErrors` – BackfillError 객체의 배열입니다.

  기존 테이블에 대한 파티션 인덱스를 등록할 때 발생할 수 있는 오류 목록입니다.

BackfillError 구조

기존 테이블에 대한 파티션 인덱스를 등록할 때 발생할 수 있는 오류 목록입니다.

이러한 오류는 인덱스 등록이 실패한 이유에 대한 세부 정보를 제공하고 응답에 제한된 수의 파티션을 제공하므로 결함이 있는 파티션을 수정하고 인덱스 등록을 다시 시도할 수 있습니다. 발생할 수 있는 가장 일반적인 오류 집합은 다음과 같이 분류됩니다.

- EncryptedPartitionError: 파티션이 암호화되었습니다.
- InvalidPartitionTypeDataError: 파티션 값이 해당 파티션 열의 데이터 유형과 일치하지 않습니다.
- MissingPartitionValueError: 파티션이 암호화되었습니다.
- UnsupportedPartitionCharacterError: 파티션 값 내의 문자는 지원되지 않습니다. 예: U+0000 , U+0001, U+0002.
- InternalError: 다른 오류 코드에 속하지 않는 모든 오류입니다.

필드

- `Code` – UTF-8 문자열입니다(유효한 값: `ENCRYPTED_PARTITION_ERROR` | `INTERNAL_ERROR` | `INVALID_PARTITION_TYPE_DATA_ERROR` | `MISSING_PARTITION_VALUE_ERROR` | `UNSUPPORTED_PARTITION_CHARACTER_ERROR`).

  기존 테이블에 대한 파티션 인덱스를 등록할 때 발생한 오류에 대한 오류 코드입니다.

- `Partitions` – PartitionValueList 객체의 배열입니다.

  응답의 제한된 수의 파티션 목록입니다.

IcebergInput 구조

카탈로그에서 생성할 Apache Iceberg 메타데이터 테이블을 정의하는 구조입니다.

필드

- `MetadataOperation` – 필수: UTF-8 문자열입니다(유효한 값: `CREATE`).

  필수 메타데이터 작업입니다. `CREATE`로만 설정할 수 있습니다.

- `Version` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Iceberg 테이블의 테이블 버전입니다. 기본값은 2입니다.

- `CreateIcebergTableInput` – CreateIcebergTableInput 객체입니다.

  테이블 속성 및 메타데이터 사양을 포함하여 AWS Glue Data Catalog에서 새 Iceberg 테이블을 생성하는 데 필요한 구성 파라미터입니다.

OpenTableFormatInput 구조

오픈 형식 테이블을 나타내는 구조입니다.

필드

- `IcebergInput` – IcebergInput 객체입니다.

  Apache Iceberg 메타데이터 테이블을 정의하는 `IcebergInput` 구조를 지정합니다.

ViewDefinition 구조

표현에 대한 세부 정보를 포함하는 구조입니다.

필드

- `IsProtected` – 부울입니다.

이 플래그를 true로 설정하면 쿼리 계획 중에 사용자가 제공한 작업을 뷰의 논리적 계획으로 푸시하지 않도록 엔진에 지시할 수 있습니다. 하지만 이 플래그를 설정한다고 해서 엔진이 지시를 따른다는 보장은 없습니다. 제공되는 보장에 대해 알아보려면 엔진 설명서를 참조하세요(있는 경우).

- `Definer` – Single-line string pattern과 일치하는 UTF-8 문자열입니다(20~2,048바이트).

  SQL에서 뷰의 정의자입니다.

- `SubObjects` – 10개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  Amazon 리소스 이름(ARN) 테이블 목록입니다.

- `Representations` – ViewRepresentation 객체의 배열이며 구조는 1~1,000개입니다.

  표현 목록입니다.

ViewDefinitionInput 구조

AWS Glue 뷰를 생성하거나 업데이트하기 위한 세부 정보를 포함하는 구조입니다.

필드

- `IsProtected` – 부울입니다.

  이 플래그를 true로 설정하면 쿼리 계획 중에 사용자가 제공한 작업을 뷰의 논리적 계획으로 푸시하지 않도록 엔진에 지시할 수 있습니다. 하지만 이 플래그를 설정한다고 해서 엔진이 지시를 따른다는 보장은 없습니다. 제공되는 보장에 대해 알아보려면 엔진 설명서를 참조하세요(있는 경우).

- `Definer` – Single-line string pattern과 일치하는 UTF-8 문자열입니다(20~2,048바이트).

  SQL에서 뷰의 정의자입니다.

- `Representations` – ViewRepresentationInput 객체의 배열이며 구조는 1~10개입니다.

  뷰의 언어와 뷰를 정의하는 쿼리를 포함하는 구조의 목록입니다.

- `SubObjects` – 10개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  뷰를 구성하는 기본 테이블 ARN의 목록입니다.

ViewRepresentation 구조

뷰의 언어와 뷰를 정의하는 쿼리를 포함하는 구조입니다.

필드

- `Dialect` – UTF-8 문자열입니다(유효한 값: `REDSHIFT | ATHENA | SPARK`).

  쿼리 엔진의 언어입니다.

- `DialectVersion` – 1~255바이트 길이의 UTF-8 문자열입니다.

  쿼리 엔진의 언어 버전입니다. 예를 들어 3.0.0입니다.

- `ViewOriginalText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  `CREATE VIEW` DDL 중에 고객이 제공한 `SELECT` 쿼리입니다. 이 SQL은 뷰에서 쿼리를 수행하는 동안에는 사용되지 않습니다(대신 `ViewExpandedText` 사용). `ViewOriginalText`는 사용자가 뷰를 생성한 원본 DDL 명령을 보려고 하는 `SHOW CREATE VIEW`와 같은 경우에 사용됩니다.

- `ViewExpandedText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  뷰를 위한 확장된 SQL입니다. 이 SQL은 엔진이 뷰에서 쿼리를 처리하는 동안 사용됩니다. 엔진은 뷰 생성 중에 `ViewOriginalText`를 `ViewExpandedText`로 변환하기 위해 작업을 수행할 수 있습니다. 예시:

  - 정규화된 식별자: `SELECT * from table1 -> SELECT * from db1.table1`

- `ValidationConnection` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  뷰의 특정 표현을 검증하는 데 사용할 연결의 이름입니다.

- `IsStale` – 부울입니다.

  기한 경과로 표시된 언어는 더 이상 유효하지 않으므로 해당 쿼리 엔진에서 쿼리하려면 먼저 업데이트해야 합니다.

ViewRepresentationInput 구조

Lake Formation 뷰를 업데이트하거나 생성하기 위한 표현의 세부 정보를 포함하는 구조입니다.

필드

- `Dialect` – UTF-8 문자열입니다(유효한 값: `REDSHIFT | ATHENA | SPARK`).

  특정 표현의 엔진 유형을 지정하는 파라미터입니다.

- `DialectVersion` – 1~255바이트 길이의 UTF-8 문자열입니다.

특정 표현의 엔진 버전을 지정하는 파라미터입니다.

- `ViewOriginalText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  뷰를 설명하는 원본 SQL 쿼리를 나타내는 문자열입니다.

- `ValidationConnection` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  뷰의 특정 표현을 검증하는 데 사용할 연결의 이름입니다.

- `ViewExpandedText` – 409,600바이트 이하 길이의 UTF-8 문자열입니다.

  확장된 리소스 ARN으로 뷰를 설명하는 SQL 쿼리를 나타내는 문자열

UpdateOpenTableFormatInput 구조

AWS Glue Data Catalog에서 열린 테이블 형식 테이블을 업데이트하기 위한 입력 파라미터로, Apache Iceberg와 같은 형식별 업데이트 작업의 래퍼 역할을 합니다.

필드

- `UpdateIcebergInput` – UpdateIcebergInput 객체입니다.

  스키마 변경, 파티션 사양 및 테이블 속성을 포함하여 적용할 테이블 수정을 정의하는 Apache Iceberg별 업데이트 파라미터입니다.

UpdateIcebergInput 구조

AWS Glue Data Catalog의 Apache Iceberg 테이블 업데이트와 관련된 입력 파라미터로, 기존 Iceberg 테이블에 적용할 업데이트 작업을 포함합니다.

필드

- `UpdateIcebergTableInput` – 필수(Required): UpdateIcebergTableInput 객체입니다.

  Iceberg 테이블에 적용할 특정 업데이트 작업으로, 스키마, 파티션 및 속성을 포함하여 테이블의 새 상태를 정의하는 업데이트 목록을 포함합니다.

CreateIcebergTableInput 구조

테이블 속성 및 메타데이터 사양을 포함하여 AWS Glue Data Catalog에서 새 Iceberg 테이블을 생성하는 데 필요한 구성 파라미터입니다.

필드

- `Location` - 필수: URI address multi-line string pattern과 일치하는 2,056바이트 이하 길이의 위치 문자열입니다.

  Iceberg 테이블 데이터가 저장되는 S3 위치입니다.

- `Schema` – 필수(Required): IcebergSchema 객체입니다.

  Iceberg 테이블의 구조, 필드 유형 및 메타데이터를 지정하는 스키마 정의입니다.

- `PartitionSpec` – IcebergPartitionSpec 객체입니다.

  최적의 쿼리 성능을 위해 Iceberg 테이블 데이터가 구성 및 파티셔닝되는 방법을 정의하는 파티셔닝 사양입니다.

- `WriteOrder` – IcebergSortOrder 객체입니다.

  쿼리 성능을 최적화하기 위해 각 파티션 내에서 데이터를 정렬하는 방법을 정의하는 정렬 순서 사양입니다.

- `Properties` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  Iceberg 테이블의 추가 테이블 속성 및 구성 설정의 키-값 페어입니다.

UpdateIcebergTableInput 구조

AWS Glue Data Catalog의 기존 Iceberg 테이블에 적용할 업데이트 작업을 포함하고, 테이블 메타데이터의 새 상태를 정의합니다.

필드

- `Updates` – 필수(Required): IcebergTableUpdate 객체의 배열입니다.

Iceberg 테이블에 적용할 변경 사항을 지정하는 테이블 업데이트 작업 목록으로, 스키마 수정, 파티션 사양 및 테이블 속성을 포함합니다.

IcebergSortOrder 구조

Iceberg 테이블의 정렬 순서 사양을 정의하여 쿼리 성능 최적화를 위해 파티션 내에서 데이터를 정렬하는 방법을 결정합니다.

필드

- `OrderId` – 필수(Required): 숫자(정수)입니다.

  Iceberg 테이블의 메타데이터 내에서 이 정렬 순서 사양의 고유 식별자입니다.
- `Fields` – 필수(Required): IcebergSortField 객체의 배열입니다.

  Iceberg 테이블 데이터의 순서 기준을 정의하는 필드 목록 및 정렬 방향입니다.

IcebergSortField 구조

소스 필드, 변환, 정렬 방향 및 null 값 순서를 포함하여 Iceberg 정렬 순서 사양 내에서 단일 필드를 정의합니다.

필드

- `SourceId` – 필수(Required): 숫자(정수)입니다.

  이 정렬 필드의 기반이 되는 테이블 스키마의 소스 필드 식별자입니다.
- `Transform` – 필수: UTF-8 문자열입니다.

  정렬 전에 소스 필드에 적용되는 변환 함수(예: identity, bucket, truncate)입니다.
- `Direction` – 필수: UTF-8 문자열입니다(유효한 값: `asc="ASC"` | `desc="DESC"`).

  이 필드의 정렬 방향(오름차순 또는 내림차순)입니다.
- `NullOrder` – 필수: UTF-8 문자열입니다(유효한 값: `nulls-first="NULLS_FIRST"` | `nulls-last="NULLS_LAST"`).

  이 필드의 null 값에 대한 순서 지정 동작으로, null이 정렬 순서에서 첫 번째 또는 마지막에 나타나야 하는지 여부를 지정합니다.

IcebergPartitionSpec 구조

Iceberg 테이블의 파티셔닝 사양을 정의하여 최적의 쿼리 성능을 위해 테이블 데이터가 구성 및 파티셔닝되는 방법을 결정합니다.

필드

- `Fields` – 필수(Required): IcebergPartitionField 객체의 배열입니다.

  소스 필드 및 변환을 포함하여 테이블 데이터가 파티셔닝되는 방법을 정의하는 파티션 필드 목록입니다.

- `SpecId` - 숫자(정수)입니다.

  Iceberg 테이블의 메타데이터 기록 내에서 이 파티션 사양의 고유 식별자입니다.

IcebergPartitionField 구조

소스 필드, 변환 함수, 파티션 이름 및 고유 식별자를 포함하여 Iceberg 파티션 사양 내에서 단일 파티션 필드를 정의합니다.

필드

- `SourceId` – 필수(Required): 숫자(정수)입니다.

  이 파티션 필드의 기반이 되는 테이블 스키마의 소스 필드 식별자입니다.

- `Transform` – 필수: UTF-8 문자열입니다.

  파티션을 생성하기 위해 소스 필드에 적용되는 변환 함수(예: identity, bucket, truncate, year, month, day, hour)입니다.

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~1,024바이트 길이의 UTF-8 문자열입니다.

  파티셔닝된 테이블 구조에 표시될 파티션 필드의 이름입니다.

- `FieldId` - 숫자(정수)입니다.

  Iceberg 테이블의 파티션 사양 내에서 이 파티션 필드에 할당된 고유 식별자입니다.

IcebergSchema 구조

필드 정의, 데이터 유형 및 스키마 메타데이터를 포함하여 Iceberg 테이블의 스키마 구조를 정의합니다.

필드

- `SchemaId` - 숫자(정수)입니다.

  Iceberg 테이블의 스키마 진화 기록 내에 있는 이 스키마 버전의 고유 식별자입니다.
- `IdentifierFieldIds` - 서명된 32비트 정수의 배열입니다.

  테이블의 레코드를 고유하게 식별하는 필드 식별자 목록으로, 행 수준 작업 및 중복 제거에 사용됩니다.
- `Type` – UTF-8 문자열입니다(유효한 값: `struct="STRUCT"`).

  스키마 구조의 루트 유형으로, 일반적으로 Iceberg 테이블 스키마의 경우 'struct'입니다.
- `Fields` – 필수(Required): IcebergStructField 객체의 배열입니다.

  필드 이름, 유형 및 메타데이터를 포함하여 테이블 스키마를 구성하는 필드 정의 목록입니다.

IcebergStructField 구조

식별자, 이름, 데이터 유형, null 허용 여부 및 문서화를 포함하여 Iceberg 테이블 스키마 내의 단일 필드를 정의합니다.

필드

- `Id` – 필수(Required): 숫자(정수)입니다.

  Iceberg 테이블 스키마 내에서 이 필드에 할당된 고유 식별자로, 스키마 진화 및 필드 추적에 사용됩니다.
- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~1,024바이트 길이의 UTF-8 문자열입니다.

  테이블 스키마 및 쿼리 작업에 표시되는 필드의 이름입니다.
- `Type` - 필수: 이름이 `IcebergDocument`인 빈 구조입니다.

  이 필드에 대한 데이터 유형 정의로, 포함된 데이터의 구조와 형식을 지정합니다.

- `Required` – 필수(Required): 부울.

  테이블 스키마에 이 필드가 필요한지(null 비허용) 또는 선택 사항인지(null 허용) 여부를 나타냅니다.

- `Doc` – Single-line string pattern과(와) 일치하는 255바이트 이하 길이의 주석 문자열입니다.

  이 필드의 용도 및 사용에 대한 추가 컨텍스트를 제공하는 선택적 문서화 또는 설명 텍스트입니다.

IcebergTableUpdate 구조

스키마 변경, 파티셔닝 수정, 정렬 순서 조정, 위치 업데이트, 속성 변경 등 Iceberg 테이블에 적용할 전체 업데이트 세트를 정의합니다.

필드

- `Schema` – 필수(Required): IcebergSchema 객체입니다.

  필드 구조, 데이터 유형 또는 스키마 메타데이터에 대한 변경 사항을 지정하여 Iceberg 테이블에 대해 업데이트된 스키마 정의입니다.

- `PartitionSpec` – IcebergPartitionSpec 객체입니다.

  테이블 데이터가 재구성 및 파티셔닝되는 방법을 정의하는 업데이트된 파티셔닝 사양입니다.

- `SortOrder` – IcebergSortOrder 객체입니다.

  최적의 쿼리 성능을 위해 파티션 내에서 데이터를 정렬하는 방법을 정의하는 업데이트된 정렬 순서 사양입니다.

- `Location` - 필수: URI address multi-line string pattern과 일치하는 2,056바이트 이하 길이의 위치 문자열입니다.

  Iceberg 테이블 데이터가 저장되는 업데이트된 S3 위치입니다.

- `Properties` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  Iceberg 테이블의 테이블 속성 및 구성 설정의 업데이트된 키-값 페어입니다.

운영

- CreateTable 작업(Python: create_table)
- UpdateTable 작업(Python: update_table)
- DeleteTable 작업(Python: delete_table)
- BatchDeleteTable 작업(Python: batch_delete_table)
- GetTable 작업(Python: get_table)
- GetTables 작업(Python: get_tables)
- GetTableVersion 작업(Python: get_table_version)
- GetTableVersions 작업(Python: get_table_versions)
- DeleteTableVersion 작업(Python: delete_table_version)
- BatchDeleteTableVersion 작업(Python: batch_delete_table_version)
- SearchTables 작업(Python: search_tables)
- GetPartitionIndexes 작업(Python: get_partition_indexes)
- CreatePartitionIndex 작업(Python: create_partition_index)
- DeletePartitionIndex 작업(Python: delete_partition_index)
- GetColumnStatisticsForTable 작업(Python: get_column_statistics_for_table)
- UpdateColumnStatisticsForTable 작업(Python: update_column_statistics_for_table)
- DeleteColumnStatisticsForTable 작업(Python: delete_column_statistics_for_table)

CreateTable 작업(Python: create_table)

데이터 카탈로그에서 새로운 테이블 정의를 생성합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  `Table`을 생성할 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

새로운 테이블을 생성할 카탈로그 데이터베이스입니다. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue Data Catalog에 생성될 지정된 데이터베이스 내 테이블의 고유 식별자입니다.

- TableInput – 필수(Required): TableInput 객체입니다.

  TableInput 객체는 카탈로그에 생성할 메타데이터 테이블을 정의합니다.

- PartitionIndexes – PartitionIndex 객체의 배열이며 구조는 3개 이하입니다.

  테이블에 생성할 파티션 인덱스(PartitionIndex 구조)의 목록입니다.

- TransactionId – Custom string pattern #43과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  트랜잭션의 ID입니다.

- OpenTableFormatInput – OpenTableFormatInput 객체입니다.

  오픈 형식 테이블을 생성하는 경우 OpenTableFormatInput 구조를 지정합니다.

응답

- 무응답 파라미터.

오류

- AlreadyExistsException
- InvalidInputException
- EntityNotFoundException
- ResourceNumberLimitExceededException
- InternalServiceException
- OperationTimeoutException
- GlueEncryptionException
- ConcurrentModificationException
- ResourceNotReadyException

- FederationSourceException
- FederationSourceRetryableException

UpdateTable 작업(Python: update_table)

데이터 카탈로그에서 메타데이터 테이블을 업데이트합니다.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- DatabaseName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터 데이터베이스의 이름입니다. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.
- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue Data Catalog에 생성될 지정된 데이터베이스 내 테이블의 고유 식별자입니다.
- TableInput – TableInput 객체입니다.

  업데이트된 TableInput 객체는 카탈로그에 생성할 메타데이터 테이블을 정의합니다.
- SkipArchive – 부울입니다.

  기본적으로 UpdateTable는 항상 업데이트하기 전에 테이블 보관 버전을 생성합니다. 그러나 skipArchive이 true이면 UpdateTable는 보관된 버전을 생성하지 않습니다.
- TransactionId – Custom string pattern #43과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 콘텐츠를 업데이트할 트랜잭션 ID입니다.
- VersionId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 콘텐츠를 업데이트할 버전 ID입니다.
- ViewUpdateAction – UTF-8 문자열입니다(유효한 값: ADD | REPLACE | ADD_OR_REPLACE | DROP).

뷰를 업데이트할 때 수행할 작업입니다.

- `Force` – 부울입니다.

  이 플래그를 true로 설정하면 일치하는 스토리지 서술자와 하위 객체 매칭 요구 사항을 무시하도록 지정할 수 있습니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`
- `AlreadyExistsException`

DeleteTable 작업(Python: delete_table)

데이터 카탈로그에서 테이블 정의를 제거합니다.

> **Note**
>
> 이 작업을 완료하면 삭제된 테이블에 속한 테이블 버전 및 파티션에 더 이상 액세스할 수 없습니다. AWS Glue는 이러한 "분리된" 리소스를 서비스 재량에 따라 적시에 비동기로 삭제합니다.

관련된 모든 리소스가 즉시 삭제되도록 DeleteTable 호출 전에 DeleteTableVersion 또는 BatchDeleteTableVersion과 DeletePartition 또는 BatchDeletePartition을 사용하여 테이블에 속한 모든 리소스를 삭제하십시오.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- DatabaseName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터 데이터베이스의 이름입니다. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.
- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제된 테이블 이름. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.
- TransactionId – Custom string pattern #43과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 콘텐츠를 삭제할 트랜잭션 ID입니다.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- ConcurrentModificationException

- ResourceNotReadyException
- FederationSourceException
- FederationSourceRetryableException

BatchDeleteTable 작업(Python: batch_delete_table)

한 번에 여러 테이블 삭제

> **Note**
>
> 이 작업을 완료하면 삭제된 테이블에 속한 테이블 버전 및 파티션에 더 이상 액세스할 수 없습니다. AWS Glue는 이러한 "분리된" 리소스를 서비스 재량에 따라 적시에 비동기로 삭제합니다.
> 관련된 모든 리소스가 즉시 삭제되도록 BatchDeleteTable 호출 전에 DeleteTableVersion 또는 BatchDeleteTableVersion과 DeletePartition 또는 BatchDeletePartition을 사용하여 테이블에 속한 모든 리소스를 삭제하십시오.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- DatabaseName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 테이블이 있는 카탈로그 데이터베이스의 이름입니다. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- TablesToDelete – 필수(Required): 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  삭제할 테이블 목록.

- TransactionId – Custom string pattern #43과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 콘텐츠를 삭제할 트랜잭션 ID입니다.

응답

- `Errors` – TableError 객체의 배열입니다.

  지정된 테이블을 삭제하는 중 발생한 오류 목록입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`

GetTable 작업(Python: get_table)

지정된 테이블의 데이터 카탈로그에서 `Table`정의를 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스 이름입니다. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  정의를 가져오는 테이블의 이름입니다. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `TransactionId` – Custom string pattern #43과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 콘텐츠를 읽을 트랜잭션 ID입니다.

- `QueryAsOfTime` – 타임스탬프입니다.

  테이블 콘텐츠를 읽을 기준 시간입니다. 설정하지 않으면 가장 최근의 트랜잭션 커밋 시간이 사용됩니다. `TransactionId`과(와) 함께 지정할 수 없습니다.

- `IncludeStatusDetails` – 부울입니다.

  AWS Glue 데이터 카탈로그 보기를 생성하거나 업데이트하기 위한 요청과 관련된 상태 세부 정보를 포함할지 여부를 지정합니다.

응답

- `Table` – 표 객체입니다.

  지정한 테이블을 정의하는 `Table` 객체.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

GetTables 작업(Python: get_tables)

주어진 `Database`의 테이블 중 몇 개 혹은 모두의 정의를 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기록할 테이블이 있는 카탈로그의 데이터베이스입니다. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `Expression` – Single-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  정규 표현식 패턴 테이블이 있으면 이름이 패턴과 일치하는 테이블만 반환됩니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰을 포함합니다.

- `MaxResults` – 1~100의 숫자(정수)입니다.

  한 번의 응답으로 반환할 최대 테이블 수.

- `TransactionId` – Custom string pattern #43과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 콘텐츠를 읽을 트랜잭션 ID입니다.

- `QueryAsOfTime` – 타임스탬프입니다.

  테이블 콘텐츠를 읽을 기준 시간입니다. 설정하지 않으면 가장 최근의 트랜잭션 커밋 시간이 사용됩니다. `TransactionId`과(와) 함께 지정할 수 없습니다.

- `IncludeStatusDetails` – 부울입니다.

  AWS Glue 데이터 카탈로그 보기를 생성하거나 업데이트하기 위한 요청과 관련된 상태 세부 정보를 포함할지 여부를 지정합니다.

- `AttributesToGet` – UTF-8 문자열의 배열입니다.

  `GetTables` 호출에서 반환된 테이블 필드를 지정합니다. 이 파라미터는 빈 목록을 허용하지 않습니다. 요청에는 `NAME`이(가) 포함되어야 합니다.

유효한 값 조합은 다음과 같습니다.

- NAME - 데이터베이스의 모든 테이블 이름.
- NAME, TABLE_TYPE - 모든 테이블의 이름 및 테이블 유형.

응답

- TableList – 표 객체의 배열입니다.

  요청한 Table 객체의 목록입니다.
- NextToken – UTF-8 문자열입니다.

  현재 목록 부분이 유지가 되지 않으면 연속 토큰이 존재합니다.

오류

- EntityNotFoundException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException
- GlueEncryptionException
- FederationSourceException
- FederationSourceRetryableException

GetTableVersion 작업(Python: get_table_version)

테이블의 지정된 버전을 가져옵니다.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `VersionId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  가져올 테이블 버전의 ID값입니다. `VersionID`는 정수의 문자열 표현입니다. 각 버전은 1씩 증가합니다.

응답

- `TableVersion` – TableVersion 객체입니다.

  요청한 테이블 버전

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

GetTableVersions 작업(Python: get_table_versions)

문자열 목록을 가져와 지정된 테이블의 사용 가능한 버전을 식별합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 첫 번째 호출이 아니면 지속적인 토큰입니다.

- `MaxResults` – 1~100의 숫자(정수)입니다.

  한 번의 응답으로 반환될 최대 테이블 버전 수입니다.

응답

- `TableVersions` – TableVersion 객체의 배열입니다.

  문자열 목록은 지정된 테이블의 사용 가능한 버전을 식별합니다.

- `NextToken` – UTF-8 문자열입니다.

  사용 가능한 버전 목록이 마지막 버전을 포함하지 않은 경우의 연속 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeleteTableVersion 작업(Python: delete_table_version)

테이블의 지정된 버전을 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `VersionId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제될 테이블 버전의 ID입니다. `VersionID`는 정수의 문자열 표현입니다. 각 버전은 1씩 증가합니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchDeleteTableVersion 작업(Python: batch_delete_table_version)

테이블 버전의 지정된 배치를 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스. 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름 이름은 모두 소문자로 되어 있어야 Hive 호환성을 유지합니다.

- `VersionIds` – 필수(Required): 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  삭제될 ID 버전 목록입니다. `VersionId`는 정수의 문자열 표현입니다. 각 버전은 1씩 증가합니다.

응답

- `Errors` – TableVersionError 객체의 배열입니다.

  지정된 테이블 버전을 삭제하고자 할 때 발생한 오류 목록입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

SearchTables 작업(Python: search_tables)

상위 데이터베이스 뿐만 아니라 테이블 메타데이터의 속성을 기반으로 테이블 집합을 검색합니다. 텍스트 또는 필터 조건으로 검색할 수 있습니다.

Lake Formation에 정의된 보안 정책을 기반으로 액세스할 수 있는 테이블만 가져올 수 있습니다. 테이블이 반환되기 위해서는 최소한 읽기 전용 액세스 권한이 필요합니다. 테이블의 모든 열에 액세스할 수 없으면, 테이블 목록이 다시 반환될 때 이러한 열이 검색되지 않습니다. 열에 액세스할 수 있지만 열의 데이터에는 액세스할 수 없는 경우 해당 열 및 해당 열의 관련 메타데이터가 검색에 포함됩니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  *account_id*로 구성된 고유한 식별자입니다.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰을 포함합니다.
- `Filters` – PropertyPredicate 객체의 배열입니다.

  키-값 페어의 목록 및 검색 결과를 필터링하는 데 사용되는 비교기입니다. 조건자와 일치하는 모든 엔터티를 반환합니다.

  `PropertyPredicate` 구조체의 `Comparator` 멤버는 시간 필드에만 사용되며 다른 필드 유형에서는 생략할 수 있습니다. 또한 `Key=Name`과 같이 문자열 값을 비교할 때 퍼지 일치 알고리즘을 사용합니다. `Key` 필드(예: `Name` 필드 값)는 특정 구두점 문자(예: -, :, # 등)에서 토큰으로 분할됩니다. 그런 다음 각 토큰은 `PropertyPredicate`의 `Value` 멤버와 정확히 일치합니다. 예를 들어 `Key=Name`과 `Value=link`가 있으면 `customer-link` 및 `xx-link-yy`라는 테이블은 반환되지만 `xxlinkyy`는 반환되지 않는다.
- `SearchText` – 값 문자열입니다(1~1,024바이트).

  텍스트 검색에 사용되는 문자열입니다.

  인용 부호로 값을 지정하면, 정확도를 기준으로 필터링합니다.
- `SortCriteria` – SortCriterion 객체의 배열이며 구조는 1개 이하입니다.

  필드 이름을 기준으로 결과를 오름차순 또는 내림차순으로 정렬하는 기준 목록입니다.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

한 번의 응답으로 반환할 최대 테이블 수.

- `ResourceShareType` – UTF-8 문자열입니다(유효한 값: `FOREIGN` | `ALL` | `FEDERATED`).

  계정과 공유된 테이블을 검색하도록 지정할 수 있습니다. 허용 가능 값은 `FOREIGN` 또는 `ALL`입니다.

  - `FOREIGN`으로 설정하면 계정과 공유된 테이블이 검색됩니다.
  - `ALL`로 설정하면 계정과 공유된 테이블과 로컬 계정의 테이블이 검색됩니다.
- `IncludeStatusDetails` – 부울입니다.

  AWS Glue 데이터 카탈로그 보기를 생성하거나 업데이트하기 위한 요청과 관련된 상태 세부 정보를 포함할지 여부를 지정합니다.

응답

- `NextToken` – UTF-8 문자열입니다.

  현재 목록 부분이 유지가 되지 않으면 연속 토큰이 존재합니다.
- `TableList` – 표 객체의 배열입니다.

  요청한 `Table` 객체의 목록입니다. `SearchTables` 응답은 액세스할 수 있는 테이블만 반환합니다.

오류

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

GetPartitionIndexes 작업(Python: get_partition_indexes)

테이블과 연결된 파티션 인덱스를 검색합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 있는 카탈로그 ID입니다.

- DatabaseName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스를 검색하려는 데이터베이스의 이름을 지정합니다.

- TableName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스를 검색할 테이블의 이름을 지정합니다.

- NextToken – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰을 포함합니다.

응답

- PartitionIndexDescriptorList – PartitionIndexDescriptor 객체의 배열입니다.

  인덱스 설명자의 목록입니다.

- NextToken – UTF-8 문자열입니다.

  현재 목록 부분이 유지가 되지 않으면 연속 토큰이 존재합니다.

오류

- InternalServiceException
- OperationTimeoutException
- InvalidInputException
- EntityNotFoundException
- ConflictException

CreatePartitionIndex 작업(Python: create_partition_index)

기존 테이블에 지정된 파티션 인덱스를 생성합니다.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

테이블이 있는 카탈로그 ID입니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스를 생성하려는 데이터베이스의 이름을 지정합니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스를 생성하려는 테이블의 이름을 지정합니다.

- `PartitionIndex` – 필수(Required): PartitionIndex 객체입니다.

  `PartitionIndex` 구조를 지정하여 기존 테이블에 파티션 인덱스를 생성합니다.

응답

- 무응답 파라미터.

오류

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeletePartitionIndex 작업(Python: delete_partition_index)

기존 테이블에서 지정된 파티션 인덱스를 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

테이블이 있는 카탈로그 ID입니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스를 삭제하려는 데이터베이스의 이름을 지정합니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 인덱스를 삭제하려는 테이블의 이름을 지정합니다.

- `IndexName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 파티션 인덱스의 이름입니다.

응답

- 무응답 파라미터.

오류

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ConflictException`
- `GlueEncryptionException`

GetColumnStatisticsForTable 작업(Python: get_column_statistics_for_table)

열의 테이블 통계를 검색합니다.

이 작업에 필요한 Identity and Access Management(IAM) 권한은 `GetTable`입니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.

- `ColumnNames` – 필수(Required): 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  열 이름의 목록입니다.

응답

- `ColumnStatisticsList` – ColumnStatistics 객체의 배열입니다.

  ColumnStatistics 목록입니다.

- `Errors` – ColumnError 객체의 배열입니다.

  검색에 실패한 ColumnStatistics 목록입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

UpdateColumnStatisticsForTable 작업(Python: update_column_statistics_for_table)

열의 테이블 통계를 생성하거나 업데이트합니다.

이 작업에 필요한 Identity and Access Management(IAM) 권한은 `UpdateTable`입니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.

- `ColumnStatisticsList` – 필수(Required): ColumnStatistics 객체의 배열이며 구조는 25개 이하입니다.

  열 통계의 목록입니다.

응답

- `Errors` – ColumnStatisticsError 객체의 배열입니다.

  ColumnStatisticsErrors의 목록입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

- GlueEncryptionException

DeleteColumnStatisticsForTable 작업(Python: delete_column_statistics_for_table)

열의 테이블 통계를 검색합니다.

이 작업에 필요한 Identity and Access Management(IAM) 권한은 DeleteTable입니다.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- DatabaseName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.
- TableName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.
- ColumnName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열의 이름입니다.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException
- InternalServiceException
- OperationTimeoutException

- `GlueEncryptionException`

파티션 API

파티션 API는 파티션을 실행할 데이터 형식 및 작업에 대해 설명합니다.

데이터 타입

- 파티션 구조
- PartitionInput 구조
- PartitionSpecWithSharedStorageDescriptor 구조
- PartitionListComposingSpec 구조
- PartitionSpecProxy 구조
- PartitionValueList 구조
- 세그먼트 구조
- PartitionError 구조
- BatchUpdatePartitionFailureEntry 구조
- BatchUpdatePartitionRequestEntry 구조
- StorageDescriptor 구조
- SchemaReference 구조
- SerDeInfo 구조
- SkewedInfo 구조

파티션 구조

테이블 데이터의 조각을 나타냅니다.

필드

- `Values` – UTF-8 문자열의 배열입니다.

  파티션 값입니다.
- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

파티션이 생성되는 카탈로그 데이터베이스의 이름입니다.

- `TableName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 생성되는 데이터베이스 테이블의 이름입니다.

- `CreationTime` – 타임스탬프입니다.

  파티션이 생성된 시간.

- `LastAccessTime` – 타임스탬프입니다.

  파티션이 액세스된 마지막 시간.

- `StorageDescriptor` – StorageDescriptor 객체입니다.

  파티션이 저장된 물리적 위치에 대한 정보를 제공합니다.

- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 키-값 쌍은 파티션 파라미터를 정의합니다.

- `LastAnalyzedTime` – 타임스탬프입니다.

  이 파티션을 위해 계산된 열 통계 마지막 시간.

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  파티션이 있는 Data Catalog의 ID입니다.

PartitionInput 구조

파티션을 생성 및 업데이트할 때 사용되는 구조입니다.

필드

- `Values` – UTF-8 문자열의 배열입니다.

  파티션 값입니다. 이 파라미터는 SDK에 필수는 아니지만 유효한 입력을 위해 이 파라미터를 지정해야 합니다.

새 파티션의 키 값은 Amazon S3 접두사에 표시되는 파티션 키와 동일한 순서로 정렬되어야 하는 문자열 객체의 배열로 전달되어야 합니다. 그렇지 않으면 AWS Glue가 잘못된 키에 값을 추가합니다.

- `LastAccessTime` – 타임스탬프입니다.

  파티션이 액세스된 마지막 시간.

- `StorageDescriptor` – StorageDescriptor 객체입니다.

  파티션이 저장된 물리적 위치에 대한 정보를 제공합니다.

- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 키-값 쌍은 파티션 파라미터를 정의합니다.

- `LastAnalyzedTime` – 타임스탬프입니다.

  이 파티션을 위해 계산된 열 통계 마지막 시간.

PartitionSpecWithSharedStorageDescriptor 구조

물리적 위치를 공유하는 파티션의 파티션 스펙.

필드

- `StorageDescriptor` – StorageDescriptor 객체입니다.

  공유된 물리적 스토리지 정보.

- `Partitions` – Partition 객체의 배열입니다.

  이 물리적 위치를 공유하는 파티션 목록.

PartitionListComposingSpec 구조

관련된 파티션을 나열합니다.

필드

- `Partitions` – Partition 객체의 배열입니다.

구성 스펙의 파티션 목록.

PartitionSpecProxy 구조

지정된 파티션으로의 루트 경로를 제공합니다.

필드

- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스입니다.
- `TableName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 포함된 테이블의 이름입니다.
- `RootPath` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 설명되어 있는 프록시 루트 경로.
- `PartitionSpecWithSharedSD` – PartitionSpecWithSharedStorageDescriptor 객체입니다.

  동일한 스토리지 위치를 공유하는 파티션 스펙.
- `PartitionListComposingSpec` – PartitionListComposingSpec 객체입니다.

  파티션 목록을 지정합니다.

PartitionValueList 구조

파티션을 정의하는 값 목록을 포함합니다.

필드

- `Values` – 필수(Required): UTF-8 문자열의 배열입니다.

  값 목록.

세그먼트 구조

테이블 파티션의 중첩되지 않은 리전을 정의하여 다양한 요청이 병렬로 실행되도록 허용합니다.

필드

- `SegmentNumber` – 필수(Required): None 이하의 숫자(정수)입니다.

  세그먼트의 0 기반 인덱스 숫자입니다. 예를 들어 총 세그먼트 수가 4개인 경우, `SegmentNumber` 값의 범위는 0~3입니다.
- `TotalSegments` – 필수(Required): 1~10의 숫자(정수)입니다.

  총 세그먼트 수입니다.

PartitionError 구조

파티션 오류에 관한 정보를 포함합니다.

필드

- `PartitionValues` – UTF-8 문자열의 배열입니다.

  파티션을 정의하는 값입니다.
- `ErrorDetail` – ErrorDetail 객체입니다.

  파티션 오류에 대한 세부 정보입니다.

BatchUpdatePartitionFailureEntry 구조

배치 업데이트 파티션 오류에 관한 정보를 포함합니다.

필드

- `PartitionValueList` – 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  파티션을 정의하는 값의 목록입니다.
- `ErrorDetail` – ErrorDetail 객체입니다.

  배치 업데이트 파티션 오류에 대한 세부 정보입니다.

BatchUpdatePartitionRequestEntry 구조

파티션을 업데이트하는 데 사용되는 값과 구조를 포함하는 구조입니다.

필드

- `PartitionValueList` – 필수(Required): 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  파티션을 정의하는 값의 목록입니다.
- `PartitionInput` – 필수(Required): PartitionInput 객체입니다.

  파티션을 업데이트할 때 사용되는 구조입니다.

StorageDescriptor 구조

테이블 데이터의 물리적 스토리지를 설명합니다.

필드

- `Columns` – 열 객체의 배열입니다.

  테이블의 `Columns` 목록입니다.
- `Location` – URI address multi-line string pattern과(와) 일치하는 2,056바이트 이하 길이의 위치 문자열입니다.

  테이블의 물리적 위치입니다. 기본적으로 웨어하우스 위치, 웨어하우스의 데이터베이스 위치, 테이블 이름 순으로 나타납니다.
- `AdditionalLocations` – UTF-8 문자열의 배열입니다.

  Delta 테이블이 위치한 경로를 가리키는 위치 목록입니다.
- `InputFormat` – Single-line string pattern과(와) 일치하는 128바이트 이하 길이의 포맷 문자열입니다.

  입력 형식: `SequenceFileInputFormat`(이진), `TextInputFormat` 또는 사용자 지정 형식입니다.
- `OutputFormat` – Single-line string pattern과(와) 일치하는 128바이트 이하 길이의 포맷 문자열입니다.

  출력 형식: `SequenceFileOutputFormat`(이진), `IgnoreKeyTextOutputFormat` 또는 사용자 지정 형식입니다.
- `Compressed` – 부울입니다.

  테이블의 데이터가 압축되면 `True`이고 그렇지 않으면 `False`입니다.

- `NumberOfBuckets` - 숫자(정수)입니다.

  테이블에 차원 열이 포함되어 있는 경우 이 속성을 지정해야 합니다.
- `SerdeInfo` – SerDeInfo 객체입니다.

  직렬화/역직렬화(SerDe) 정보입니다.
- `BucketColumns` – UTF-8 문자열의 배열입니다.

  테이블의 열, 클러스터링 열 및 버킷 열을 지정하는 그룹화하는 reducer 목록입니다.
- `SortColumns` – Order 객체의 배열입니다.

  테이블에 있는 각 버킷의 정렬 순서를 지정하는 목록입니다.
- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  키 값 형식의 사용자 제공 속성입니다.
- `SkewedInfo` – SkewedInfo 객체입니다.

  열에 자주 표시되는 값에 대한 정보입니다(편향된 값).
- `StoredAsSubDirectories` – 부울입니다.

  테이블 데이터가 하위 디렉터리에 저장되면 `True`이고 그렇지 않으면 `False`입니다.
- `SchemaReference` – SchemaReference 객체입니다.

  AWS Glue Schema Registry에 저장된 스키마를 참조하는 객체입니다.

  테이블을 생성할 때 스키마에 대한 빈 열 목록을 전달하고, 대신 스키마 참조를 사용할 수 있습니다.

SchemaReference 구조

AWS Glue Schema Registry에 저장된 스키마를 참조하는 객체입니다.

필드

- `SchemaId` – SchemaId 객체입니다.

  스키마 ID 필드를 포함하는 구조입니다. 이것 또는 `SchemaVersionId`가 제공되어야 합니다.

- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전에 할당된 고유 ID입니다. 이것 또는 `SchemaId`가 제공되어야 합니다.
- `SchemaVersionNumber` - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.

SerDeInfo 구조

추출기 및 로더 역할을 하는 직렬화/역직렬화 프로그램(SerDe)에 대한 정보입니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  SerDe의 이름입니다.
- `SerializationLibrary` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  일반적으로 SerDe를 구현하는 클래스입니다. 예를 들면, `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`입니다.
- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 키 문자열입니다.

  각 값은 512000 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 키-값 쌍은 SerDe의 초기화 파라미터를 정의합니다.

SkewedInfo 구조

테이블에 왜곡된 값을 지정합니다. 왜곡된 값은 매우 높은 빈도를 통해 발생한 값입니다.

필드

- `SkewedColumnNames` – UTF-8 문자열의 배열입니다.

  왜곡된 값이 포함된 열의 이름 목록입니다.
- `SkewedColumnValues` – UTF-8 문자열의 배열입니다.

너무 자주 나타나서 왜곡된 것으로 간주되는 값의 목록입니다.

- `SkewedColumnValueLocationMaps` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  왜곡된 값을 이 값을 포함하는 열에 매핑.

운영

- CreatePartition 작업(Python: create_partition)
- BatchCreatePartition 작업(Python: batch_create_partition)
- UpdatePartition 작업(Python: update_partition)
- DeletePartition 작업(Python: delete_partition)
- BatchDeletePartition 작업(Python: batch_delete_partition)
- GetPartition 작업(Python: get_partition)
- GetPartitions 작업(Python: get_partitions)
- BatchGetPartition 작업(Python: batch_get_partition)
- BatchUpdatePartition 작업(Python: batch_update_partition)
- GetColumnStatisticsForPartition 작업(Python: get_column_statistics_for_partition)
- UpdateColumnStatisticsForPartition 작업(Python: update_column_statistics_for_partition)
- DeleteColumnStatisticsForPartition 작업(Python: delete_column_statistics_for_partition)

CreatePartition 작업(Python: create_partition)

새 파티션을 생성합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다

  파티션이 생성될 카탈로그의 AWS 계정 ID입니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  생성될 파티션이 있는 메타데이터 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  생성될 파티션이 있는 메타데이터 테이블 이름입니다.

- `PartitionInput` – 필수(Required): PartitionInput 객체입니다.

  `PartitionInput` 구조는 생성될 파티션을 정의합니다.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `GlueEncryptionException`

BatchCreatePartition 작업(Python: batch_create_partition)

배치 작업에서 하나 이상의 파티션을 만듭니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  파티션이 생성될 카탈로그 ID입니다. 현재는 AWS 계정 ID여야 합니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  생성될 파티션이 있는 메타데이터 데이터베이스 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  생성될 파티션이 있는 메타데이터 테이블 이름입니다.
- `PartitionInputList` – 필수(Required): PartitionInput 객체의 배열이며 구조는 100개 이하입니다.

  `PartitionInput` 구조 목록은 생성될 파티션을 정의합니다.

응답

- `Errors` – PartitionError 객체의 배열입니다.

  요청된 파티션을 생성하고자 할 때 발생한 오류입니다.

오류

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `GlueEncryptionException`

UpdatePartition 작업(Python: update_partition)

파티션 업데이트

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

업데이트되는 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 테이블이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트되는 파티션이 있는 테이블 이름입니다.

- `PartitionValueList` – 필수(Required): 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  업데이트할 파티션을 정의하는 파티션 키 값 목록입니다.

- `PartitionInput` – 필수(Required): PartitionInput 객체입니다.

  파티션을 업데이트할 새로운 파티션 객체.

  `Values` 속성을 변경할 수 없습니다. 파티션의 파티션 키 값을 변경하려면 파티션을 삭제하고 다시 생성합니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeletePartition 작업(Python: delete_partition)

지정된 파티션을 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  삭제되는 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 테이블이 있는 카탈로그 데이터베이스 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제될 파티션이 포함된 테이블 이름입니다.
- `PartitionValues` – 필수(Required): UTF-8 문자열의 배열입니다.

  파티션을 정의하는 값입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchDeletePartition 작업(Python: batch_delete_partition)

배치 작업에서 하나 이상의 파티션을 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

삭제되는 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 테이블이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제될 파티션이 포함된 테이블 이름입니다.

- `PartitionsToDelete` – 필수(Required): PartitionValueList 객체의 배열이며 구조는 25개 이하입니다.

  `PartitionInput` 구조 목록은 삭제되는 파티션을 정의합니다.

응답

- `Errors` – PartitionError 객체의 배열입니다.

  요청된 파티션을 삭제하고자 할 때 발생한 오류입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetPartition 작업(Python: get_partition)

지정된 파티션에 대한 정보를 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.

- `PartitionValues` – 필수(Required): UTF-8 문자열의 배열입니다.

  파티션을 정의하는 값입니다.

응답

- `Partition` – Partition 객체입니다.

  `Partition` 객체의 형식으로 요청된 정보입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `FederationSourceException`
- `FederationSourceRetryableException`

GetPartitions 작업(Python: get_partitions)

테이블의 파티션에 대한 정보를 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.

- `Expression` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 조건자 문자열입니다.

  반환할 파티션을 필터링하는 표현식입니다.

  이 표현식은 SQL `WHERE` 필터 절과 유사한 SQL 구문을 사용합니다. SQL 문 파서 JSQLParser는 이 표현식을 구문 분석합니다.

  연산자: 다음은 `Expression` API 호출에서 사용할 수 있는 연산자입니다.

  =

  두 피연산자의 값이 동일한지 여부를 확인하고, 동일한 경우에는 이 조건이 true가 됩니다.

  예제: '변수 a'에는 10이 들어 있고 '변수 b'에는 20이 들어 있다고 가정해 보겠습니다.

  (a = b)는 true가 아닙니다.

  < >

  두 피연산자의 값이 동일한지 여부를 확인하고, 값이 동일하지 않으면 이 조건이 true가 됩니다.

  예제: (a < > b)는 true입니다.

  >

  왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 큰지 여부를 확인하고, 크면 이 조건이 true가 됩니다.

예제: (a > b)는 true가 아닙니다.

<

왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작은지 여부를 확인하고, 크면 이 조건이 true가 됩니다.

예제: (a < b)는 true입니다.

>=

왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 크거나 같은지 여부를 확인하고, 크거나 같으면 이 조건이 true가 됩니다.

예제: (a >= b)는 true가 아닙니다.

<=

왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작거나 같은지 여부를 확인하고, 작거나 같으면 이 조건이 true가 됩니다.

예제: (a <= b)는 true입니다.

AND, OR, IN, BETWEEN, LIKE, NOT, IS NULL

논리 연산자

지원되는 파티션 키 유형: 다음은 지원되는 파티션 키입니다.

- `string`
- `date`
- `timestamp`
- `int`
- `bigint`
- `long`
- `tinyint`
- `smallint`
- `decimal`

유효하지 않은 유형이 있으면 예외가 발생합니다.

다음 목록에는 각 유형에 대한 유효 연산자가 표시됩니다. 크롤러를 정의할 때 partitionKey 유형이 카탈로그 파티션과 호환될 STRING으로 생성됩니다.

단순 API 호출:

Example

twitter_partition 표에는 파티션 세 개가 있습니다.

```
year = 2015
        year = 2016
        year = 2017
```

Example

Get partition year가 2015와 같음

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year*=*'2015'"
```

Example

Get Partition year가 2016~2018(제외)

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year>'2016' AND year<'2018'"
```

Example

Get Partition year가 2015~2018(포함) 다음 API 호출은 서로에 상응합니다.

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year>='2015' AND year<='2018'"

        aws glue get-partitions --database-name dbname --table-name
 twitter_partition
        --expression "year BETWEEN 2015 AND 2018"

        aws glue get-partitions --database-name dbname --table-name
 twitter_partition
```

```
        --expression "year IN (2015,2016,2017,2018)"
```

Example

와일드카드 파티션 필터. 여기서는 다음 호출 출력이 partition year=2017입니다. 정규식은 LIKE에서 지원되지 않습니다.

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year LIKE '%7'"
```

- NextToken – UTF-8 문자열입니다.

  이것이 파티션을 가져오기 위한 첫 번째 호출이 아니면 지속적인 토큰입니다.
- Segment – 세그먼트 객체입니다.

  이 요청에 따라 스캔할 테이블 파티션의 세그먼트입니다.
- MaxResults – 1~1,000의 숫자(정수)입니다.

  한 번의 응답으로 반환할 최대 파티션 수.
- ExcludeColumnSchema – 부울입니다.

  true이면 파티션 열 스키마를 반환하지 않도록 지정합니다. 파티션 값이나 위치와 같은 다른 파티션 속성에만 관심이 있을 때 유용합니다. 이 접근 방식을 사용하면 중복 데이터가 반환되지 않으므로 큰 응답 문제를 피할 수 있습니다.
- TransactionId – Custom string pattern #43과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 콘텐츠를 읽을 트랜잭션 ID입니다.
- QueryAsOfTime – 타임스탬프입니다.

  파티션 콘텐츠를 읽을 기준 시간입니다. 설정하지 않으면 가장 최근의 트랜잭션 커밋 시간이 사용됩니다. TransactionId과(와) 함께 지정할 수 없습니다.

응답

- Partitions – Partition 객체의 배열입니다.

  요청된 파티션 목록.
- NextToken – UTF-8 문자열입니다.

반환된 파티션 목록에 마지막 항목이 포함되지 않은 경우의 연속 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

BatchGetPartition 작업(Python: batch_get_partition)

배치 요청에 따라 파티션을 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.
- `PartitionsToGet` – 필수(Required): PartitionValueList 객체의 배열이며 구조는 1,000개 이하입니다.

파티션 값의 목록은 가져올 파티션을 식별합니다.

응답

- `Partitions` – Partition 객체의 배열입니다.

  요청한 파티션의 목록입니다.
- `UnprocessedKeys` – PartitionValueList 객체의 배열이며 구조는 1,000개 이하입니다.

  파티션이 반환되지 않은 요청의 파티션 값 목록입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `FederationSourceException`
- `FederationSourceRetryableException`

BatchUpdatePartition 작업(Python: batch_update_partition)

배치 작업에서 하나 이상의 파티션을 업데이트합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  파티션이 업데이트될 카탈로그의 ID입니다. 현재는 AWS 계정 ID여야 합니다.
- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트될 파티션이 있는 메타데이터 데이터베이스의 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트될 파티션이 있는 메타데이터 테이블의 이름입니다.

- `Entries` – 필수(Required): BatchUpdatePartitionRequestEntry 객체의 배열이며 구조는 1~100개입니다.

  업데이트할 최대 100개의 `BatchUpdatePartitionRequestEntry` 객체 목록입니다.

응답

- `Errors` – BatchUpdatePartitionFailureEntry 객체의 배열입니다.

  요청된 파티션을 업데이트하고자 할 때 발생한 오류입니다. `BatchUpdatePartitionFailureEntry` 객체의 목록.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`

GetColumnStatisticsForPartition 작업(Python: get_column_statistics_for_partition)

열의 파티션 통계를 검색합니다.

이 작업에 필요한 Identity and Access Management(IAM) 권한은 `GetPartition`입니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.

- `PartitionValues` – 필수(Required): UTF-8 문자열의 배열입니다.

  파티션을 식별하는 파티션 값의 목록입니다.

- `ColumnNames` – 필수(Required): 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  열 이름의 목록입니다.

응답

- `ColumnStatisticsList` – ColumnStatistics 객체의 배열입니다.

  검색에 실패한 ColumnStatistics 목록입니다.

- `Errors` – ColumnError 객체의 배열입니다.

  열 통계 데이터를 검색하는 동안 오류가 발생했습니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

UpdateColumnStatisticsForPartition 작업(Python: update_column_statistics_for_partition)

열의 파티션 통계를 생성하거나 업데이트합니다.

이 작업에 필요한 Identity and Access Management(IAM) 권한은 `UpdatePartition`입니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.
- `PartitionValues` – 필수(Required): UTF-8 문자열의 배열입니다.

  파티션을 식별하는 파티션 값의 목록입니다.
- `ColumnStatisticsList` – 필수(Required): ColumnStatistics 객체의 배열이며 구조는 25개 이하입니다.

  열 통계의 목록입니다.

응답

- `Errors` – ColumnStatisticsError 객체의 배열입니다.

  열 통계 데이터를 업데이트하는 동안 오류가 발생했습니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeleteColumnStatisticsForPartition 작업(Python: delete_column_statistics_for_partition)

열의 파티션 열 통계를 삭제합니다.

이 작업에 필요한 Identity and Access Management(IAM) 권한은 `DeletePartition`입니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  문제의 파티션이 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션이 있는 카탈로그 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  파티션 테이블의 이름입니다.

- `PartitionValues` – 필수(Required): UTF-8 문자열의 배열입니다.

  파티션을 식별하는 파티션 값의 목록입니다.

- `ColumnName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`

- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

연결 API

연결 API는 AWS Glue에서의 연결 작업과 관련된 API 및 데이터 유형에 대해 설명합니다.

주제

- 연결 API
- 연결 유형 API
- 연결 메타데이터 및 미리 보기 API

연결 API

연결 API는 AWS Glue 연결 데이터 유형과 연결을 생성, 삭제, 업데이트 및 나열하기 위한 API에 대해 설명합니다.

데이터 타입

- 연결 구조
- ConnectionInput 구조
- TestConnectionInput 구조
- PhysicalConnectionRequirements 구조
- GetConnectionsFilter 구조
- AuthenticationConfiguration 구조
- AuthenticationConfigurationInput 구조
- OAuth2Properties 구조
- OAuth2PropertiesInput 구조
- OAuth2ClientApplication 구조
- AuthorizationCodeProperties 구조
- BasicAuthenticationCredentials 구조
- OAuth2Credentials 구조

연결 구조

데이터 원본으로 연결을 정의합니다.

필드

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결 정의 이름입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  연결에 대한 설명입니다.

- ConnectionType – UTF-8 문자열(유효한 값: JDBC | SFTP | MONGODB | KAFKA | NETWORK | MARKETPLACE | CUSTOM | SALESFORCE | VIEW_VALIDATION_REDSHIFT | VIEW_VALIDATION_ATHENA | GOOGLEADS | GOOGLESHEETS | GOOGLEANALYTICS4 | SERVICENOW | MARKETO | SAPODATA | ZENDESK | JIRACLOUD | NETSUITEERP | HUBSPOT | FACEBOOKADS | INSTAGRAMADS | ZOHOCRM | SALESFORCEPARDOT | SALESFORCEMARKETINGCLOUD | SLACK | STRIPE | INTERCOM | SNAPCHATADS).

  연결 유형입니다. 현재 SFTP는 지원되지 않습니다.

- MatchCriteria – 10개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  이 연결을 선택할 때 사용할 수 있는 기준입니다.

- ConnectionProperties – 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다(유효한 값: HOST | PORT | USERNAME="USER_NAME" | PASSWORD | ENCRYPTED_PASSWORD | JDBC_DRIVER_JAR_URI | JDBC_DRIVER_CLASS_NAME | JDBC_ENGINE | JDBC_ENGINE_VERSION | CONFIG_FILES | INSTANCE_ID | JDBC_CONNECTION_URL | JDBC_ENFORCE_SSL | CUSTOM_JDBC_CERT | SKIP_CUSTOM_JDBC_CERT_VALIDATION | CUSTOM_JDBC_CERT_STRING | CONNECTION_URL | KAFKA_BOOTSTRAP_SERVERS | KAFKA_SSL_ENABLED | KAFKA_CUSTOM_CERT | KAFKA_SKIP_CUSTOM_CERT_VALIDATION | KAFKA_CLIENT_KEYSTORE | KAFKA_CLIENT_KEYSTORE_PASSWORD | KAFKA_CLIENT_KEY_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD | KAFKA_SASL_MECHANISM | KAFKA_SASL_PLAIN_USERNAME | KAFKA_SASL_PLAIN_PASSWORD | ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD | KAFKA_SASL_SCRAM_USERNAME

| KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_SCRAM_SECRETS_ARN | ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_GSSAPI_KEYTAB | KAFKA_SASL_GSSAPI_KRB5_CONF | KAFKA_SASL_GSSAPI_SERVICE | KAFKA_SASL_GSSAPI_PRINCIPAL | SECRET_ID | CONNECTOR_URL | CONNECTOR_TYPE | CONNECTOR_CLASS_NAME | ENDPOINT | ENDPOINT_TYPE | ROLE_ARN | REGION | WORKGROUP_NAME | CLUSTER_IDENTIFIER | DATABASE).

각 값은 길이가 1~1,024바이트인 값 문자열입니다.

이러한 키-값 페어는 버전 1 연결 스키마를 사용할 때 연결에 대한 파라미터를 정의합니다.

- HOST - 호스트 URI: 데이터베이스 호스트의 IPv4 주소 또는 FQDN(fully qualified domain name: 정규화된 도메인 이름).
- PORT - 데이터베이스 호스트가 데이터베이스 연결을 수신 중인 포트의 포트 번호(1,024~65,535) 입니다.
- USER_NAME- 데이터베이스에 로그인할 이름. USER_NAME의 값 문자열은 "USERNAME"입니다.
- PASSWORD- 사용자 이름에 해당하는 암호(사용할 경우).
- ENCRYPTED_PASSWORD - 데이터 카탈로그 암호화 설정에서 ConnectionPasswordEncryption을 설정하여 연결 암호 보호를 활성화하면 이 필드가 암호화된 암호를 저장합니다.
- JDBC_DRIVER_JAR_URI - 사용할 JDBC 드라이버가 포함된 JAR 파일의 Amazon Simple Storage Service(Amazon S3) 경로.
- JDBC_DRIVER_CLASS_NAME- 사용할 JDBC 드라이버의 클래스 이름.
- JDBC_ENGINE- 사용할 JDBC 엔진의 이름.
- JDBC_ENGINE_VERSION - 사용할 JDBC 엔진의 버전.
- CONFIG_FILES - (추후 사용 예약.)
- INSTANCE_ID- 사용할 인스턴스 ID.
- JDBC_CONNECTION_URL - JDBC 데이터 원본에 연결하기 위한 URL입니다.
- JDBC_ENFORCE_SSL - 호스트 이름이 일치하는 SSL(Secure Sockets Layer)을 클라이언트의 JDBC 연결용으로 적용할지 여부를 지정하는, 대소문자 구분이 없는 부울 문자열(true, false)입니다. 기본값은 false입니다.
- CUSTOM_JDBC_CERT - 고객의 루트 인증서를 지정하는 Amazon S3 위치입니다. AWS Glue는 이 루트 인증서를 사용하여 고객 데이터베이스에 연결할 때 고객의 인증서를 검증합니다. AWS Glue는 X.509 인증서만 처리합니다. 인증서는 DER로 인코딩되어 Base64 인코딩 PEM 형식으로 제공되어야 합니다.

- SKIP_CUSTOM_JDBC_CERT_VALIDATION - 기본적으로 false입니다. AWS Glue는 고객 인증서의 서명 알고리즘 및 주제 퍼블릭 키 알고리즘을 검증합니다. 서명 알고리즘에 허용되는 유일한 알고리즘은 SHA256withRSA, SHA384withRSA 또는 SHA512withRSA. 주제 퍼블릭 키 알고리즘의 경우, 키 길이는 2048 이상이어야 합니다. 이 속성 값을 true로 설정하여 AWS Glue의 고객 인증서 검증을 건너뛸 수 있습니다.
- CUSTOM_JDBC_CERT_STRING - 중간자 (man-in-the-middle) 공격을 방지하기 위해 도메인 일치 또는 고유 이름 일치에 사용되는 사용자 지정 JDBC 인증서 문자열입니다. Oracle Database에서는 SSL_SERVER_CERT_DN으로, Microsoft SQL Server에서는 hostNameInCertificate로 사용됩니다.
- CONNECTION_URL - 일반(비JDBC) 데이터 원본에 연결하기 위한 URL입니다.
- SECRET_ID - 자격 증명의 보안 관리자에 사용되는 보안 암호 ID입니다.
- CONNECTOR_URL - MARKETPLACE 또는 CUSTOM 연결을 위한 커넥터 URL입니다.
- CONNECTOR_TYPE - MARKETPLACE 또는 CUSTOM 연결을 위한 커넥터 유형입니다.
- CONNECTOR_CLASS_NAME - MARKETPLACE 또는 CUSTOM 연결을 위한 커넥터 클래스 이름입니다.
- KAFKA_BOOTSTRAP_SERVERS - Kafka 클라이언트에서 연결하고 자체 부트스트랩하는 Kafka 클러스터에 있는 Apache Kafka 브로커의 주소입니다 호스트 및 포트 쌍의 쉼표로 구분된 목록입니다.
- KAFKA_SSL_ENABLED - Apache Kafka 연결에서 SSL을 사용할지 아니면 사용 중지할지 여부입니다. 기본값은 "true"입니다.
- KAFKA_CUSTOM_CERT - 프라이빗 CA 인증서 파일(.pem 포맷)의 Amazon S3 URL입니다. 기본값은 빈 문자열입니다.
- KAFKA_SKIP_CUSTOM_CERT_VALIDATION - CA 인증서 파일의 검증을 건너뛸지 여부입니다. AWS Glue는 SHA256withRSA, SHA384withRSA 및 SHA512withRSA의 세 가지 알고리즘에 대해 검증합니다. 기본값은 "false"입니다.
- KAFKA_CLIENT_KEYSTORE - Kafka 클라이언트 측 인증을 위한 클라이언트 키 스토어 파일의 Amazon S3 위치입니다(선택 사항).
- KAFKA_CLIENT_KEYSTORE_PASSWORD - 제공된 키 스토어에 액세스하기 위한 암호입니다(선택 사항).
- KAFKA_CLIENT_KEY_PASSWORD - 키 스토어는 여러 키로 구성 될 수 있으므로 Kafka 서버 측 키와 함께 사용할 클라이언트 키에 액세스하기 위한 암호입니다(선택 사항).
- ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD - Kafka 클라이언트 키 스토어 암호의 암호화된 버전입니다(사용자가 AWS Glue 암호 암호화 설정을 선택한 경우).

- ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD - Kafka 클라이언트 키 암호의 암호화된 버전입니다(사용자가 AWS Glue 암호 암호화 설정을 선택한 경우).
- KAFKA_SASL_MECHANISM - "SCRAM-SHA-512", "GSSAPI", "AWS_MSK_IAM" 또는 "PLAIN". 다음은 지원되는 SASL 메커니즘입니다.
- KAFKA_SASL_PLAIN_USERNAME - "PLAIN" 메커니즘으로 인증하는 데 사용되는 일반 텍스트 사용자 이름입니다.
- KAFKA_SASL_PLAIN_PASSWORD - "PLAIN" 메커니즘으로 인증하는 데 사용되는 일반 텍스트 암호입니다.
- ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD - Kafka SASL PLAIN 암호의 암호화된 버전입니다(사용자가 AWS Glue 암호 암호화 설정을 선택한 경우).
- KAFKA_SASL_SCRAM_USERNAME - "SCRAM-SHA-512" 메커니즘으로 인증하는 데 사용되는 일반 텍스트 사용자 이름입니다.
- KAFKA_SASL_SCRAM_PASSWORD - "SCRAM-SHA-512" 메커니즘으로 인증하는 데 사용되는 일반 텍스트 암호입니다.
- ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD - Kafka SASL SCRAM 암호의 암호화된 버전입니다(사용자가 AWS Glue 암호 암호화 설정을 선택한 경우).
- KAFKA_SASL_SCRAM_SECRETS_ARN - AWS Secrets Manager에서 보안 암호의 Amazon 리소스 이름(ARN)입니다.
- KAFKA_SASL_GSSAPI_KEYTAB - Kerberos keytab 파일의 S3 위치입니다. keytab은 하나 이상의 보안 주체에 대한 장기 키를 저장합니다. 자세한 내용은 MIT Kerberos Documentation: Keytab(MIT Kerberos 설명서: Keytab)을 참조하세요.
- KAFKA_SASL_GSSAPI_KRB5_CONF - Kerberos krb5.conf 파일의 S3 위치입니다. krb5.conf는 KDC 서버의 위치와 같은 Kerberos 구성 정보를 저장합니다. 자세한 내용은 MIT Kerberos Documentation: krb5.conf(MIT Kerberos 설명서: krb5.conf)를 참조하세요.
- KAFKA_SASL_GSSAPI_SERVICE - Kafka 구성에서 sasl.kerberos.service.name으로 설정된 Kerberos 서비스 이름입니다.
- KAFKA_SASL_GSSAPI_PRINCIPAL - AWS Glue에서 사용하는 Kerberos 보안 주체의 이름입니다. 자세한 내용은 Kafka Documentation: Configuring Kafka Brokers(Kafka 설명서: Kafka 브로커 구성)를 참조하세요.
- ROLE_ARN - 쿼리를 실행하는 데 사용할 역할.
- REGION - 쿼리를 실행할 AWS 리전.
- WORKGROUP_NAME - 쿼리가 실행되는 Amazon Redshift Serverless 작업 그룹 또는 Amazon Athena 작업 그룹의 이름.

  - `CLUSTER_IDENTIFIER` - 쿼리가 실행되는 Amazon Redshift 클러스터의 클러스터 식별자.
  - `DATABASE` - 연결하려는 Amazon Redshift 데이터베이스.
- `SparkProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~2,048바이트인 UTF-8 문자열입니다.

  Spark 컴퓨팅 환경과 관련된 연결 속성입니다.
- `AthenaProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~2,048바이트인 UTF-8 문자열입니다.

  Athena 컴퓨팅 환경과 관련된 연결 속성입니다.
- `PythonProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~2,048바이트인 UTF-8 문자열입니다.

  Python 컴퓨팅 환경과 관련된 연결 속성입니다.
- `PhysicalConnectionRequirements` – PhysicalConnectionRequirements 객체입니다.

  Virtual Private Cloud(VPC) 및 `SecurityGroup`과 같이 이 연결을 설정하는 데 필요한 물리적 연결 요구 사항입니다.
- `CreationTime` – 타임스탬프입니다.

  이 연결 정의가 생성된 시간의 타임스탬프입니다.
- `LastUpdatedTime` – 타임스탬프입니다.

  이 연결 정의가 업데이트된 마지막 시간의 타임스탬프입니다.
- `LastUpdatedBy` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 연결 정의를 마지막으로 업데이트한 사용자, 그룹 혹은 역할입니다.
- `Status` – UTF-8 문자열입니다(유효한 값: `READY` | `IN_PROGRESS` | `FAILED`).

연결 상태입니다. READY, IN_PROGRESS 또는 FAILED 중 하나일 수 있습니다.

- StatusReason – UTF-8 문자열입니다(1~16384바이트).

  연결 상태의 이유입니다.

- LastConnectionValidationTime – 타임스탬프입니다.

  이 연결이 마지막으로 검증된 시간의 타임스탬프입니다.

- AuthenticationConfiguration – AuthenticationConfiguration 객체입니다.

  연결의 인증 속성입니다.

- ConnectionSchemaVersion - 1 이상 2 이하의 숫자(정수)입니다.

  이 연결에 대한 연결 스키마의 버전입니다. 버전 2는 특정 컴퓨팅 환경에 대한 속성을 지원합니다.

- CompatibleComputeEnvironments – UTF-8 문자열의 배열입니다.

  연결과 호환되는 컴퓨팅 환경 목록입니다.

ConnectionInput 구조

생성 혹은 업데이트를 위한 연결을 지정할 때 사용되는 구조입니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결의 이름입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  연결에 대한 설명입니다.

- ConnectionType – 필수: UTF-8 문자열(유효한 값: JDBC | SFTP | MONGODB | KAFKA | NETWORK | MARKETPLACE | CUSTOM | SALESFORCE | VIEW_VALIDATION_REDSHIFT | VIEW_VALIDATION_ATHENA | GOOGLEADS | GOOGLESHEETS | GOOGLEANALYTICS4 | SERVICENOW | MARKETO | SAPODATA | ZENDESK | JIRACLOUD | NETSUITEERP | HUBSPOT | FACEBOOKADS | INSTAGRAMADS | ZOHOCRM | SALESFORCEPARDOT | SALESFORCEMARKETINGCLOUD | SLACK | STRIPE | INTERCOM | SNAPCHATADS).

연결 유형입니다. 현재 이러한 유형이 지원됩니다.

- JDBC - JDBC(Java Database Connectivity)를 통해 데이터베이스에 대한 연결을 지정합니다.

  JDBC 연결은 다음 ConnectionParameters를 사용합니다.

  - 필수 항목: JDBC_CONNECTION_URL 또는 (HOST, PORT, JDBC_ENGINE) 모두.
  - 필수 항목: SECRET_ID 또는 (USERNAME, PASSWORD) 모두.
  - 선택 사항: JDBC_ENFORCE_SSL, CUSTOM_JDBC_CERT, CUSTOM_JDBC_CERT_STRING, SKIP_CUSTOM_JDBC_CERT_VALIDATION. 이러한 파라미터는 JDBC로 SSL을 구성하는 데 사용됩니다.

- KAFKA - Apache Kafka 스트리밍 플랫폼에 대한 연결을 지정합니다.

  KAFKA 연결은 다음 ConnectionParameters를 사용합니다.

  - 필수 항목: KAFKA_BOOTSTRAP_SERVERS.
  - 선택 사항: KAFKA_SSL_ENABLED, KAFKA_CUSTOM_CERT, KAFKA_SKIP_CUSTOM_CERT_VALIDATION. 이러한 파라미터는 KAFKA로 SSL을 구성하는 데 사용됩니다.
  - 선택 사항: KAFKA_CLIENT_KEYSTORE, KAFKA_CLIENT_KEYSTORE_PASSWORD, KAFKA_CLIENT_KEY_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD. 이러한 파라미터는 KAFKA에서 SSL로 TLS 클라이언트 구성을 지정하는 데 사용됩니다.
  - 선택 사항: KAFKA_SASL_MECHANISM. SCRAM-SHA-512, GSSAPI 또는 AWS_MSK_IAM으로 지정할 수 있습니다.
  - 선택 사항: KAFKA_SASL_SCRAM_USERNAME, KAFKA_SASL_SCRAM_PASSWORD, ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD. 이러한 파라미터는 KAFKA로 SASL/SCRAM-SHA-512 인증을 구성하는 데 사용됩니다.
  - 선택 사항: KAFKA_SASL_GSSAPI_KEYTAB, KAFKA_SASL_GSSAPI_KRB5_CONF, KAFKA_SASL_GSSAPI_SERVICE, KAFKA_SASL_GSSAPI_PRINCIPAL. 이러한 파라미터는 KAFKA로 SASL/GSSAPI 인증을 구성하는 데 사용됩니다.

- MONGODB - MongoDB 문서 데이터베이스에 대한 연결을 지정합니다.

  MONGODB 연결은 다음 ConnectionParameters를 사용합니다.

  - 필수 항목: CONNECTION_URL.
  - 필수 항목: SECRET_ID 또는 (USERNAME, PASSWORD) 모두.

- VIEW_VALIDATION_REDSHIFT - Amazon Redshift에서 보기 검증에 사용되는 연결을 지정합니다.
- VIEW_VALIDATION_ATHENA - Amazon Athena에서 보기 검증에 사용되는 연결을 지정합니다.
- NETWORK - Amazon Virtual Private Cloud(Amazon VPC) 환경 내의 데이터 원본에 대한 네트워크 연결을 지정합니다.

  NETWORK 연결에는 ConnectionParameters가 필요하지 않습니다. 대신 PhysicalConnectionRequirements를 제공합니다.
- MARKETPLACE - AWS Marketplace에서 구입한 커넥터에 포함된 구성 설정을 사용하여 AWS Glue에서 기본적으로 지원하지 않는 데이터 스토어에서 읽고 씁니다.

  MARKETPLACE 연결은 다음 ConnectionParameters를 사용합니다.
  - 필수 항목: CONNECTOR_TYPE, CONNECTOR_URL, CONNECTOR_CLASS_NAME, CONNECTION_URL.
  - JDBC CONNECTOR_TYPE 연결에 필요: SECRET_ID 또는 (USERNAME, PASSWORD) 모두.
- CUSTOM - AWS Glue에서 기본적으로 지원하지 않는 데이터 스토어에서 읽고 쓰기 위해 사용자 정의 커넥터에 포함된 구성 설정을 사용합니다.

또한 다음 SaaS 커넥터에 대한 ConnectionType이 지원됩니다.

- FACEBOOKADS - Facebook Ads에 대한 연결을 지정합니다.
- GOOGLEADS - Google Ads에 대한 연결을 지정합니다.
- GOOGLESHEETS - Google 시트에 대한 연결을 지정합니다.
- GOOGLEANALYTICS4 - Google Analytics 4에 대한 연결을 지정합니다.
- HUBSPOT - HubSpot에 대한 연결을 지정합니다.
- INSTAGRAMADS - Instagram Ads에 대한 연결을 지정합니다.
- INTERCOM - Intercom에 대한 연결을 지정합니다.
- JIRACLOUD - Jira Cloud에 대한 연결을 지정합니다.
- MARKETO - Adobe Marketo Engage에 대한 연결을 지정합니다.
- NETSUITEERP - Oracle NetSuite에 대한 연결을 지정합니다.
- SALESFORCE - OAuth 인증을 사용하여 Salesforce 연결을 지정합니다.
- SALESFORCEMARKETINGCLOUD - Salesforce Marketing Cloud에 대한 연결을 지정합니다.
- SALESFORCEPARDOT - Salesforce Marketing Cloud Account Engagement(MCAE)에 대한 연결을 지정합니다.

- SAPODATA - SAP OData에 대한 연결을 지정합니다.
- SERVICENOW - ServiceNow에 대한 연결을 지정합니다.
- SLACK - Slack에 대한 연결을 지정합니다.
- SNAPCHATADS - Snapchat Ads에 대한 연결을 지정합니다.
- STRIPE - Stripe에 대한 연결을 지정합니다.
- ZENDESK - Zendesk에 대한 연결을 지정합니다.
- ZOHOCRM - Zoho CRM에 대한 연결을 지정합니다.

특정 커넥터에 필요한 연결 파라미터에 대한 자세한 내용은 AWS Glue 사용 설명서의 AWS Glue 연결 추가에서 커넥터에 대한 설명서를 참조하세요.

SFTP는 지원되지 않습니다.

선택적 ConnectionProperties를 사용하여 AWS Glue의 기능을 구성하는 방법에 대한 자세한 내용을 알아보려면 AWS Glue 연결 속성을 참조하세요.

선택적 ConnectionProperties를 사용하여 AWS Glue Studio의 기능을 구성하는 방법에 대한 자세한 내용을 알아보려면 커넥터 및 연결 사용을 참조하세요.

- MatchCriteria – 10개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  이 연결을 선택할 때 사용할 수 있는 기준입니다.

- ConnectionProperties – 필수(Required): 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다(유효한 값: HOST | PORT | USERNAME="USER_NAME" | PASSWORD | ENCRYPTED_PASSWORD | JDBC_DRIVER_JAR_URI | JDBC_DRIVER_CLASS_NAME | JDBC_ENGINE | JDBC_ENGINE_VERSION | CONFIG_FILES | INSTANCE_ID | JDBC_CONNECTION_URL | JDBC_ENFORCE_SSL | CUSTOM_JDBC_CERT | SKIP_CUSTOM_JDBC_CERT_VALIDATION | CUSTOM_JDBC_CERT_STRING | CONNECTION_URL | KAFKA_BOOTSTRAP_SERVERS | KAFKA_SSL_ENABLED | KAFKA_CUSTOM_CERT | KAFKA_SKIP_CUSTOM_CERT_VALIDATION | KAFKA_CLIENT_KEYSTORE | KAFKA_CLIENT_KEYSTORE_PASSWORD | KAFKA_CLIENT_KEY_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD | KAFKA_SASL_MECHANISM | KAFKA_SASL_PLAIN_USERNAME | KAFKA_SASL_PLAIN_PASSWORD | ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD | KAFKA_SASL_SCRAM_USERNAME

| KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_SCRAM_SECRETS_ARN | ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_GSSAPI_KEYTAB | KAFKA_SASL_GSSAPI_KRB5_CONF | KAFKA_SASL_GSSAPI_SERVICE | KAFKA_SASL_GSSAPI_PRINCIPAL | SECRET_ID | CONNECTOR_URL | CONNECTOR_TYPE | CONNECTOR_CLASS_NAME | ENDPOINT | ENDPOINT_TYPE | ROLE_ARN | REGION | WORKGROUP_NAME | CLUSTER_IDENTIFIER | DATABASE).

각 값은 길이가 1~1,024바이트인 값 문자열입니다.

이러한 키-값 쌍은 연결용 파라미터를 정의합니다.

- `SparkProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~2,048바이트인 UTF-8 문자열입니다.

  Spark 컴퓨팅 환경과 관련된 연결 속성입니다.
- `AthenaProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~2,048바이트인 UTF-8 문자열입니다.

  Athena 컴퓨팅 환경과 관련된 연결 속성입니다.
- `PythonProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~2,048바이트인 UTF-8 문자열입니다.

  Python 컴퓨팅 환경과 관련된 연결 속성입니다.
- `PhysicalConnectionRequirements` – PhysicalConnectionRequirements 객체입니다.

  Virtual Private Cloud(VPC) 및 `SecurityGroup`과 같이 이 연결을 설정하는 데 필요한 물리적 연결 요구 사항입니다.
- `AuthenticationConfiguration` – AuthenticationConfigurationInput 객체입니다.

  연결의 인증 속성입니다.
- `ValidateCredentials` – 부울입니다.

연결 생성 중에 자격 증명을 검증하기 위한 플래그입니다. 기본값은 true입니다.

- ValidateForComputeEnvironments – UTF-8 문자열의 배열입니다.

  지정된 연결 속성을 검증하는 데 기준이 되는 컴퓨팅 환경입니다.

TestConnectionInput 구조

서비스에 대한 연결 테스트를 지정할 때 사용되는 구조입니다.

필드

- ConnectionType – 필수: UTF-8 문자열(유효한 값: JDBC | SFTP | MONGODB | KAFKA | NETWORK | MARKETPLACE | CUSTOM | SALESFORCE | VIEW_VALIDATION_REDSHIFT | VIEW_VALIDATION_ATHENA | GOOGLEADS | GOOGLESHEETS | GOOGLEANALYTICS4 | SERVICENOW | MARKETO | SAPODATA | ZENDESK | JIRACLOUD | NETSUITEERP | HUBSPOT | FACEBOOKADS | INSTAGRAMADS | ZOHOCRM | SALESFORCEPARDOT | SALESFORCEMARKETINGCLOUD | SLACK | STRIPE | INTERCOM | SNAPCHATADS).

  테스트할 연결의 유형입니다. 이 작업은 JDBC 또는 SALESFORCE 연결 유형에만 사용할 수 있습니다.

- ConnectionProperties – 필수(Required): 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다(유효한 값: HOST | PORT | USERNAME="USER_NAME" | PASSWORD | ENCRYPTED_PASSWORD | JDBC_DRIVER_JAR_URI | JDBC_DRIVER_CLASS_NAME | JDBC_ENGINE | JDBC_ENGINE_VERSION | CONFIG_FILES | INSTANCE_ID | JDBC_CONNECTION_URL | JDBC_ENFORCE_SSL | CUSTOM_JDBC_CERT | SKIP_CUSTOM_JDBC_CERT_VALIDATION | CUSTOM_JDBC_CERT_STRING | CONNECTION_URL | KAFKA_BOOTSTRAP_SERVERS | KAFKA_SSL_ENABLED | KAFKA_CUSTOM_CERT | KAFKA_SKIP_CUSTOM_CERT_VALIDATION | KAFKA_CLIENT_KEYSTORE | KAFKA_CLIENT_KEYSTORE_PASSWORD | KAFKA_CLIENT_KEY_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD | KAFKA_SASL_MECHANISM | KAFKA_SASL_PLAIN_USERNAME | KAFKA_SASL_PLAIN_PASSWORD | ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD | KAFKA_SASL_SCRAM_USERNAME | KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_SCRAM_SECRETS_ARN | ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_GSSAPI_KEYTAB

| KAFKA_SASL_GSSAPI_KRB5_CONF | KAFKA_SASL_GSSAPI_SERVICE | KAFKA_SASL_GSSAPI_PRINCIPAL | SECRET_ID | CONNECTOR_URL | CONNECTOR_TYPE | CONNECTOR_CLASS_NAME | ENDPOINT | ENDPOINT_TYPE | ROLE_ARN | REGION | WORKGROUP_NAME | CLUSTER_IDENTIFIER | DATABASE).

각 값은 길이가 1~1,024바이트인 값 문자열입니다.

이 키-값 페어는 연결에 대한 파라미터를 정의합니다.

JDBC 연결에는 다음 연결 속성이 사용됩니다.

- 필수 항목: JDBC_CONNECTION_URL 또는 (HOST, PORT, JDBC_ENGINE) 모두.
- 필수 항목: SECRET_ID 또는 (USERNAME, PASSWORD) 모두.
- 선택 사항: JDBC_ENFORCE_SSL, CUSTOM_JDBC_CERT, CUSTOM_JDBC_CERT_STRING, SKIP_CUSTOM_JDBC_CERT_VALIDATION. 이러한 파라미터는 JDBC로 SSL을 구성하는 데 사용됩니다.

SALESFORCE 연결을 사용하려면 AuthenticationConfiguration 멤버를 구성해야 합니다.

- AuthenticationConfiguration – AuthenticationConfigurationInput 객체입니다.

  TestConnection 요청에서 인증 구성을 포함하는 구조입니다. OAuth 인증을 사용하여 Salesforce에 연결하는 데 필요합니다.

PhysicalConnectionRequirements 구조

GetConnection 응답의 OAuth 클라이언트 앱입니다.

필드

- SubnetId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결에서 사용하는 서브넷 ID입니다.
- SecurityGroupIdList – 50개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  연결에서 사용하는 보안 그룹 ID 목록입니다.
- AvailabilityZone – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결의 가용 영역입니다.

GetConnectionsFilter 구조

GetConnections API 작업에서 반환하는 연결 정의를 필터링합니다.

필드

- MatchCriteria – 10개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  연결 정의가 반환되도록 연결 정의에 기록된 기준과 반드시 일치해야 하는 기준 문자열입니다.
- ConnectionType – UTF-8 문자열(유효한 값: JDBC | SFTP | MONGODB | KAFKA | NETWORK | MARKETPLACE | CUSTOM | SALESFORCE | VIEW_VALIDATION_REDSHIFT | VIEW_VALIDATION_ATHENA | GOOGLEADS | GOOGLESHEETS | GOOGLEANALYTICS4 | SERVICENOW | MARKETO | SAPODATA | ZENDESK | JIRACLOUD | NETSUITEERP | HUBSPOT | FACEBOOKADS | INSTAGRAMADS | ZOHOCRM | SALESFORCEPARDOT | SALESFORCEMARKETINGCLOUD | SLACK | STRIPE | INTERCOM | SNAPCHATADS).

  반환할 연결 유형입니다. 현재 SFTP는 지원되지 않습니다.
- ConnectionSchemaVersion - 1 이상 2 이하의 숫자(정수)입니다.

  스키마 버전 1 또는 2로 연결이 생성되었는지 여부를 나타냅니다.

AuthenticationConfiguration 구조

인증 구성을 포함하는 구조입니다.

필드

- AuthenticationType – UTF-8 문자열입니다(유효한 값: BASIC | OAUTH2 | CUSTOM | IAM).

  인증 구성을 포함하는 구조입니다.
- SecretArn – Custom string pattern #36과(와) 일치하는 UTF-8 문자열입니다.

  자격 증명을 저장할 보안 관리자 ARN입니다.
- KmsKeyArn – Custom string pattern #29과(와) 일치하는 UTF-8 문자열입니다.

  민감한 인증 정보를 암호화하는 데 사용되는 KMS 키의 Amazon 리소스 이름(ARN)입니다. 이 키는 인증 구성 내에 저장된 자격 증명 및 기타 민감한 데이터를 보호하는 데 사용됩니다.
- OAuth2Properties – OAuth2Properties 객체입니다.

  OAuth2 인증을 위한 속성입니다.

AuthenticationConfigurationInput 구조

CreateConnection 요청에서 인증 구성을 포함하는 구조입니다.

필드

- `AuthenticationType` – UTF-8 문자열입니다(유효한 값: `BASIC` | `OAUTH2` | `CUSTOM` | `IAM`).

  CreateConnection 요청에서 인증 구성을 포함하는 구조입니다.
- `OAuth2Properties` – OAuth2PropertiesInput 객체입니다.

  CreateConnection 요청에서 OAuth2 인증의 속성입니다.
- `SecretArn` – Custom string pattern #36과(와) 일치하는 UTF-8 문자열입니다.

  CreateConnection 요청에서 자격 증명을 저장할 보안 관리자 ARN입니다.
- `KmsKeyArn` – Custom string pattern #29과(와) 일치하는 UTF-8 문자열입니다.

  연결을 암호화하는 데 사용되는 KMS 키의 ARN입니다. 요청에서 입력만 가져와 Secret Manager에 저장합니다.
- `BasicAuthenticationCredentials` – BasicAuthenticationCredentials 객체입니다.

  인증 유형이 기본 인증일 때 사용되는 자격 증명입니다.
- `CustomAuthenticationCredentials` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~2,048바이트인 UTF-8 문자열입니다.

  인증 유형이 사용자 지정 인증일 때 사용되는 자격 증명입니다.

OAuth2Properties 구조

OAuth2 인증을 위한 속성을 포함하는 구조입니다.

필드

- `OAuth2GrantType` – UTF-8 문자열입니다(유효한 값: `AUTHORIZATION_CODE` | `CLIENT_CREDENTIALS` | `JWT_BEARER`).

  OAuth2 권한 부여 유형입니다. 예: `AUTHORIZATION_CODE`, `JWT_BEARER` 또는 `CLIENT_CREDENTIALS`.

- OAuth2ClientApplication – OAuth2ClientApplication 객체입니다.

  클라이언트 애플리케이션 유형입니다. 예: AWS_MANAGED 또는 USER_MANAGED.

- TokenUrl – Custom string pattern #40과(와) 일치하는 256바이트 이하 길이의 UTF-8 문자열입니다.

  인증 코드를 액세스 토큰으로 교환하기 위한 제공업체 인증 서버 URL입니다.

- TokenUrlParametersMap – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~512바이트인 UTF-8 문자열입니다.

  토큰 GET 요청에 추가된 파라미터의 맵입니다.

OAuth2PropertiesInput 구조

CreateConnection 요청에서 OAuth2를 위한 속성을 포함하는 구조입니다.

필드

- OAuth2GrantType – UTF-8 문자열입니다(유효한 값: AUTHORIZATION_CODE | CLIENT_CREDENTIALS | JWT_BEARER).

  CreateConnection 요청에서 OAuth2 권한 부여 유형입니다. 예: AUTHORIZATION_CODE, JWT_BEARER 또는 CLIENT_CREDENTIALS.

- OAuth2ClientApplication – OAuth2ClientApplication 객체입니다.

  CreateConnection 요청에서 클라이언트 애플리케이션 유형입니다. 예: AWS_MANAGED또는 USER_MANAGED.

- TokenUrl – Custom string pattern #40과(와) 일치하는 256바이트 이하 길이의 UTF-8 문자열입니다.

  인증 코드를 액세스 토큰으로 교환하기 위한 제공업체 인증 서버 URL입니다.

- TokenUrlParametersMap – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~512바이트인 UTF-8 문자열입니다.

토큰 GET 요청에 추가된 파라미터의 맵입니다.

- AuthorizationCodeProperties – AuthorizationCodeProperties 객체입니다.

  OAuth2 AUTHORIZATION_CODE 권한 부여 유형에 필요한 속성 세트입니다.

- OAuth2Credentials – OAuth2Credentials 객체입니다.

  인증 유형이 OAuth2 인증일 때 사용되는 자격 증명입니다.

OAuth2ClientApplication 구조

연결에 사용된 OAuth2 클라이언트 앱입니다.

필드

- UserManagedClientApplicationClientId – Custom string pattern #37과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  ClientAppType이 USER_MANAGED인 경우 클라이언트 애플리케이션의 clientID입니다.

- AWSManagedClientApplicationReference – Custom string pattern #37과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  AWS 관리형인 SaaS 측 클라이언트 앱에 대한 참조입니다.

AuthorizationCodeProperties 구조

OAuth2 AUTHORIZATION_CODE 권한 부여 유형 워크플로에 필요한 속성 세트입니다.

필드

- AuthorizationCode – Custom string pattern #37과(와) 일치하는 1~4096바이트 길이의 UTF-8 문자열입니다.

  AUTHORIZATION_CODE 권한 부여 워크플로의 세 번째 레그에서 사용할 인증 코드입니다. 이 코드는 액세스 토큰으로 교환되면 유효하지 않게 되는 일회용 코드이므로 이 값을 요청 파라미터로 사용하는 것이 허용됩니다.

- RedirectUri – Custom string pattern #41과(와) 일치하는 512바이트 이하 길이의 UTF-8 문자열입니다.

인증 코드를 발급할 때 권한 부여 서버가 사용자를 리디렉션하는 리디렉션 URI입니다. 계속해서 인증 코드를 액세스 토큰으로 교환할 때 이 URI가 사용됩니다.

BasicAuthenticationCredentials 구조

`SecretArn` 값을 제공하지 않는 경우 기본 인증 자격 증명을 제공하는 데 사용됩니다.

필드

- `Username` – Custom string pattern #37과(와) 일치하는 512바이트 이하 길이의 UTF-8 문자열입니다.

  데이터 소스에 연결할 사용자 이름.
- `Password` – Custom string pattern #33과(와) 일치하는 512바이트 이하 길이의 UTF-8 문자열입니다.

  데이터 소스에 연결할 암호.

OAuth2Credentials 구조

인증 유형이 OAuth2 인증일 때 사용되는 자격 증명입니다.

필드

- `UserManagedClientApplicationClientSecret` – Custom string pattern #38과(와) 일치하는 512바이트 이하 길이의 UTF-8 문자열입니다.

  클라이언트 애플리케이션이 사용자 관리형인 경우 클라이언트 애플리케이션 클라이언트 보안 암호.
- `AccessToken` – Custom string pattern #38과(와) 일치하는 4,096바이트 이하 길이의 UTF-8 문자열입니다.

  인증 유형이 OAuth2일 때 사용되는 액세스 토큰.
- `RefreshToken` – Custom string pattern #38과(와) 일치하는 4,096바이트 이하 길이의 UTF-8 문자열입니다.

  인증 유형이 OAuth2일 때 사용되는 새로 고침 토큰.
- `JwtToken` – Custom string pattern #39과(와) 일치하는 8,000바이트 이하 길이의 UTF-8 문자열입니다.

인증 유형이 OAuth2일 때 사용되는 JSON 웹 토큰(JWT).

운영

- CreateConnection 작업(Python: create_connection)
- DeleteConnection 작업(Python: delete_connection)
- GetConnection 작업(Python: get_connection)
- GetConnections 작업(Python: get_connections)
- UpdateConnection 작업(Python: update_connection)
- TestConnection 작업(Python: test_connection)
- BatchDeleteConnection 작업(Python: batch_delete_connection)

CreateConnection 작업(Python: create_connection)

데이터 카탈로그에서 연결 정의를 생성합니다.

페더레이션 리소스를 생성하는 데 사용되는 연결에는 IAM `glue:PassConnection` 권한이 필요합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결을 생성할 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `ConnectionInput` – 필수(Required): ConnectionInput 객체입니다.

  생성할 연결을 정의하는 `ConnectionInput` 객체입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  연결에 할당하는 태그입니다.

응답

- `CreateConnectionStatus` – UTF-8 문자열입니다(유효한 값: `READY` | `IN_PROGRESS` | `FAILED`).

  연결 생성 요청의 상태입니다. VPC를 통해 토큰을 교환하는 OAuth 연결을 생성하는 경우와 같이, 특정 인증 유형의 경우 요청에 다소 시간이 걸릴 수 있습니다.

오류

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

DeleteConnection 작업(Python: delete_connection)

데이터 카탈로그에서 연결을 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 지속될 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `ConnectionName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 연결의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`

- OperationTimeoutException

GetConnection 작업(Python: get_connection)

데이터 카탈로그에서 연결 정의를 검색합니다.

요청

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 지속될 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색할 연결 정의 이름입니다.
- HidePassword – 부울입니다.

  암호를 반환하지 않고 연결 메타데이터를 검색할 수 있습니다. 예를 들어 AWS Glue 콘솔은 이 플래그를 사용하여 연결을 검색하되 암호를 표시하지 않습니다. 호출자에게 AWS KMS 키를 사용하여 암호를 복호화할 권한이 없지만 연결 속성의 나머지에 액세스할 권한이 있는 경우 이 파라미터를 설정합니다.
- ApplyOverrideForComputeEnvironment – UTF-8 문자열입니다(유효한 값: SPARK | ATHENA | PYTHON).

  여러 서비스에서 사용할 수 있는 연결의 경우, 지정된 컴퓨팅 환경에 대한 반환 속성을 지정합니다.

응답

- Connection – 연결 객체입니다.

  요청한 연결 정의입니다.

오류

- EntityNotFoundException
- OperationTimeoutException

- `InvalidInputException`
- `GlueEncryptionException`

GetConnections 작업(Python: get_connections)

데이터 카탈로그에서 연결 정의의 목록을 검색합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 지속될 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `Filter` – GetConnectionsFilter 객체입니다.

  반환되는 연결을 관리하는 필터입니다.

- `HidePassword` – 부울입니다.

  암호를 반환하지 않고 연결 메타데이터를 검색할 수 있습니다. 예를 들어 AWS Glue 콘솔은 이 플래그를 사용하여 연결을 검색하되 암호를 표시하지 않습니다. 호출자에게 AWS KMS 키를 사용하여 암호를 복호화할 권한이 없지만 연결 속성의 나머지에 액세스할 권한이 있는 경우 이 파라미터를 설정합니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  한 번의 응답으로 반환될 최대 결과 수입니다.

응답

- `ConnectionList` – 연결 객체의 배열입니다.

  요청한 연결 정의의 목록입니다.

- `NextToken` – UTF-8 문자열입니다.

  반환된 연결 목록이 필터링된 연결의 마지막을 포함하지 않는 경우의 연속 토큰입니다.

오류

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

UpdateConnection 작업(Python: update_connection)

데이터 카탈로그에서 연결 정의를 업데이트합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 지속될 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트할 연결 정의 이름입니다.
- `ConnectionInput` – 필수(Required): ConnectionInput 객체입니다.

  질의에 따른 연결을 재정의하는 `ConnectionInput` 객체입니다.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

TestConnection 작업(Python: test_connection)

서비스에 대한 연결을 테스트하여 사용자가 제공하는 서비스 자격 증명을 검증합니다.

기존 연결 이름을 제공하거나, 기존 연결 이외의 연결을 테스트하는 경우 `TestConnectionInput`을 제공할 수 있습니다. 둘 다 동시에 제공하면 오류가 발생합니다.

작업이 성공하면 서비스가 HTTP 200 응답을 반송합니다.

요청

- `ConnectionName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  선택 사항. 테스트에 대한 연결의 이름입니다. 이름만 제공하면 이 작업은 해당 연결을 가져와 테스트에 사용합니다.
- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 상주하는 카탈로그 ID.
- `TestConnectionInput` – TestConnectionInput 객체입니다.

  서비스에 대한 연결 테스트를 지정할 때 사용되는 구조입니다.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`
- `FederationSourceException`
- `AccessDeniedException`
- `EntityNotFoundException`

- `ConflictException`
- `InternalServiceException`

BatchDeleteConnection 작업(Python: batch_delete_connection)

데이터 카탈로그에서 연결 정의의 목록을 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 지속될 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `ConnectionNameList` – 필수(Required): 25개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  삭제할 연결 이름의 목록입니다.

응답

- `Succeeded` – UTF-8 문자열의 배열입니다.

  성공적으로 삭제된 연결 정의 이름의 목록입니다.
- `Errors` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 ErrorDetail 객체입니다.

  오류 세부 정보에서 성공적으로 삭제된 연결 이름의 맵입니다.

오류

- `InternalServiceException`
- `OperationTimeoutException`

연결 유형 API

연결 유형 API는 연결 유형 설명과 관련된 AWS Glue API를 설명합니다.

연결 관리 API

- DescribeConnectionType 작업(Python: describe_connection_type)
- ListConnectionTypes 작업(Python: list_connection_types)
- ConnectionTypeBrief 구조
- ConnectionTypeVariant 구조

DescribeConnectionType 작업(Python: describe_connection_type)

`DescribeConnectionType` API는 AWS Glue에 지정된 연결 유형에 대해 지원되는 옵션에 대한 전체 세부 정보를 제공합니다.

요청

- `ConnectionType` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  설명할 연결 유형의 이름입니다.

응답

- `ConnectionType` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결 유형의 이름입니다.
- `Description` – UTF-8 문자열(1,024바이트 이하).

  연결 유형에 대한 설명입니다.
- `Capabilities` – 기능 객체입니다.

  지원되는 인증 유형, 데이터 인터페이스 유형(컴퓨팅 환경), 커넥터의 데이터 작업입니다.
- `ConnectionProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  컴퓨팅 환경 전체에서 공통적인 연결 속성입니다.

- `ConnectionOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  `ConnectionInput.ConnectionProperties`에서 연결을 생성할 때 설정할 수 있는 속성을 반환합니다. `ConnectionOptions`는 데이터프레임에 전달되는 연결 옵션 맵의 Spark ETL 스크립트에서 설정할 수 있는 파라미터를 정의합니다.
- `AuthenticationConfiguration` – AuthConfiguration 객체입니다.

  연결에 사용되는 인증의 유형입니다.
- `ComputeEnvironmentConfigurations` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 ComputeEnvironmentConfiguration 객체입니다.

  연결에서 지원하는 컴퓨팅 환경입니다.
- `PhysicalConnectionRequirements` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  VPC, 서브넷, 보안 그룹 사양 등, 연결에 대한 물리적 요구 사항입니다.
- `AthenaConnectionProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  Athena 컴퓨팅 환경과 관련된 연결 속성입니다.
- `PythonConnectionProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  Python 컴퓨팅 환경과 관련된 연결 속성입니다.
- `SparkConnectionProperties` – 키-값 페어의 맵 배열입니다.

각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

각 값은 속성 객체입니다.

Spark 컴퓨팅 환경과 관련된 연결 속성입니다.

오류

- `ValidationException`
- `InvalidInputException`
- `InternalServiceException`

ListConnectionTypes 작업(Python: list_connection_types)

`ListConnectionTypes` API는 AWS Glue에서 사용 가능한 연결 유형을 학습하는 검색 메커니즘을 제공합니다. 응답에는 각 연결 유형에 지원되는 항목에 대한 상위 수준 세부 정보가 포함된 연결 유형 목록이 포함되어 있습니다. 나열된 연결 유형은 `CreateConnection` API의 `ConnectionType` 값에 지원되는 옵션 세트입니다.

요청

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.
- `NextToken` – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `ConnectionTypes` – ConnectionTypeBrief 객체의 배열입니다.

  지원되는 연결 유형에 대한 간략한 정보가 포함된 `ConnectionTypeBrief` 객체의 목록입니다.
- `NextToken` – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  현재 list 세그먼트가 마지막이 아닌 경우의 연속 토큰입니다.

오류

- InternalServiceException

ConnectionTypeBrief 구조

ListConnectionTypes API에서 반환하는 지원되는 연결 유형에 대한 간략한 정보입니다.

필드

- ConnectionType – UTF-8 문자열(유효한 값: JDBC | SFTP | MONGODB | KAFKA | NETWORK | MARKETPLACE | CUSTOM | SALESFORCE | VIEW_VALIDATION_REDSHIFT | VIEW_VALIDATION_ATHENA | GOOGLEADS | GOOGLESHEETS | GOOGLEANALYTICS4 | SERVICENOW | MARKETO | SAPODATA | ZENDESK | JIRACLOUD | NETSUITEERP | HUBSPOT | FACEBOOKADS | INSTAGRAMADS | ZOHOCRM | SALESFORCEPARDOT | SALESFORCEMARKETINGCLOUD | SLACK | STRIPE | INTERCOM | SNAPCHATADS).

  연결 유형의 이름입니다.
- DisplayName – 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 콘솔에 표시되는 연결 유형의 이름이며 인간이 읽을 수 있는 이름입니다.
- Vendor – 1~128바이트 길이의 UTF-8 문자열입니다.

  이 연결 유형을 생성 또는 유지 관리하는 공급업체 또는 공급자의 이름입니다.
- Description – UTF-8 문자열(1,024바이트 이하).

  연결 유형에 대한 설명입니다.
- Categories – UTF-8 문자열의 배열입니다.

  이 연결 유형이 속한 범주 목록입니다. 범주는 사용자가 사용 사례에 기반하여 적절한 연결 유형을 필터링하고 찾는 데 도움이 됩니다.
- Capabilities – 기능 객체입니다.

  지원되는 인증 유형, 데이터 인터페이스 유형(컴퓨팅 환경), 커넥터의 데이터 작업입니다.
- LogoUrl – UTF-8 문자열입니다.

  연결 유형과 연결된 로고의 URL입니다.
- ConnectionTypeVariants – ConnectionTypeVariant 객체의 배열입니다.

이 연결 유형에 사용할 수 있는 변형 목록입니다. 각 변형은 동일한 일반 연결 유형의 특정 사용 사례 또는 구현에 대한 특별 구성을 제공할 수 있습니다.

ConnectionTypeVariant 구조

AWS Glue 데이터 카탈로그에서 연결 유형의 변형을 나타냅니다. 연결 유형 변형은 동일한 일반 연결 유형의 다양한 구현을 위한 특정 구성 및 동작을 제공합니다.

필드

- `ConnectionTypeVariantName` – 1~128바이트 길이의 UTF-8 문자열입니다.

  연결 유형 변형의 고유 식별자입니다. 이 이름은 내부적으로 연결 유형의 특정 변형을 식별하는 데 사용됩니다.
- `DisplayName` – 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 콘솔에 표시되는 연결 유형 변형의 이름이며 인간이 읽을 수 있는 이름입니다.
- `Description` – UTF-8 문자열(1,024바이트 이하).

  용도, 사용 사례 및 특정 구성 요구 사항 등 연결 유형 변형에 대한 자세한 설명입니다.
- `LogoUrl` – UTF-8 문자열입니다.

  연결 유형 변형과 연결된 로고의 URL입니다.

데이터 유형

- Validation 구조
- AuthConfiguration 구조
- Capabilities 구조
- Property 구조
- AllowedValue 구조
- ComputeEnvironmentConfiguration 구조

Validation 구조

연결 속성에서 검증이 수행되는 방법을 정의합니다.

필드

- `ValidationType` – 필수: UTF-8 문자열입니다(유효한 값: `REGEX` | `RANGE`).

  수행할 검증 유형(예: `REGEX`)입니다.
- `Patterns` – UTF-8 문자열의 배열입니다.

  검증에 적용되는 패턴의 목록입니다.
- `Description` – 필수: 1~1,024바이트 길이의 UTF-8 문자열입니다.

  검증에 대한 설명입니다.
- `MaxLength` - 숫자(정수)입니다.

  문자열 연결 속성의 최대 길이입니다.
- `Maximum` - 숫자(정수)입니다.

  검증의 `RANGE` 유형을 지정할 때의 최댓값입니다.
- `Minimum` - 숫자(정수)입니다.

  검증의 `RANGE` 유형을 지정할 때의 최솟값입니다.

AuthConfiguration 구조

`DescribeConnectionType` API에서 반환되는 연결에 대한 인증 구성입니다.

필드

- `AuthenticationType` – 필수: 속성 객체입니다.

  연결의 인증 유형입니다.
- `SecretArn` – 속성 객체입니다.

  Secrets Manager의 Amazon 리소스 이름(ARN)입니다.
- `OAuth2Properties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  OAuth2 속성의 키-값 페어 맵입니다. 각 값은 `Property` 객체입니다.

- `BasicAuthenticationProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  OAuth2 속성의 키-값 페어 맵입니다. 각 값은 `Property` 객체입니다.
- `CustomAuthenticationProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  사용자 지정 인증 속성의 키-값 페어 맵입니다. 각 값은 `Property` 객체입니다.

Capabilities 구조

`DescribeConnectionType` API에서 반환하는 지원되는 인증 유형을 지정합니다.

필드

- `SupportedAuthenticationTypes` – 필수: UTF-8 문자열의 배열입니다.

  지원되는 인증 유형의 목록입니다.
- `SupportedDataOperations` – 필수: UTF-8 문자열의 배열입니다.

  지원되는 데이터 작업의 목록입니다.
- `SupportedComputeEnvironments` – 필수: UTF-8 문자열의 배열입니다.

  지원되는 컴퓨팅 환경의 목록입니다.

Property 구조

컴퓨팅 환경의 연결 유형을 정의하는 객체입니다.

필드

- `Name` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  속성의 이름입니다.
- `Description` – 필수: UTF-8 문자열입니다(1,024바이트 이하).

속성에 대한 설명입니다.

- `Required` – 필수(Required): 부울.

  속성이 필요한지 여부를 나타냅니다.

- `DefaultValue` – UTF-8 문자열입니다.

  속성의 기본값입니다.

- `PropertyTypes` – 필수: UTF-8 문자열의 배열입니다.

  속성의 유형을 설명합니다.

- `AllowedValues` – AllowedValue 객체의 배열입니다.

  속성에 허용되는 값을 나타내는 `AllowedValue` 객체의 목록입니다.

- `DataOperationScopes` – UTF-8 문자열의 배열입니다.

  속성에 적용할 수 있는 데이터 작업을 나타냅니다.

AllowedValue 구조

속성에 허용되는 값을 나타내는 객체입니다.

필드

- `Description` – UTF-8 문자열(1,024바이트 이하).

  허용되는 값에 대한 설명입니다.

- `Value` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  속성에 허용되는 값입니다.

ComputeEnvironmentConfiguration 구조

`DescribeConnectionType` API에서 반환하는 컴퓨팅 환경(예: Spark, Python 또는 Athena)에 대한 구성이 포함되는 객체입니다.

필드

- `Name` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

컴퓨팅 환경 구성의 이름입니다.

- `Description` – 필수: UTF-8 문자열입니다(1,024바이트 이하).

  컴퓨팅 환경에 대한 설명입니다.

- `ComputeEnvironment` – 필수: UTF-8 문자열입니다(유효한 값: `SPARK` | `ATHENA` | `PYTHON`).

  컴퓨팅 환경의 유형입니다.

- `SupportedAuthenticationTypes` – 필수: UTF-8 문자열의 배열입니다.

  컴퓨팅 환경에 지원되는 인증 유형입니다.

- `ConnectionOptions` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 속성 객체입니다.

  컴퓨팅 환경의 연결 옵션으로 사용되는 파라미터입니다.

- `ConnectionPropertyNameOverrides` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~128바이트인 UTF-8 문자열입니다.

  연결 속성 이름은 컴퓨팅 환경에 대해 재정의됩니다.

- `ConnectionOptionNameOverrides` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 길이가 1~128바이트인 UTF-8 문자열입니다.

  연결 옵션 이름은 컴퓨팅 환경에 대해 재정의됩니다.

- `ConnectionPropertiesRequiredOverrides` – 필수: UTF-8 문자열의 배열입니다.

  컴퓨팅 환경에 대한 재정의로서 요구되는 연결 속성입니다.

- `PhysicalConnectionPropertiesRequired` – 부울입니다.

  `PhysicalConnectionProperties`가 컴퓨팅 환경에 필요한지 여부를 나타냅니다.

연결 메타데이터 및 미리 보기 API

다음 연결 API는 연결 메타데이터를 설명하는 작업에 대해 설명합니다.

데이터 타입

- Entity 구조
- Field 구조

Entity 구조

지정된 ConnectionType에서 지원하는 개체입니다.

필드

- EntityName – UTF-8 문자열입니다.

  개체의 이름입니다.
- Label – UTF-8 문자열입니다.

  엔터티에 사용되는 레이블입니다.
- IsParentEntity – 부울입니다.

  나열할 수 있는 하위 객체가 있는지 확인하는 데 도움이 되는 부울 값입니다.
- Description – UTF-8 문자열입니다.

  엔터티에 대한 설명입니다.
- Category – UTF-8 문자열입니다.

  응답에 있는 엔터티의 유형입니다. 이 값은 소스 연결에 따라 다릅니다. 예를 들어 Salesforce의 경우 SObjects이고 Amazon Redshift와 같은 소스의 경우 databases, schemas 또는 tables입니다.
- CustomProperties – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  커넥터가 엔터티에 대해 반환할 수 있는 키의 선택적 맵입니다.

Field 구조

Field 객체에는 커넥터의 필드와 연결된 다양한 속성에 대한 정보가 있습니다.

필드

- FieldName – UTF-8 문자열입니다.

  필드의 고유 식별자입니다.
- Label – UTF-8 문자열입니다.

  필드에 사용되는 읽기 가능한 레이블입니다.
- Description – UTF-8 문자열입니다.

  필드에 대한 설명입니다.
- FieldType – UTF-8 문자열입니다(유효한 값: INT | SMALLINT | BIGINT | FLOAT | LONG | DATE | BOOLEAN | MAP | ARRAY | STRING | TIMESTAMP | DECIMAL | BYTE | SHORT | DOUBLE | STRUCT).

  필드의 데이터 유형입니다.
- IsPrimaryKey – 부울입니다.

  이 필드를 지정된 엔터티의 기본 키로 사용할 수 있는지 여부를 나타냅니다.
- IsNullable – 부울입니다.

  이 필드에 null 값이 허용되는지 여부를 나타냅니다.
- IsRetrievable – 부울입니다.

  SQL 쿼리의 Select 절에 이 필드를 추가할 수 있는지 또는 이 필드를 검색할 수 있는지 여부를 나타냅니다.
- IsFilterable – 부울입니다.

  데이터를 쿼리할 때 SQL 문의 filter 절(WHERE 절)에 이 필드를 사용할 수 있는지 여부를 나타냅니다.
- IsPartitionable – 부울입니다.

  SaaS에 대한 쿼리를 분할하는 데 지정된 필드를 사용할 수 있는지 여부를 나타냅니다.
- IsCreateable – 부울입니다.

  이 필드를 대상 쓰기의 일부로 생성할 수 있는지 여부를 나타냅니다.
- IsUpdateable – 부울입니다.

이 필드를 대상 쓰기의 일부로 업데이트할 수 있는지 여부를 나타냅니다.

- `IsUpsertable` – 부울입니다.

  이 필드를 대상 쓰기의 일부로 삽입할 수 있는지 여부를 나타냅니다.

- `IsDefaultOnCreate` – 부울입니다.

  타임스탬프에서 생성되는 객체와 같은 객체가 생성될 때 이 필드가 자동으로 채워지는지 여부를 나타냅니다.

- `SupportedValues` – UTF-8 문자열의 배열입니다.

  필드의 지원되는 값 목록입니다.

- `SupportedFilterOperators` – UTF-8 문자열의 배열입니다.

  이 필드에 대한 지원 필터 연산자를 나타냅니다.

- `ParentField` – UTF-8 문자열입니다.

  중첩된 필드의 상위 필드 이름.

- `NativeDataType` – UTF-8 문자열입니다.

  Salesforce의 "picklist" 또는 "textarea"와 같은 SaaS API에서 반환되는 데이터 유형.

- `CustomProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  반환되는 키의 선택적 맵입니다.

운영

- ListEntities 작업(Python: list_entities)
- DescribeEntity 작업(Python: describe_entity)
- GetEntityRecords 작업(Python: get_entity_records)

ListEntities 작업(Python: list_entities)

연결 유형에서 지원하는 사용 가능한 엔터티를 반환합니다.

요청

- `ConnectionName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  모든 연결 유형을 쿼리하는 데 필요한 자격 증명이 있는 연결의 이름입니다.
- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 포함된 카탈로그의 카탈로그 ID입니다. 이 값은 null일 수 있습니다. 기본적으로 AWS 계정 ID는 카탈로그 ID입니다.
- `ParentEntityName` – UTF-8 문자열입니다.

  하위 항목을 나열하려는 상위 엔터티의 이름입니다. 이 파라미터는 하위 엔터티를 나열하기 위해 엔터티의 정규화된 경로를 사용합니다.
- `NextToken` – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰을 포함합니다.
- `DataStoreApiVersion` – Custom string pattern #23과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  SaaS 커넥터의 API 버전입니다.

응답

- `Entities` – 개체 객체의 배열입니다.

  `Entity` 객체의 목록.
- `NextToken` – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  현재 세그먼트가 마지막이 아닌 경우 연속 토큰이 존재합니다.

오류

- `EntityNotFoundException`
- `OperationTimeoutException`

- InvalidInputException
- GlueEncryptionException
- ValidationException
- FederationSourceException
- AccessDeniedException

DescribeEntity 작업(Python: describe_entity)

선택한 엔터티의 각 필드의 데이터 모델에 대한 설명과 함께 연결 유형에 사용되는 엔터티에 대한 세부 정보를 제공합니다.

엔터티를 구성하는 모든 필드가 응답에 포함됩니다.

요청

- ConnectionName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결 유형 자격 증명이 포함된 연결의 이름입니다.
- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 포함된 카탈로그의 카탈로그 ID입니다. 이 값은 null일 수 있습니다. 기본적으로 AWS 계정 ID는 카탈로그 ID입니다.
- EntityName – 필수: UTF-8 문자열입니다.

  연결 유형에서 설명할 엔터티의 이름입니다.
- NextToken – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰을 포함합니다.
- DataStoreApiVersion – Custom string pattern #23과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  데이터 스토어에 사용되는 API의 버전입니다.

응답

- `Fields` – 필드 객체의 배열입니다.

  해당 커넥터 엔터티의 필드를 설명합니다. `Field` 객체의 목록입니다. `Field`는 데이터베이스의 열과 매우 유사합니다. `Field` 객체에는 커넥터의 필드와 연결된 다양한 속성에 대한 정보가 있습니다.
- `NextToken` – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  현재 세그먼트가 마지막이 아닌 경우 연속 토큰이 존재합니다.

오류

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`
- `ValidationException`
- `FederationSourceException`
- `AccessDeniedException`

GetEntityRecords 작업(Python: get_entity_records)

이 API는 지정된 연결 유형 또는 기본 Amazon S3 기반 AWS Glue Data Catalog에서 미리 보기 데이터를 쿼리하는 데 사용됩니다.

레코드를 JSON Blob의 배열로 반환합니다. 각 레코드는 `DescribeEntity` API에서 정의한 필드 유형에 따라 Jackson JsonNode를 사용하여 형식이 지정됩니다.

Spark 커넥터는 `DescribeEntity` API와 동일한 데이터 형식 매핑에 따라 스키마를 생성합니다. Spark 커넥터는 행을 반환할 때 데이터를 스키마와 일치하는 적절한 데이터 유형으로 변환합니다.

요청

- `ConnectionName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  연결 유형 자격 증명이 포함된 연결의 이름입니다.

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  연결이 포함된 카탈로그의 카탈로그 ID입니다. 이 값은 null일 수 있습니다. 기본적으로 AWS 계정 ID는 카탈로그 ID입니다.

- `EntityName` – 필수: UTF-8 문자열입니다.

  지정된 연결 유형에서 미리 보기 데이터를 쿼리하려는 엔터티의 이름입니다.

- `NextToken` – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰을 포함합니다.

- `DataStoreApiVersion` – Custom string pattern #23과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  SaaS 커넥터의 API 버전입니다.

- `ConnectionOptions` – 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #18과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  각 값은 Custom string pattern #17과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  데이터를 쿼리하는 데 필요한 커넥터 옵션입니다.

- `FilterPredicate` – UTF-8 문자열입니다(1~100,000바이트).

  쿼리 요청에 적용할 수 있는 필터 조건자입니다.

- `Limit` – 필수: 1~1,000의 숫자(long)입니다.

  요청으로 가져오는 레코드 수를 제한합니다.

- `OrderBy` – UTF-8 문자열입니다.

  응답 미리 보기 데이터의 순서를 지정하는 파라미터입니다.

- `SelectedFields` – UTF-8 문자열의 배열입니다(1~1,000개의 문자열).

  미리 보기 데이터의 일부로 가져오려는 필드 목록입니다.

응답

- `Records` - 구조의 배열입니다.

  요청한 객체의 목록입니다.

- `NextToken` – Custom string pattern #11과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  현재 세그먼트가 마지막이 아닌 경우 연속 토큰이 존재합니다.

오류

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`
- `ValidationException`
- `FederationSourceException`
- `AccessDeniedException`

사용자 정의 함수 API

사용자 지정 함수 API는 함수로 작업할 때 사용되는 AWS Glue 데이터 유형 및 작업에 대해 설명합니다.

데이터 타입

- UserDefinedFunction 구조
- UserDefinedFunctionInput 구조

UserDefinedFunction 구조

Hive 사용자 정의 함수(UDF) 정의와 동등한 수준을 보여줍니다.

필드

- FunctionName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수의 이름입니다.

- DatabaseName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수를 포함하는 카탈로그 데이터베이스의 이름입니다.

- ClassName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수 코드가 포함된 Java 클래스입니다.

- OwnerName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수의 소유자입니다.

- OwnerType – UTF-8 문자열입니다(유효 값: USER | ROLE | GROUP).

  소유자 유형입니다.

- CreateTime – 타임스탬프입니다.

  함수가 생성된 시간.

- ResourceUris – ResourceUri 객체의 배열이며 구조는 1,000개 이하입니다.

  함수 리소스 URI

- CatalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  함수가 있는 Data Catalog의 ID입니다.

UserDefinedFunctionInput 구조

사용자 정의 함수를 생성하거나 업데이트할 때 사용되는 구조입니다.

필드

- FunctionName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

함수의 이름입니다.

- `ClassName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수 코드가 포함된 Java 클래스입니다.

- `OwnerName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수의 소유자입니다.

- `OwnerType` – UTF-8 문자열입니다(유효 값: `USER` | `ROLE` | `GROUP`).

  소유자 유형입니다.

- `ResourceUris` – ResourceUri 객체의 배열이며 구조는 1,000개 이하입니다.

  함수 리소스 URI

운영

- CreateUserDefinedFunction 작업(Python: create_user_defined_function)
- UpdateUserDefinedFunction 작업 (Python: update_user_defined_function)
- DeleteUserDefinedFunction 작업 (Python: delete_user_defined_function)
- GetUserDefinedFunction 작업 (Python: get_user_defined_function)
- GetUserDefinedFunctions 작업 (Python: get_user_defined_functions)

CreateUserDefinedFunction 작업(Python: create_user_defined_function)

데이터 카탈로그에서 새로운 함수 정의를 생성합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  함수를 생성할 데이터 카탈로그의 ID입니다. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

함수가 생성되는 카탈로그 데이터베이스의 이름입니다.

- `FunctionInput` – 필수(Required): UserDefinedFunctionInput 객체입니다.

  `FunctionInput` 객체는 데이터 카탈로그에 생성할 함수를 정의합니다.

응답

- 무응답 파라미터.

Errors

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

UpdateUserDefinedFunction 작업 (Python: update_user_defined_function)

데이터 카탈로그에서 기존 함수를 업데이트합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  업데이트되는 함수가 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트되는 함수가 존재하는 카탈로그 데이터베이스 이름

- `FunctionName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

함수의 이름입니다.

- `FunctionInput` – 필수(Required): UserDefinedFunctionInput 객체입니다.

  `FunctionInput` 객체는 데이터 카탈로그의 함수를 재정의합니다.

응답

- 무응답 파라미터.

Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeleteUserDefinedFunction 작업 (Python: delete_user_defined_function)

데이터 카탈로그에서 기존 함수를 삭제합니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  삭제되는 함수가 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수가 존재하는 카탈로그 데이터베이스 이름

- `FunctionName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제된 함수 정의 이름입니다.

응답

- 무응답 파라미터.

Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetUserDefinedFunction 작업 (Python: get_user_defined_function)

데이터 카탈로그에서 지정된 함수를 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  가져올 함수가 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수가 존재하는 카탈로그 데이터베이스 이름

- `FunctionName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수의 이름입니다.

응답

- `UserDefinedFunction` – UserDefinedFunction 객체입니다.

  요청한 함수 정의입니다.

Errors

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

GetUserDefinedFunctions 작업 (Python: get_user_defined_functions)

데이터 카탈로그에서 다양한 함수 정의를 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  가져올 함수가 존재하는 데이터 카탈로그 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  함수가 존재하는 카탈로그 데이터베이스 이름 아무 것도 제공되지 않으면 카탈로그에 있는 모든 데이터베이스의 함수가 반환됩니다.
- `Pattern` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  반환된 함수 정의를 필터링하는 조건부 함수 이름 패턴 문자열입니다.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.
- `MaxResults` – 1~100의 숫자(정수)입니다.

  한 번의 응답으로 반환될 최대 함수 수입니다.

응답

- `UserDefinedFunctions` – UserDefinedFunction 객체의 배열입니다.

요청한 함수 정의의 목록입니다.

- NextToken – UTF-8 문자열입니다.

  반환된 함수 목록이 마지막으로 요청된 함수가 아닌 경우의 연속 토큰입니다.

Errors

- EntityNotFoundException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException
- GlueEncryptionException

Athena 카탈로그를 AWS Glue로 가져오기

마이그레이션 API는 Athena 데이터 카탈로그를 AWS Glue로 마이그레이션하는 것과 관련된 AWS Glue 데이터 유형 및 작업에 대해 설명합니다.

데이터 타입

- CatalogImportStatus 구조

CatalogImportStatus 구조

이송 상태 정보를 포함한 구조.

필드

- ImportCompleted – 부울입니다.

  마이그레이션이 완료되면 True이고 그렇지 않으면 False입니다.
- ImportTime – 타임스탬프입니다.

  이송이 시작된 시간.
- ImportedBy – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이송을 시작한 사람의 이름입니다.

운영

- ImportCatalogToGlue 작업(Python: import_catalog_to_glue)
- GetCatalogImportStatus 작업(Python: get_catalog_import_status)

ImportCatalogToGlue 작업(Python: import_catalog_to_glue)

기존 Amazon Athena Data Catalog를 AWS Glue로 가져옵니다.

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  가져올 객체의 ID입니다. 현재는 AWS 계정 ID여야 합니다.

응답

- 무응답 파라미터.

Errors

- `InternalServiceException`
- `OperationTimeoutException`

GetCatalogImportStatus 작업(Python: get_catalog_import_status)

이송 작업 상태 검색

요청

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  이송할 카탈로그의 ID입니다. 현재는 AWS 계정 ID여야 합니다.

응답

- `ImportStatus` – CatalogImportStatus 객체입니다.

  지정된 카탈로그 이송 상태.

Errors

- `InternalServiceException`
- `OperationTimeoutException`

테이블 옵티마이저 API

테이블 옵티마이저 API는 컴팩션을 활성화하여 읽기 성능을 향상시키는 AWS Glue API를 설명합니다.

데이터 타입

- TableOptimizer 구조
- TableOptimizerConfiguration 구조
- TableOptimizerVpcConfiguration 구조
- CompactionConfiguration 구조
- IcebergCompactionConfiguration 구조
- TableOptimizerRun 구조
- BatchGetTableOptimizerEntry 구조
- BatchTableOptimizer 구조
- BatchGetTableOptimizerError 구조
- RetentionConfiguration 구조
- IcebergRetentionConfiguration 구조
- OrphanFileDeletionConfiguration 구조
- IcebergOrphanFileDeletionConfiguration 구조
- CompactionMetrics 구조
- RetentionMetrics 구조
- OrphanFileDeletionMetrics 구조
- IcebergCompactionMetrics 구조

- IcebergRetentionMetrics 구조
- IcebergOrphanFileDeletionMetrics 구조
- RunMetrics 구조

TableOptimizer 구조

테이블과 관련된 옵티마이저에 대한 세부 정보가 들어 있습니다.

필드

- `type` – UTF-8 문자열입니다(유효한 값: `compaction="COMPACTION"` | `retention="RETENTION"` | `orphan_file_deletion="ORPHAN_FILE_DELETION"`).

  테이블 옵티마이저 유형. 유효한 값은 다음과 같습니다.

  - `compaction`: 테이블 옵티마이저 압축 관리.
  - `retention`: 테이블 옵티마이저 스냅샷 보존 관리.
  - `orphan_file_deletion`: 테이블 옵티마이저 분리된 파일의 삭제 관리.
- `configuration` – TableOptimizerConfiguration 객체입니다.

  테이블 옵티마이저를 만들거나 업데이트할 때 지정된 `TableOptimizerConfiguration` 객체입니다.
- `lastRun` – TableOptimizerRun 객체입니다.

  테이블 옵티마이저의 마지막 실행을 나타내는 `TableOptimizerRun` 객체입니다.

TableOptimizerConfiguration 구조

테이블 옵티마이저의 구성에 대한 세부 정보가 들어 있습니다. 테이블 옵티마이저를 만들거나 업데이트할 때 이 구성을 전달합니다.

필드

- `roleArn` – Single-line string pattern과 일치하는 UTF-8 문자열입니다(20~2,048바이트).

  호출자가 전달하는 역할로, 호출자를 대신하여 최적기와 관련된 리소스를 업데이트할 수 있는 권한을 서비스에 부여합니다.
- `enabled` – 부울입니다.

테이블 최적화가 활성화되었는지 여부.

- `vpcConfiguration` – TableOptimizerVpcConfiguration 객체입니다.

  테이블 옵티마이저에 대한 구성을 나타내는 `TableOptimizerVpcConfiguration` 객체.

  이 구성은 고객 VPC에 있는 테이블에서 최적화를 수행하는 데 필요합니다.
- `compactionConfiguration` – CompactionConfiguration 객체입니다.

  압축 옵티마이저의 구성입니다. 이 구성은 쿼리 성능을 개선하고 스토리지 비용을 줄이기 위해 테이블의 데이터 파일이 압축되는 방식을 정의합니다.
- `retentionConfiguration` – RetentionConfiguration 객체입니다.

  스냅샷 보존 옵티마이저의 구성.
- `orphanFileDeletionConfiguration` – OrphanFileDeletionConfiguration 객체입니다.

  분리된 파일 삭제 옵티마이저의 구성.

TableOptimizerVpcConfiguration 구조

테이블 옵티마이저에 대한 VPC 구성을 설명하는 객체.

이 구성은 고객 VPC에 있는 테이블에서 최적화를 수행하는 데 필요합니다.

필드

- `glueConnectionName` – UTF-8 문자열입니다(최소 1바이트).

  테이블 옵티마이저에 대한 VPC에 사용되는 AWS Glue 연결의 이름.

CompactionConfiguration 구조

압축 옵티마이저의 구성입니다. 이 구성은 쿼리 성능을 개선하고 스토리지 비용을 줄이기 위해 테이블의 데이터 파일이 압축되는 방식을 정의합니다.

필드

- `icebergConfiguration` – IcebergCompactionConfiguration 객체입니다.

  Iceberg 압축 옵티마이저의 구성입니다.

IcebergCompactionConfiguration 구조

Iceberg 압축 옵티마이저의 구성입니다. 이 구성은 Iceberg 테이블의 데이터 파일 레이아웃을 최적화하기 위한 파라미터를 정의합니다.

필드

- `strategy` – UTF-8 문자열입니다(유효한 값: `binpack="BINPACK"` | `sort="SORT"` | `z-order="ZORDER"`).

  압축에 사용할 전략입니다. 유효한 값은 다음과 같습니다.

  - `binpack`: 작은 파일을 더 큰 파일로 결합합니다. 일반적으로 100MB 이상의 크기를 대상으로 하는 동시에 보류 중인 삭제를 적용합니다. 대부분의 사용 사례에 권장되는 압축 전략입니다.
  - `sort`: 압축 도중 계층적으로 정렬된 지정된 열을 기반으로 데이터를 구성하여 필터링된 작업에 대한 쿼리 성능을 개선합니다. 이 전략은 쿼리가 특정 열을 자주 필터링할 때 권장됩니다. 이 전략을 사용하려면 먼저 `sort_order` 테이블 속성을 사용하여 Iceberg 테이블 속성에서 정렬 순서를 정의해야 합니다.
  - `z-order`: 여러 속성을 정렬에 사용할 수 있는 단일 스칼라 값으로 블렌딩하여 데이터 구성을 최적화하므로 여러 차원에 걸쳐 효율적인 쿼리가 가능합니다. 이 전략은 여러 차원에서 동시에 데이터를 쿼리해야 하는 경우에 권장됩니다. 이 전략을 사용하려면 먼저 `sort_order` 테이블 속성을 사용하여 Iceberg 테이블 속성에서 정렬 순서를 정의해야 합니다.

  입력이 제공되지 않으면 기본값인 'binpack'이 사용됩니다.

TableOptimizerRun 구조

테이블 옵티마이저 실행에 대한 세부 정보가 들어 있습니다.

필드

- `eventType` – UTF-8 문자열입니다(유효한 값: `starting="STARTING"` | `completed="COMPLETED"` | `failed="FAILED"` | `in_progress="IN_PROGRESS"`).

  테이블 옵티마이저 실행 상태를 나타내는 이벤트 유형입니다.

- `startTimestamp` – 타임스탬프입니다.

  Lake Formation 내에서 압축 작업이 시작된 시점의 에포크 타임스탬프를 나타냅니다.

- `endTimestamp` – 타임스탬프입니다.

컴팩션 작업이 종료된 에포크 타임스탬프를 나타냅니다.

- `metrics` – RunMetrics 객체입니다.

  옵티마이저 실행에 대한 메트릭이 포함된 `RunMetrics` 객체입니다.

  이 멤버는 더 이상 사용되지 않습니다. 압축, 보존 및 분리된 파일 삭제에 대해서는 개별 지표 구성원을 참조하세요.

- `error` – UTF-8 문자열입니다.

  옵티마이저 실행 중에 발생한 오류입니다.

- `compactionMetrics` – CompactionMetrics 객체입니다.

  옵티마이저 실행에 대한 메트릭이 포함된 `CompactionMetrics` 객체입니다.

- `compactionStrategy` – UTF-8 문자열입니다(유효한 값: `binpack="BINPACK"` | `sort="SORT"` | `z-order="ZORDER"`).

  압축 실행에 사용되는 전략입니다. 압축 프로세스 중에 파일을 선택하고 결합하는 방법을 결정하는 데 적용된 알고리즘을 나타냅니다. 유효한 값은 다음과 같습니다.

  - `binpack`: 작은 파일을 더 큰 파일로 결합합니다. 일반적으로 100MB 이상의 크기를 대상으로 하는 동시에 보류 중인 삭제를 적용합니다. 대부분의 사용 사례에 권장되는 압축 전략입니다.
  - `sort`: 압축 도중 계층적으로 정렬된 지정된 열을 기반으로 데이터를 구성하여 필터링된 작업에 대한 쿼리 성능을 개선합니다. 이 전략은 쿼리가 특정 열을 자주 필터링할 때 권장됩니다. 이 전략을 사용하려면 먼저 `sort_order` 테이블 속성을 사용하여 Iceberg 테이블 속성에서 정렬 순서를 정의해야 합니다.
  - `z-order`: 여러 속성을 정렬에 사용할 수 있는 단일 스칼라 값으로 블렌딩하여 데이터 구성을 최적화하므로 여러 차원에 걸쳐 효율적인 쿼리가 가능합니다. 이 전략은 여러 차원에서 동시에 데이터를 쿼리해야 하는 경우에 권장됩니다. 이 전략을 사용하려면 먼저 `sort_order` 테이블 속성을 사용하여 Iceberg 테이블 속성에서 정렬 순서를 정의해야 합니다.

- `retentionMetrics` – RetentionMetrics 객체입니다.

  옵티마이저 실행에 대한 메트릭이 포함된 `RetentionMetrics` 객체입니다.

- `orphanFileDeletionMetrics` – OrphanFileDeletionMetrics 객체입니다.

  옵티마이저 실행에 대한 메트릭이 포함된 `OrphanFileDeletionMetrics` 객체입니다.

BatchGetTableOptimizerEntry 구조

BatchGetTableOptimizer 작업에서 검색할 테이블 옵티마이저를 나타냅니다.

필드

- catalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.
- databaseName – UTF-8 문자열입니다(최소 1바이트).

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.
- tableName – UTF-8 문자열입니다(최소 1바이트).

  테이블의 이름
- type – UTF-8 문자열입니다(유효한 값: compaction="COMPACTION" | retention="RETENTION" | orphan_file_deletion="ORPHAN_FILE_DELETION").

  테이블 옵티마이저 유형.

BatchTableOptimizer 구조

BatchGetTableOptimizer 작업에서 반환된 테이블 옵티마이저 중 하나에 대한 세부 정보가 들어 있습니다.

필드

- catalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.
- databaseName – UTF-8 문자열입니다(최소 1바이트).

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.
- tableName – UTF-8 문자열입니다(최소 1바이트).

  테이블의 이름
- tableOptimizer – TableOptimizer 객체입니다.

테이블 옵티마이저의 구성 및 마지막 실행에 대한 세부 정보가 포함된 TableOptimizer 객체입니다.

BatchGetTableOptimizerError 구조

BatchGetTableOptimizer 작업에서 반환된 오류 목록의 오류 중 하나에 대한 세부 정보가 들어 있습니다.

필드

- error – ErrorDetail 객체입니다.

  오류에 대한 코드 및 메시지 세부 정보가 포함된 ErrorDetail 객체입니다.

- catalogId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.

- databaseName – UTF-8 문자열입니다(최소 1바이트).

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.

- tableName – UTF-8 문자열입니다(최소 1바이트).

  테이블의 이름

- type – UTF-8 문자열입니다(유효한 값: compaction="COMPACTION" | retention="RETENTION" | orphan_file_deletion="ORPHAN_FILE_DELETION").

  테이블 옵티마이저 유형.

RetentionConfiguration 구조

스냅샷 보존 옵티마이저의 구성.

필드

- icebergConfiguration – IcebergRetentionConfiguration 객체입니다.

  Iceberg 스냅샷 보존 옵티마이저의 구성.

IcebergRetentionConfiguration 구조

Iceberg 스냅샷 보존 옵티마이저의 구성.

필드

- `snapshotRetentionPeriodInDays` - 숫자(정수)입니다.

  Iceberg 스냅샷을 보존하는 기간(일). 입력이 없으면 해당 Iceberg 테이블 구성 필드가 사용되거나 필드가 없는 경우 기본값 5가 사용됩니다.
- `numberOfSnapshotsToRetain` - 숫자(정수)입니다.

  보존 기간 내에 유지해야 하는 Iceberg 스냅샷의 수. 입력이 없으면 해당 Iceberg 테이블 구성 필드가 사용되거나 필드가 없는 경우 기본값 1이 사용됩니다.
- `cleanExpiredFiles` – 부울입니다.

  false로 설정하면 스냅샷은 테이블 메타데이터에서만 삭제되고, 그에 속한 데이터 및 메타데이터 파일은 삭제되지 않습니다.

OrphanFileDeletionConfiguration 구조

분리된 파일 삭제 옵티마이저의 구성.

필드

- `icebergConfiguration` – IcebergOrphanFileDeletionConfiguration 객체입니다.

  Iceberg 분리된 파일 삭제 옵티마이저의 구성.

IcebergOrphanFileDeletionConfiguration 구조

Iceberg 분리된 파일 삭제 옵티마이저의 구성.

필드

- `orphanFileRetentionPeriodInDays` - 숫자(정수)입니다.

  파일이 삭제되기 전까지 분리된 파일이 보존되어야 하는 기간(일). 입력이 없으면 기본값 3이 사용됩니다.

- `location` – UTF-8 문자열입니다.

  파일을 찾을 디렉터리를 지정합니다(기본값은 테이블 위치). 최상위 테이블 위치 대신 하위 디렉터리를 선택할 수 있습니다.

CompactionMetrics 구조

옵티마이저 실행에 대한 압축 지표가 포함된 구조.

필드

- `IcebergMetrics` – IcebergCompactionMetrics 객체입니다.

  옵티마이저 실행에 대한 Iceberg 압축 지표가 포함된 구조.

RetentionMetrics 구조

옵티마이저 실행에 대한 보존 지표가 포함된 구조.

필드

- `IcebergMetrics` – IcebergRetentionMetrics 객체입니다.

  옵티마이저 실행에 대한 Iceberg 보존 지표가 포함된 구조.

OrphanFileDeletionMetrics 구조

옵티마이저 실행에 대한 분리된 파일 삭제 지표가 포함된 구조.

필드

- `IcebergMetrics` – IcebergOrphanFileDeletionMetrics 객체입니다.

  옵티마이저 실행에 대한 Iceberg 분리된 파일 삭제 지표가 포함된 구조.

IcebergCompactionMetrics 구조

옵티마이저 실행에 대한 Iceberg 압축 지표.

필드

- DpuHours - 숫자(double)입니다.

  작업에 사용된 DPU 시간.
- NumberOfDpus - 숫자(정수)입니다.

  작업에서 사용하는 DPU의 수로, 가장 가까운 정수로 반올림됩니다.
- JobDurationInHour - 숫자(double)입니다.

  작업 기간(시간).

IcebergRetentionMetrics 구조

옵티마이저 실행에 대한 Iceberg 스냅샷 보존 지표.

필드

- DpuHours - 숫자(double)입니다.

  작업에 사용된 DPU 시간.
- NumberOfDpus - 숫자(정수)입니다.

  작업에서 사용하는 DPU의 수로, 가장 가까운 정수로 반올림됩니다.
- JobDurationInHour - 숫자(double)입니다.

  작업 기간(시간).

IcebergOrphanFileDeletionMetrics 구조

옵티마이저 실행에 대한 Iceberg 분리된 파일 삭제 지표.

필드

- DpuHours - 숫자(double)입니다.

  작업에 사용된 DPU 시간.
- NumberOfDpus - 숫자(정수)입니다.

  작업에서 사용하는 DPU의 수로, 가장 가까운 정수로 반올림됩니다.

- `JobDurationInHour` - 숫자(double)입니다.

  작업 기간(시간).

RunMetrics 구조

옵티마이저 실행에 대한 지표.

이 구조는 더 이상 사용되지 않습니다. 압축, 보존 및 분리된 파일 삭제에 대해서는 개별 지표 구성원을 참조하세요.

필드

- `NumberOfBytesCompacted` – UTF-8 문자열입니다.

  압축 작업 실행으로 제거된 바이트 수입니다.

- `NumberOfFilesCompacted` – UTF-8 문자열입니다.

  압축 작업 실행으로 제거된 파일 수입니다.

- `NumberOfDpus` – UTF-8 문자열입니다.

  작업에서 사용하는 DPU의 수로, 가장 가까운 정수로 반올림됩니다.

- `JobDurationInHour` – UTF-8 문자열입니다.

  작업 기간(시간).

운영

- GetTableOptimizer 작업 (Python: get_table_optimizer)
- BatchGetTableOptimizer 작업 (Python: batch_get_table_optimizer)
- ListTableOptimizerRuns 작업 (Python: list_table_optimizer_runs)
- CreateTableOptimizer 작업 (Python: create_table_optimizer)
- DeleteTableOptimizer 작업 (Python: delete_table_optimizer)
- UpdateTableOptimizer 작업 (Python: update_table_optimizer)

GetTableOptimizer 작업 (Python: get_table_optimizer)

지정된 테이블과 관련된 모든 옵티마이저의 구성을 반환합니다.

요청

- `CatalogId` – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.
- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름
- `Type` – 필수: UTF-8 문자열입니다(유효한 값: `compaction="COMPACTION"` | `retention="RETENTION"` | `orphan_file_deletion="ORPHAN_FILE_DELETION"`).

  테이블 옵티마이저 유형.

응답

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.
- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.
- `TableName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름
- `TableOptimizer` – TableOptimizer 객체입니다.

  지정된 테이블에 연결된 옵티마이저입니다.

오류

- EntityNotFoundException
- InvalidInputException
- AccessDeniedException
- InternalServiceException
- ThrottlingException

BatchGetTableOptimizer 작업 (Python: batch_get_table_optimizer)

지정된 테이블 옵티마이저의 구성을 반환합니다.

요청

- Entries – 필수(Required): BatchGetTableOptimizerEntry 객체의 배열입니다.

  검색할 테이블 옵티마이저를 지정하는 BatchGetTableOptimizerEntry 객체 목록입니다.

응답

- TableOptimizers – BatchTableOptimizer 객체의 배열입니다.

  BatchTableOptimizer 객체의 목록.
- Failures – BatchGetTableOptimizerError 객체의 배열입니다.

  작업의 오류 목록.

오류

- EntityNotFoundException
- InvalidInputException
- AccessDeniedException
- InternalServiceException
- ThrottlingException

ListTableOptimizerRuns 작업 (Python: list_table_optimizer_runs)

특정 테이블에 대한 이전 옵티마이저 실행 기록을 나열합니다.

요청

- `CatalogId` – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.
- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름
- `Type` – 필수: UTF-8 문자열입니다(유효한 값: `compaction="COMPACTION"` | `retention="RETENTION"` | `orphan_file_deletion="ORPHAN_FILE_DELETION"`).

  테이블 옵티마이저 유형.
- `MaxResults` - 숫자(정수)입니다.

  각 호출에서 반환되는 옵티마이저 실행의 최대 수입니다.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.
- `DatabaseName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.

- `TableName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름

- `NextToken` – UTF-8 문자열입니다.

  목록의 현재 세그먼트가 마지막이 아니면 반환된 옵티마이저 실행 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

- `TableOptimizerRuns` – TableOptimizerRun 객체의 배열입니다.

  테이블에 연결된 옵티마이저 실행 목록입니다.

오류

- `EntityNotFoundException`
- `AccessDeniedException`
- `InvalidInputException`
- `ValidationException`
- `InternalServiceException`
- `ThrottlingException`

CreateTableOptimizer 작업 (Python: create_table_optimizer)

특정 함수에 대한 새 테이블 옵티마이저를 생성합니다.

요청

- `CatalogId` – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

테이블의 이름

- `Type` – 필수: UTF-8 문자열입니다(유효한 값: `compaction="COMPACTION"` | `retention="RETENTION"` | `orphan_file_deletion="ORPHAN_FILE_DELETION"`).

  테이블 옵티마이저 유형.

- `TableOptimizerConfiguration` – 필수(Required): TableOptimizerConfiguration 객체입니다.

  테이블 옵티마이저의 구성을 나타내는 `TableOptimizerConfiguration` 객체입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `ValidationException`
- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `InternalServiceException`
- `ThrottlingException`

DeleteTableOptimizer 작업 (Python: delete_table_optimizer)

테이블의 옵티마이저 및 모든 관련 메타데이터를 삭제합니다. 최적화는 더 이상 테이블에서 수행되지 않습니다.

요청

- `CatalogId` – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.

- DatabaseName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 카탈로그의 데이터베이스 이름입니다.
- TableName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름
- Type – 필수: UTF-8 문자열입니다(유효한 값: compaction="COMPACTION" | retention="RETENTION" | orphan_file_deletion="ORPHAN_FILE_DELETION").

  테이블 옵티마이저 유형.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException
- AccessDeniedException
- InternalServiceException
- ThrottlingException

UpdateTableOptimizer 작업 (Python: update_table_optimizer)

기존 테이블 옵티마이저의 구성을 업데이트합니다.

요청

- CatalogId – 필수: Single-line string pattern과 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블의 카탈로그 ID.
- DatabaseName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

테이블이 있는 카탈로그의 데이터베이스 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름

- `Type` – 필수: UTF-8 문자열입니다(유효한 값: `compaction="COMPACTION"` | `retention="RETENTION"` | `orphan_file_deletion="ORPHAN_FILE_DELETION"`).

  테이블 옵티마이저 유형.

- `TableOptimizerConfiguration` – 필수(Required): TableOptimizerConfiguration 객체입니다.

  테이블 옵티마이저의 구성을 나타내는 `TableOptimizerConfiguration` 객체입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `ValidationException`
- `InternalServiceException`
- `ThrottlingException`
- `ConcurrentModificationException`

크롤러 및 분류자 API

크롤러 및 분류자 API는 AWS Glue 크롤러 및 분류자 데이터 유형에 대해 설명하며 크롤러 또는 분류자를 생성, 삭제, 업데이트 및 나열하기 위한 API를 포함합니다.

주제

- 분류자 API

- 크롤러 API
- 열 통계의 API
- 크롤러 스케줄러 API

분류자 API

분류자 API는 AWS Glue 분류자 데이터 유형에 대해 설명하며 분류자를 생성, 삭제, 업데이트 및 나열하기 위한 API를 포함합니다.

데이터 타입

- 분류자 구조
- GrokClassifier 구조
- XMLClassifier 구조
- JsonClassifier 구조
- CsvClassifier 구조
- CreateGrokClassifierRequest 구조
- UpdateGrokClassifierRequest 구조
- CreateXMLClassifierRequest 구조
- UpdateXMLClassifierRequest 구조
- CreateJsonClassifierRequest 구조
- UpdateJsonClassifierRequest 구조
- CreateCsvClassifierRequest 구조
- UpdateCsvClassifierRequest 구조

분류자 구조

분류자는 크롤링 작업 중에 시작됩니다. 분류자는 지정된 파일이 처리할 수 있는 형식인지 여부를 확인합니다. 처리할 수 있는 형식인 경우, 분류자는 해당 데이터 형식과 일치하는 `StructType` 객체의 형태로 스키마를 생성합니다.

AWS Glue가 제공하는 스탠더드 분류자를 사용하거나 자체 분류자를 작성하여 데이터 원본을 분류하고 분류자를 고려해 사용할 적절한 스키마를 지정합니다. 분류자는 `grok` 분류자이거나, `XML` 분류자이

거나, JSON 분류자이거나, Classifier 객체의 필드 중 하나에 지정된 사용자 지정 CSV 분류자일 수 있습니다.

필드

- GrokClassifier – GrokClassifier 객체입니다.

  grok을 사용하는 분류자입니다.
- XMLClassifier – XMLClassifier 객체입니다.

  XML 콘텐츠의 분류자입니다.
- JsonClassifier – JsonClassifier 객체입니다.

  JSON 콘텐츠의 분류자입니다.
- CsvClassifier – CsvClassifier 객체입니다.

  쉼표로 구분된 값(CSV)의 분류자입니다.

GrokClassifier 구조

grok 패턴을 사용하는 분류자입니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.
- Classification – 필수(Required): UTF-8 문자열입니다.

  Twitter, JSON, Omniture 로그, 등과 같이 분류자가 일치하는 데이터 양식의 분류자입니다.
- CreationTime – 타임스탬프입니다.

  이 분류자가 등록된 시간입니다.
- LastUpdated – 타임스탬프입니다.

  이 분류자가 마지막으로 업데이트된 시간입니다.
- Version - 숫자(정수)입니다.

이 분류자 버전

- `GrokPattern` – 필수(Required): A Logstash Grok string pattern과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  이 분류자에 의해 grok 패턴이 데이터 스토어로 적용됩니다. 자세한 내용은 사용자 지정 분류자 작성에서 기본 설정 패턴을 참조하십시오.
- `CustomPatterns` – URI address multi-line string pattern과(와) 일치하는 16,000바이트 이하 길이의 UTF-8 문자열입니다.

  이 분류자가 정의한 조건부 사용자 지정 grok 패턴입니다. 자세한 내용은 사용자 지정 분류자 작성에서 사용자 지정 패턴을 참조하십시오.

XMLClassifier 구조

`XML` 내용의 분류자

필드

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.
- `Classification` – 필수(Required): UTF-8 문자열입니다.

  분류자가 일치하는 데이터 양식의 분류자입니다.
- `CreationTime` – 타임스탬프입니다.

  이 분류자가 등록된 시간입니다.
- `LastUpdated` – 타임스탬프입니다.

  이 분류자가 마지막으로 업데이트된 시간입니다.
- `Version` - 숫자(정수)입니다.

  이 분류자 버전
- `RowTag` – UTF-8 문자열입니다.

  각 기록을 구문 분석된 XML 문서에 포함하는 요소를 설계하는 XML 태그입니다. 이 태그는 자기 닫기 요소(/>에 의해 닫힌 요소)를 식별할 수 없습니다. 요소가 닫는 태그를 통해 종료되는 한 구문 분

석된 속성만 포함하는 빈 행 요소입니다 (예를 들어, `<row item_a="A" item_b="B"></row>`는 괜찮지만 `<row item_a="A" item_b="B" />`는 괜찮지 않습니다).

JsonClassifier 구조

JSON 내용의 분류자

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.

- CreationTime – 타임스탬프입니다.

  이 분류자가 등록된 시간입니다.

- LastUpdated – 타임스탬프입니다.

  이 분류자가 마지막으로 업데이트된 시간입니다.

- Version - 숫자(정수)입니다.

  이 분류자 버전

- JsonPath – 필수(Required): UTF-8 문자열입니다.

  분류자가 분류할 JSON 데이터를 정의하는 JsonPath 문자열입니다. AWS Glue는 Writing JsonPath 사용자 정의 분류자 작성에 설명된 대로 JsonPath의 하위 집합을 지원합니다.

CsvClassifier 구조

사용자 지정 CSV 콘텐츠의 분류자입니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.

- CreationTime – 타임스탬프입니다.

이 분류자가 등록된 시간입니다.

- `LastUpdated` – 타임스탬프입니다.

  이 분류자가 마지막으로 업데이트된 시간입니다.

- `Version` - 숫자(정수)입니다.

  이 분류자 버전

- `Delimiter` – Custom string pattern #26과(와) 일치하는 1~1바이트 길이의 UTF-8 문자열입니다.

  행의 열 입력 항목 각각을 구분하는 것을 나타내기 위한 사용자 지정 기호입니다.

- `QuoteSymbol` – Custom string pattern #26과(와) 일치하는 1~1바이트 길이의 UTF-8 문자열입니다.

  단일 열 값에 내용을 결합하는 것을 나타내기 위한 사용자 지정 기호입니다. 열 구분 기호와 달라야 합니다.

- `ContainsHeader` – UTF-8 문자열입니다(유효 값: `UNKNOWN` | `PRESENT` | `ABSENT`).

  CSV 파일에 헤더가 포함되어 있는지 여부를 나타냅니다.

- `Header` – UTF-8 문자열의 배열입니다.

  열 이름을 나타내는 문자열 목록입니다.

- `DisableValueTrimming` – 부울입니다.

  열 값의 유형을 식별하기 전에 값을 트리밍하지 않도록 지정합니다. 기본값은 `true`입니다.

- `AllowSingleColumn` – 부울입니다.

  오직 하나의 열만 포함하는 파일을 처리할 수 있도록 합니다.

- `CustomDatatypeConfigured` – 부울입니다.

  사용자 지정 데이터 유형을 구성할 수 있습니다.

- `CustomDatatypes` – UTF-8 문자열의 배열입니다.

  사용자 지정 데이터 유형 목록에는 “바이너리”, “부울”, “날짜”, “십진수”, “더블”, “플로트”, “INT”, “롱”, “쇼트”, “문자열”, “타임스탬프” 등이 포함됩니다.

- `Serde` – UTF-8 문자열입니다(유효한 값: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

분류자에서 CSV를 처리하기 위한 Serde를 설정합니다. 이는 데이터 카탈로그에서 적용됩니다. 유효한 값은 `OpenCSVSerDe`, `LazySimpleSerDe`, `None`입니다. 크롤러에서 감지하려는 경우 `None` 값을 지정할 수 있습니다.

CreateGrokClassifierRequest 구조

생성할 `CreateClassifier`를 위해 `grok` 분류자를 지정합니다.

필드

- `Classification` – 필수(Required): UTF-8 문자열입니다.

  Twitter, JSON, Omniture Logs, Amazon CloudWatch Logs 등과 같이 분류자가 일치하는 데이터 양식의 분류자입니다.
- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  새로운 분류자의 이름입니다.
- `GrokPattern` – 필수(Required): A Logstash Grok string pattern과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  이 분류자가 사용하는 grok 패턴입니다.
- `CustomPatterns` – URI address multi-line string pattern과(와) 일치하는 16,000바이트 이하 길이의 UTF-8 문자열입니다.

  이 분류자가 사용하는 조건부 사용자 지정 grok 패턴입니다.

UpdateGrokClassifierRequest 구조

`UpdateClassifier`로 전달되면 업데이트될 grok 분류자를 지정합니다.

필드

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  `GrokClassifier`의 이름입니다.
- `Classification` – UTF-8 문자열입니다.

Twitter, JSON, Omniture Logs, Amazon CloudWatch Logs 등과 같이 분류자가 일치하는 데이터 양식의 분류자입니다.

- `GrokPattern` – A Logstash Grok string pattern과(와) 일치하는 1~2,048바이트 길이의 UTF-8 문자열입니다.

  이 분류자가 사용하는 grok 패턴입니다.

- `CustomPatterns` – URI address multi-line string pattern과(와) 일치하는 16,000바이트 이하 길이의 UTF-8 문자열입니다.

  이 분류자가 사용하는 조건부 사용자 지정 grok 패턴입니다.

CreateXMLClassifierRequest 구조

생성할 `CreateClassifier`를 위해 XML 분류자를 지정합니다.

필드

- `Classification` – 필수(Required): UTF-8 문자열입니다.

  분류자가 일치하는 데이터 양식의 분류자입니다.

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.

- `RowTag` – UTF-8 문자열입니다.

  각 기록을 구문 분석된 XML 문서에 포함하는 요소를 설계하는 XML 태그입니다. 이 태그는 자기 닫기 요소(`/>`에 의해 닫힌 요소)를 식별할 수 없습니다. 요소가 닫는 태그를 통해 종료되는 한 구문 분석된 속성만 포함하는 빈 행 요소입니다 (예를 들어, `<row item_a="A" item_b="B"></row>`는 괜찮지만 `<row item_a="A" item_b="B" />`는 괜찮지 않습니다).

UpdateXMLClassifierRequest 구조

업데이트될 XML 분류자를 지정합니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.
- Classification – UTF-8 문자열입니다.

  분류자가 일치하는 데이터 양식의 분류자입니다.
- RowTag – UTF-8 문자열입니다.

  각 기록을 구문 분석된 XML 문서에 포함하는 요소를 설계하는 XML 태그입니다. 이 태그는 자기 닫기 요소(/>에 의해 닫힌 요소)를 식별할 수 없습니다. 요소가 닫는 태그를 통해 종료되는 한 구문 분석된 속성만 포함하는 빈 행 요소입니다 (예를 들어, `<row item_a="A" item_b="B"></row>`는 괜찮지만 `<row item_a="A" item_b="B" />`는 괜찮지 않습니다).

CreateJsonClassifierRequest 구조

생성할 CreateClassifier를 위해 JSON 분류자를 지정합니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.
- JsonPath – 필수(Required): UTF-8 문자열입니다.

  분류자가 분류할 JSON 데이터를 정의하는 JsonPath 문자열입니다. AWS Glue는 Writing JsonPath 사용자 정의 분류자 작성에 설명된 대로 JsonPath의 하위 집합을 지원합니다.

UpdateJsonClassifierRequest 구조

업데이트될 JSON 분류자를 지정합니다.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

분류자의 이름입니다.

- `JsonPath` – UTF-8 문자열입니다.

  분류자가 분류할 JSON 데이터를 정의하는 `JsonPath` 문자열입니다. AWS Glue는 Writing JsonPath 사용자 정의 분류자 작성에 설명된 대로 JsonPath의 하위 집합을 지원합니다.

CreateCsvClassifierRequest 구조

생성할 `CreateClassifier`를 위해 사용자 지정 CSV 분류자를 지정합니다.

필드

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.

- `Delimiter` – Custom string pattern #26과(와) 일치하는 1~1바이트 길이의 UTF-8 문자열입니다.

  행의 열 입력 항목 각각을 구분하는 것을 나타내기 위한 사용자 지정 기호입니다.

- `QuoteSymbol` – Custom string pattern #26과(와) 일치하는 1~1바이트 길이의 UTF-8 문자열입니다.

  단일 열 값에 내용을 결합하는 것을 나타내기 위한 사용자 지정 기호입니다. 열 구분 기호와 달라야 합니다.

- `ContainsHeader` – UTF-8 문자열입니다(유효 값: `UNKNOWN` | `PRESENT` | `ABSENT`).

  CSV 파일에 헤더가 포함되어 있는지 여부를 나타냅니다.

- `Header` – UTF-8 문자열의 배열입니다.

  열 이름을 나타내는 문자열 목록입니다.

- `DisableValueTrimming` – 부울입니다.

  열 값의 유형을 식별하기 전에 값을 트리밍하지 않도록 지정합니다. 기본값은 true입니다.

- `AllowSingleColumn` – 부울입니다.

  오직 하나의 열만 포함하는 파일을 처리할 수 있도록 합니다.

- `CustomDatatypeConfigured` – 부울입니다.

사용자 지정 데이터 유형을 구성할 수 있습니다.

- `CustomDatatypes` – UTF-8 문자열의 배열입니다.

  지원되는 사용자 지정 데이터 유형 목록을 만듭니다.

- `Serde` – UTF-8 문자열입니다(유효한 값: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  분류자에서 CSV를 처리하기 위한 Serde를 설정합니다. 이는 데이터 카탈로그에서 적용됩니다. 유효한 값은 `OpenCSVSerDe`, `LazySimpleSerDe`, `None`입니다. 크롤러에서 감지하려는 경우 `None` 값을 지정할 수 있습니다.

UpdateCsvClassifierRequest 구조

업데이트될 사용자 지정 CSV 분류자를 지정합니다.

필드

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  분류자의 이름입니다.

- `Delimiter` – Custom string pattern #26과(와) 일치하는 1~1바이트 길이의 UTF-8 문자열입니다.

  행의 열 입력 항목 각각을 구분하는 것을 나타내기 위한 사용자 지정 기호입니다.

- `QuoteSymbol` – Custom string pattern #26과(와) 일치하는 1~1바이트 길이의 UTF-8 문자열입니다.

  단일 열 값에 내용을 결합하는 것을 나타내기 위한 사용자 지정 기호입니다. 열 구분 기호와 달라야 합니다.

- `ContainsHeader` – UTF-8 문자열입니다(유효 값: `UNKNOWN` | `PRESENT` | `ABSENT`).

  CSV 파일에 헤더가 포함되어 있는지 여부를 나타냅니다.

- `Header` – UTF-8 문자열의 배열입니다.

  열 이름을 나타내는 문자열 목록입니다.

- `DisableValueTrimming` – 부울입니다.

  열 값의 유형을 식별하기 전에 값을 트리밍하지 않도록 지정합니다. 기본값은 true입니다.

- `AllowSingleColumn` – 부울입니다.

오직 하나의 열만 포함하는 파일을 처리할 수 있도록 합니다.

- `CustomDatatypeConfigured` – 부울입니다.

  사용자 지정 데이터 유형의 구성을 지정합니다.

- `CustomDatatypes` – UTF-8 문자열의 배열입니다.

  지원되는 사용자 지정 데이터 유형 목록을 지정합니다.

- `Serde` – UTF-8 문자열입니다(유효한 값: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  분류자에서 CSV를 처리하기 위한 Serde를 설정합니다. 이는 데이터 카탈로그에서 적용됩니다. 유효한 값은 `OpenCSVSerDe`, `LazySimpleSerDe`, `None`입니다. 크롤러에서 감지하려는 경우 `None` 값을 지정할 수 있습니다.

운영

- CreateClassifier 작업(Python: create_classifier)
- DeleteClassifier 작업(Python: delete_classifier)
- GetClassifier 작업(Python: get_classifier)
- GetClassifiers 작업(Python: get_classifiers)
- UpdateClassifier 작업(Python: update_classifier)

CreateClassifier 작업(Python: create_classifier)

사용자 계정에 분류자를 만듭니다. 이 분류자는 어떤 요청 필드가 존재하는지에 따라 `GrokClassifier`, `XMLClassifier`, `JsonClassifier` 또는 `CsvClassifier`일 수 있습니다.

요청

- `GrokClassifier` – CreateGrokClassifierRequest 객체입니다.

  `GrokClassifier` 객체는 생성할 분류자를 지정합니다.

- `XMLClassifier` – CreateXMLClassifierRequest 객체입니다.

  `XMLClassifier` 객체는 생성할 분류자를 지정합니다.

- `JsonClassifier` – CreateJsonClassifierRequest 객체입니다.

  `JsonClassifier` 객체는 생성할 분류자를 지정합니다.

- `CsvClassifier` – CreateCsvClassifierRequest 객체입니다.

  `CsvClassifier` 객체는 생성할 분류자를 지정합니다.

응답

- 무응답 파라미터.

오류

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`

DeleteClassifier 작업(Python: delete_classifier)

데이터 카탈로그에서 분류자를 제거합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  제거할 분류자의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `OperationTimeoutException`

GetClassifier 작업(Python: get_classifier)

이름에 따라 분류자를 검색합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색할 분류자의 이름입니다.

응답

- Classifier – 분류자 객체입니다.

  요청된 분류자

오류

- EntityNotFoundException
- OperationTimeoutException

GetClassifiers 작업(Python: get_classifiers)

데이터 카탈로그에서 분류자 객체를 열거합니다.

요청

- MaxResults – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 크기(선택 사항)입니다.
- NextToken – UTF-8 문자열입니다.

  연속 토큰(선택 사항).

응답

- Classifiers – 분류자 객체의 배열입니다.

  분류자 객체의 요청한 목록입니다.
- NextToken – UTF-8 문자열입니다.

  연속 토큰

오류

- `OperationTimeoutException`

UpdateClassifier 작업(Python: update_classifier)

기존 분류자(어떤 필드가 존재하는지에 따라 `GrokClassifier`, `XMLClassifier`, `JsonClassifier` 또는 `CsvClassifier` 분류자)를 수정합니다.

요청

- `GrokClassifier` – UpdateGrokClassifierRequest 객체입니다.

  업데이트도니 필드와 `GrokClassifier` 객체.
- `XMLClassifier` – UpdateXMLClassifierRequest 객체입니다.

  업데이트도니 필드와 `XMLClassifier` 객체.
- `JsonClassifier` – UpdateJsonClassifierRequest 객체입니다.

  업데이트도니 필드와 `JsonClassifier` 객체.
- `CsvClassifier` – UpdateCsvClassifierRequest 객체입니다.

  업데이트도니 필드와 `CsvClassifier` 객체.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `VersionMismatchException`
- `EntityNotFoundException`
- `OperationTimeoutException`

크롤러 API

크롤러 API는 크롤러를 생성, 삭제, 업데이트 및 나열하기 위한 API와 함께 AWS Glue 크롤러 데이터 유형에 대해 설명합니다.

데이터 타입

- 크롤러 구조
- 일정 구조
- CrawlerTargets 구조
- S3Target 구조
- S3DeltaCatalogTarget 구조
- S3DeltaDirectTarget 구조
- JdbcTarget 구조
- MongoDBTarget 구조
- DynamoDBTarget 구조
- DeltaTarget 구조
- IcebergTarget 구조
- HudiTarget 구조
- CatalogTarget 구조
- CrawlerMetrics 구조
- CrawlerHistory 구조
- CrawlsFilter 구조
- SchemaChangePolicy 구조
- LastCrawlInfo 구조
- RecrawlPolicy 구조
- LineageConfiguration 구조
- LakeFormationConfiguration 구조

크롤러 구조

데이터 원본을 검사하는 크롤러 프로그램을 지정하고 분류자를 사용하여 스키마를 결정합니다. 성공적이면 크롤러는 AWS Glue Data Catalog의 데이터 원본을 고려하여 메타데이터를 기록합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  크롤러의 이름입니다.

- `Role` – UTF-8 문자열입니다.

  Amazon Simple Storage Service(Amazon S3) 데이터 등의 고객 리소스에 액세스하는 데 사용되는 IAM 역할의 Amazon 리소스 이름(ARN)입니다.

- `Targets` – CrawlerTargets 객체입니다.

  크롤할 대상 모음입니다.

- `DatabaseName` – UTF-8 문자열입니다.

  크롤러의 출력이 저장되는 데이터베이스의 이름입니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  크롤러에 대한 설명.

- `Classifiers` – UTF-8 문자열의 배열입니다.

  크롤러와 연결된 사용자 지정 분류자를 지정하는 UTF-8 문자열 목록입니다.

- `RecrawlPolicy` – RecrawlPolicy 객체입니다.

  전체 데이터 집합을 다시 크롤링할지 아니면 마지막 크롤러 실행 이후 추가된 폴더만 크롤링할지 지정하는 정책입니다.

- `SchemaChangePolicy` – SchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 및 삭제 동작을 지정하는 정책입니다.

- `LineageConfiguration` – LineageConfiguration 객체입니다.

  크롤러에 대해 데이터 계보가 사용되는지 여부를 지정하는 구성입니다.

- `State` – UTF-8 문자열입니다(유효 값: `READY` | `RUNNING` | `STOPPING`).

  크롤러가 실행되거나 실행되지 않았는지 여부를 나타냅니다.

- `TablePrefix` – 128바이트 이하 길이의 UTF-8 문자열입니다.

  생성된 테이블 이름에 추가된 접두사.

- `Schedule` – 일정 객체입니다.

  일정이 짜여진 크롤러를 위한 크롤러가 실행될 때의 일정.

- `CrawlElapsedTime` - 숫자(정수)입니다.

  크롤러가 실행되면 마지막 크롤이 시작된 후부터 총 경과 시간.

- `CreationTime` – 타임스탬프입니다.

  크롤러가 생성된 시간.

- `LastUpdated` – 타임스탬프입니다.

  크롤러가 마지막으로 업데이트된 시간.

- `LastCrawl` – LastCrawlInfo 객체입니다.

  마지막 크롤 상태 및 오류가 발생한 잠재적 오류 정보.

- `Version` - 숫자(정수)입니다.

  크롤러 버전.

- `Configuration` – UTF-8 문자열입니다.

  크롤러 구성 정보. 이 버전의 JSON 문자열은 사용자가 크롤러 동작을 지정할 수 있게 만듭니다. 자세한 내용을 알아보려면 크롤러 구성 옵션 설정을 참조하세요.

- `CrawlerSecurityConfiguration` – 128바이트 이하 길이의 UTF-8 문자열입니다.

  이 크롤러가 사용할 `SecurityConfiguration` 구조의 이름입니다.

- `LakeFormationConfiguration` – LakeFormationConfiguration 객체입니다.

  크롤러가 IAM 역할 자격 증명 대신 AWS Lake Formation 자격 증명을 크롤러에 사용해야 하는지 지정합니다.

일정 구조

`cron`을 사용하여 객체의 일정을 정하여 이벤트의 일정을 정합니다.

필드

- `ScheduleExpression` – UTF-8 문자열입니다.

일정을 지정하는 데 사용되는 cron 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 cron(15 12 * * ? *)을 지정합니다.

- State – UTF-8 문자열입니다(유효 값: SCHEDULED | NOT_SCHEDULED | TRANSITIONING).

  일정 상태

CrawlerTargets 구조

크롤할 데이터 스토어 지정.

필드

- S3Targets – S3Target 객체의 배열입니다.

  Amazon Simple Storage Service(Amazon S3) 대상을 지정합니다.

- JdbcTargets – JdbcTarget 객체의 배열입니다.

  JDBC 대상 지정

- MongoDBTargets – MongoDBTarget 객체의 배열입니다.

  Amazon DocumentDB 또는 MongoDB 대상을 지정합니다.

- DynamoDBTargets – DynamoDBTarget 객체의 배열입니다.

  Amazon DynamoDB 대상을 지정합니다.

- CatalogTargets – CatalogTarget 객체의 배열입니다.

  AWS Glue Data Catalog 대상을 지정합니다.

- DeltaTargets – DeltaTarget 객체의 배열입니다.

  델타 데이터 스토어 대상을 지정합니다.

- IcebergTargets – IcebergTarget 객체의 배열입니다.

  Apache Iceberg 데이터 스토어 대상을 지정합니다.

- HudiTargets – HudiTarget 객체의 배열입니다.

  Apache Hudi 데이터 스토어 대상을 지정합니다.

S3Target 구조

Amazon Simple Storage Service(Amazon S3)의 데이터 스토어를 지정합니다.

필드

- `Path` – UTF-8 문자열입니다.

  Amazon S3 대상에 대한 경로입니다.
- `Exclusions` – UTF-8 문자열의 배열입니다.

  크롤링에서 제외하는 데 사용되는 glob 패턴 목록입니다. 자세한 내용은 크롤러를 사용하여 테이블 분류를 참조하십시오.
- `ConnectionName` – UTF-8 문자열입니다(1~2,048바이트).

  작업 또는 크롤러가 Amazon Virtual Private Cloud 환경(Amazon VPC) 내에서 Amazon S3의 데이터에 액세스할 수 있도록 허용하는 연결 이름입니다.
- `SampleSize` - 숫자(정수)입니다.

  데이터 집합의 샘플 파일을 크롤링할 때 크롤링할 각 리프 폴더의 파일 수를 설정합니다. 설정하지 않으면 모든 파일이 크롤링됩니다. 유효한 값은 1~249의 정수입니다.
- `EventQueueArn` – UTF-8 문자열입니다.

  유효한 Amazon SQS ARN입니다. 예를 들어 `arn:aws:sqs:region:account:sqs`입니다.
- `DlqEventQueueArn` – UTF-8 문자열입니다.

  유효한 Amazon 배달 못한 편지 SQS ARN입니다. 예를 들어 `arn:aws:sqs:region:account:deadLetterQueue`입니다.

S3DeltaCatalogTarget 구조

AWS Glue 데이터 카탈로그의 Delta Lake 데이터 소스에 작성하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

데이터 대상에 대한 입력인 노드입니다.

- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.

- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션을 지정합니다.

- `SchemaChangePolicy` – CatalogSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

S3DeltaDirectTarget 구조

Amazon S3에서 Delta Lake 데이터 소스에 작성하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.

- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 Delta Lake 데이터 소스의 Amazon S3 경로입니다.

- `Compression` – 필수: UTF-8 문자열입니다(유효한 값: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).

- `NumberTargetPartitions` – UTF-8 문자열입니다.

  Amazon S3 전체에 Delta Lake 데이터세트 파일을 배포할 대상 파티션 수를 지정합니다.

- `Format` – 필수: UTF-8 문자열(유효한 값: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"` | `iceberg="ICEBERG"` | `hyper="HYPER"` | `xml="XML"`).

  대상에 대한 데이터 출력 포맷을 지정합니다.

- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션을 지정합니다.

- `SchemaChangePolicy` – DirectSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

JdbcTarget 구조

크롤할 JDBC 데이터 스토어 지정.

필드

- `ConnectionName` – UTF-8 문자열입니다(1~2,048바이트).

  JDBC 대상에 연결할 연결 이름입니다.

- `Path` – UTF-8 문자열입니다.

  JDBC 대상의 경로입니다.

- `Exclusions` – UTF-8 문자열의 배열입니다.

크롤링에서 제외하는 데 사용되는 glob 패턴 목록입니다. 자세한 내용은 크롤러를 사용하여 테이블 분류를 참조하십시오.

- `EnableAdditionalMetadata` – UTF-8 문자열의 배열입니다.

  `RAWTYPES` 또는 `COMMENTS` 값을 지정하여 테이블 응답에서 추가 메타데이터를 활성화합니다. `RAWTYPES`는 기본 레벨 데이터 유형을 제공합니다. `COMMENTS`는 데이터베이스의 열 또는 테이블과 연결된 설명을 제공합니다.

  추가 메타데이터가 필요하지 않은 경우 필드를 비워 두세요.

MongoDBTarget 구조

크롤링할 Amazon DocumentDB 또는 MongoDB 데이터 스토어를 지정합니다.

필드

- `ConnectionName` – UTF-8 문자열입니다(1~2,048바이트).

  Amazon DocumentDB 또는 MongoDB 대상에 연결하는 데 사용할 연결 이름입니다.
- `Path` – UTF-8 문자열입니다.

  Amazon DocumentDB 또는 MongoDB 대상(데이터베이스/컬렉션)의 경로입니다.
- `ScanAll` – 부울입니다.

  모든 레코드를 스캔할지 또는 테이블에서 행을 샘플링할지 여부를 나타냅니다. 테이블이 높은 처리량 테이블이 아닌 경우 모든 레코드를 스캔하는 데 시간이 오래 걸릴 수 있습니다.

  `true` 값은 모든 레코드를 스캔하는 것을 의미하고 `false` 값은 레코드를 샘플링하는 것을 의미합니다. 값을 지정하지 않으면 기본값은 `true`입니다.

DynamoDBTarget 구조

크롤할 Amazon DynamoDB 테이블을 지정합니다.

필드

- `Path` – UTF-8 문자열입니다.

  크롤할 DynamoDB 테이블의 이름입니다.

- `scanAll` – 부울입니다.

  모든 레코드를 스캔할지 또는 테이블에서 행을 샘플링할지 여부를 나타냅니다. 테이블이 높은 처리량 테이블이 아닌 경우 모든 레코드를 스캔하는 데 시간이 오래 걸릴 수 있습니다.

  `true` 값은 모든 레코드를 스캔하는 것을 의미하고 `false` 값은 레코드를 샘플링하는 것을 의미합니다. 값을 지정하지 않으면 기본값은 `true`입니다.

- `scanRate` - 숫자(double)입니다.

  AWS Glue 크롤러에서 사용할 구성된 읽기 용량 단위의 비율입니다. 읽기 용량 단위는 DynamoDB에서 정의한 용어이며, 초당 해당 테이블에서 수행할 수 있는 읽기 수에 대한 속도 제한기 역할을 하는 숫자 값입니다.

  유효한 값은 null 또는 0.1~1.5의 값입니다. null 값은 사용자가 값을 제공하지 않을 때 사용되며, 기본값은 구성된 읽기 용량 단위의 0.5(프로비저닝된 테이블의 경우) 또는 구성된 최대 읽기 용량 단위의 0.25(온디맨드 모드를 사용하는 테이블의 경우)입니다.

DeltaTarget 구조

하나 이상의 델타 테이블을 크롤링할 델타 데이터 스토어를 지정합니다.

필드

- `DeltaTables` – UTF-8 문자열의 배열입니다.

  델타 테이블에 대한 Amazon S3 경로 목록입니다.

- `ConnectionName` – UTF-8 문자열입니다(1~2,048바이트).

  델타 테이블 대상에 연결하는 데 사용할 연결 이름입니다.

- `WriteManifest` – 부울입니다.

  매니페스트 파일을 델타 테이블 경로에 쓸지 지정합니다.

- `CreateNativeDeltaTable` – 부울입니다.

  크롤러가 Delta 트랜잭션 로그의 쿼리를 직접 지원하는 쿼리 엔진과 통합할 수 있도록 기본 테이블을 생성할지 여부를 지정합니다.

IcebergTarget 구조

Amazon S3에서 Iceberg 테이블이 저장되는 Apache Iceberg 데이터 소스를 지정합니다.

필드

- `Paths` – UTF-8 문자열의 배열입니다.

  Iceberg 메타데이터 폴더를 `s3://bucket/prefix`로 포함하는 하나 이상의 Amazon S3 경로입니다.

- `ConnectionName` – UTF-8 문자열입니다(1~2,048바이트).

  Iceberg 대상에 연결하는 데 사용할 연결 이름입니다.

- `Exclusions` – UTF-8 문자열의 배열입니다.

  크롤링에서 제외하는 데 사용되는 glob 패턴 목록입니다. 자세한 내용은 크롤러를 사용하여 테이블 분류를 참조하십시오.

- `MaximumTraversalDepth` - 숫자(정수)입니다.

  크롤러가 Amazon S3 경로에 있는 Iceberg 메타데이터 폴더를 검색하기 위해 탐색할 수 있는 Amazon S3 경로의 최대 깊이입니다. 크롤러 실행 시간을 제한하는 데 사용됩니다.

HudiTarget 구조

Apache Hudi 데이터 소스를 지정합니다.

필드

- `Paths` – UTF-8 문자열의 배열입니다.

  Hudi의 Amazon S3 위치 문자열 배열로, 각각 Hudi 테이블의 메타데이터 파일이 있는 루트 폴더를 나타냅니다. Hudi 폴더는 루트 폴더의 하위 폴더에 있을 수 있습니다.

  크롤러는 경로 아래에 있는 모든 폴더에서 Hudi 폴더를 스캔합니다.

- `ConnectionName` – UTF-8 문자열입니다(1~2,048바이트).

  Hudi 대상에 연결하는 데 사용할 연결 이름입니다. VPC 인증이 필요한 버킷에 Hudi 파일이 저장된 경우 여기에서 연결 속성을 설정할 수 있습니다.

- `Exclusions` – UTF-8 문자열의 배열입니다.

크롤링에서 제외하는 데 사용되는 glob 패턴 목록입니다. 자세한 내용은 크롤러를 사용하여 테이블 분류를 참조하십시오.

- `MaximumTraversalDepth` - 숫자(정수)입니다.

  크롤러가 Amazon S3 경로에 있는 Hudi 메타데이터 폴더를 검색하기 위해 탐색할 수 있는 Amazon S3 경로의 최대 깊이입니다. 크롤러 실행 시간을 제한하는 데 사용됩니다.

CatalogTarget 구조

AWS Glue Data Catalog 대상을 지정합니다.

필드

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  동기화할 데이터베이스의 이름입니다.

- `Tables` – 필수(Required): UTF-8 문자열의 배열이며 문자열은 1개 이상입니다.

  동기화할 테이블의 목록입니다.

- `ConnectionName` – UTF-8 문자열입니다(1~2,048바이트).

  `NETWORK` 연결 유형에 페어링된 `Catalog` 연결 유형을 사용할 때 Amazon S3 기반 데이터 카탈로그 테이블이 크롤링의 대상이 되도록 하는 연결의 이름입니다.

- `EventQueueArn` – UTF-8 문자열입니다.

  유효한 Amazon SQS ARN입니다. 예를 들어 `arn:aws:sqs:region:account:sqs`입니다.

- `DlqEventQueueArn` – UTF-8 문자열입니다.

  유효한 Amazon 배달 못한 편지 SQS ARN입니다. 예를 들어 `arn:aws:sqs:region:account:deadLetterQueue`입니다.

CrawlerMetrics 구조

지정한 크롤러에 대한 지표.

필드

- `CrawlerName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  크롤러의 이름입니다.

- `TimeLeftSeconds` – None 이하의 숫자(실수)입니다.

  크롤을 완료할 때까지 남은 예상 시간.

- `StillEstimating` – 부울입니다.

  이 작업이 완료될 때까지 소요 시간을 여전히 추정하고 있다면 True입니다.

- `LastRuntimeSeconds` – None 이하의 숫자(실수)입니다.

  초 단위의 크롤러의 최근 작업 실행 지속 시간.

- `MedianRuntimeSeconds` – None 이하의 숫자(실수)입니다.

  초 단위의 크롤러의 평균 작업 실행 지속 시간.

- `TablesCreated` – None 이하의 숫자(정수)입니다.

  이 크롤러가 생성한 테이블 수.

- `TablesUpdated` – None 이하의 숫자(정수)입니다.

  이 크롤러가 업데이트한 테이블 수.

- `TablesDeleted` – None 이하의 숫자(정수)입니다.

  이 크롤러가 삭제한 테이블 수.

CrawlerHistory 구조

크롤러 실행에 대한 정보가 포함되어 있습니다.

필드

- `CrawlId` – UTF-8 문자열입니다.

  각 크롤링에 대한 UUID 식별자입니다.

- `State` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `COMPLETED` | `FAILED` | `STOPPED`).

크롤의 상태.

- `StartTime` – 타임스탬프입니다.

  크롤이 시작된 날짜와 시간입니다.

- `EndTime` – 타임스탬프입니다.

  크롤이 시작된 날짜와 시간.

- `Summary` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  JSON의 특정 크롤에 대한 실행 요약. 추가, 업데이트 또는 삭제된 카탈로그 테이블 및 파티션을 포함합니다.

- `ErrorMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  (오류가 발생한 경우) 크롤과 연결된 오류 메시지.

- `LogGroup` – Log group string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  크롤과 연결된 로그 그룹입니다.

- `LogStream` – Log-stream string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  크롤과 연결된 로그 스트림입니다.

- `MessagePrefix` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 크롤에 관한 CloudWatch 메시지 접두사.

- `DPUHour` – None 이하의 숫자(실수)입니다.

  크롤에 사용된 DPU(데이터 처리 단위)의 수(시간 단위).

CrawlsFilter 구조

지정된 크롤러에 대한 크롤러 실행을 필터링하는 데 사용할 수 있는 필드, 비교기 및 값의 목록입니다.

필드

- `FieldName` – UTF-8 문자열입니다(유효한 값: `CRAWL_ID` | `STATE` | `START_TIME` | `END_TIME` | `DPU_HOUR`).

지정된 크롤러에 대한 크롤러 실행 필터링에 사용되는 키. 각 필드 이름에 유효한 값은 다음과 같습니다.

- CRAWL_ID: 크롤링의 UUID 식별자를 나타내는 문자열.
- STATE: 크롤의 상태를 나타내는 문자열.
- START_TIME 및 END_TIME: 밀리초 단위의 Epoch 타임스탬프.
- DPU_HOUR: 크롤에 사용된 DPU(데이터 처리 단위)의 수(시간 단위).

- FilterOperator – UTF-8 문자열입니다(유효 값: GT | GE | LT | LE | EQ | NE).

  값의 연산을 수행하는 정의된 비교기. 사용 가능한 연산자는 다음과 같습니다.

  - GT: 큼.
  - GE: 크거나 같음.
  - LT: 작음.
  - LE: 작거나 같음.
  - EQ: 같음.
  - NE: 같지 않음.
- FieldValue – UTF-8 문자열입니다.

  크롤 필드에서 비교하기 위해 제공된 값입니다.

SchemaChangePolicy 구조

크롤러에 대한 업데이트 및 삭제 동작을 지정하는 정책입니다.

필드

- UpdateBehavior – UTF-8 문자열입니다(유효 값: LOG | UPDATE_IN_DATABASE).

  크롤러가 변화된 객체를 찾을 때 업데이트 동작.
- DeleteBehavior – UTF-8 문자열입니다(유효 값: LOG | DELETE_FROM_DATABASE | DEPRECATE_IN_DATABASE).

  크롤러가 변화된 혹은 삭제된 객체를 찾을 때 삭제 동작.

LastCrawlInfo 구조

최신 크롤의 상태 및 오류 정보.

필드

- `Status` – UTF-8 문자열입니다(유효 값: `SUCCEEDED` | `CANCELLED` | `FAILED`).

  최종 크롤 상태

- `ErrorMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  오류가 발생할 때 마지막 크롤에 대한 오류 정보.

- `LogGroup` – Log group string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  마지막 크롤의 로그 그룹.

- `LogStream` – Log-stream string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  마지막 크롤의 로그 스트림.

- `MessagePrefix` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 크롤 메시지에 대한 접두사.

- `StartTime` – 타임스탬프입니다.

  크롤이 시작된 시간.

RecrawlPolicy 구조

첫 번째 크롤링이 완료된 후 Amazon S3 데이터 원본을 크롤링할 때 전체 데이터 집합을 다시 크롤링할지 아니면 마지막 크롤러 실행 이후에 추가된 폴더만 크롤링할지 지정합니다. 자세한 내용은 SageMaker 개발자 안내서의 AWS Glue의 증분 크롤링을 참조하세요.

필드

- `RecrawlBehavior` – UTF-8 문자열입니다(유효한 값: `CRAWL_EVERYTHING` | `CRAWL_NEW_FOLDERS_ONLY` | `CRAWL_EVENT_MODE`).

  전체 데이터 집합을 다시 크롤링할지 아니면 마지막 크롤러 실행 이후 추가된 폴더만 크롤링할지 지정합니다.

값 CRAWL_EVERYTHING은 전체 데이터 집합을 다시 크롤링하도록 지정합니다.

값 CRAWL_NEW_FOLDERS_ONLY는 마지막 크롤러 실행 이후에 추가된 폴더만 크롤링하도록 지정합니다.

값 CRAWL_EVENT_MODE는 Amazon S3 이벤트에서 식별된 변경 사항만 크롤링하도록 지정합니다.

LineageConfiguration 구조

크롤러에 대한 데이터 계보 구성 설정을 지정합니다.

필드

- CrawlerLineageSettings – UTF-8 문자열입니다(유효 값: ENABLE | DISABLE).

  크롤러에 데이터 계보가 사용되는지 여부를 지정합니다. 유효한 값은 다음과 같습니다.
  - [사용(ENABLE)]: 크롤러에 데이터 계보를 사용합니다.
  - [사용 중지(DISABLE)]: 크롤러에 데이터 계보 사용을 중지합니다.

LakeFormationConfiguration 구조

크롤러에 대한 AWS Lake Formation 구성 설정을 지정합니다.

필드

- UseLakeFormationCredentials – 부울입니다.

  IAM 역할 자격 증명 대신 AWS Lake Formation 자격 증명을 크롤러에 사용해야 하는지 지정합니다.
- AccountId – UTF-8 문자열입니다(12바이트 이하).

  교차 계정 크롤링에 필요합니다. 대상 데이터와 동일한 계정 크롤링의 경우 이 값을 null로 둘 수 있습니다.

운영

- CreateCrawler 작업(Python: create_crawler)
- DeleteCrawler 작업(Python: delete_crawler)
- GetCrawler 작업(Python: get_crawler)

- GetCrawlers 작업(Python: get_crawler)
- GetCrawlerMetrics Action(Python: get_crawler_metrics)
- UpdateCrawler 작업(Python: update_crawler)
- StartCrawler 작업(Python: start_crawler)
- StopCrawler 작업(Python: stop_crawler)
- BatchGetCrawlers 작업(Python: batch_get_crawlers)
- ListCrawlers 작업(Python: list_crawlers)
- ListCrawls 작업(Python: list_crawls)

CreateCrawler 작업(Python: create_crawler)

지정된 대상, 역할, 구성 및 선택 일정을 통해 새로운 크롤러를 생성합니다. `s3Targets` 필드, `jdbcTargets` 필드 또는 `DynamoDBTargets` 필드에서 크롤 대상을 하나 이상 지정해야 합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  새로운 크롤러의 이름.
- `Role` – 필수(Required): UTF-8 문자열입니다.

  새로운 크롤러를 사용하여 고객 리소스에 액세스하는 IAM 역할 또는 IAM 역할의 Amazon 리소스 이름(ARN)입니다.
- `DatabaseName` – UTF-8 문자열입니다.

  `arn:aws:daylight:us-east-1::database/sometable/*`와 같은 결과가 작성되는 AWS Glue 데이터베이스입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  새로운 크롤러에 대한 설명.
- `Targets` – 필수(Required): CrawlerTargets 객체입니다.

  크롤할 대상 모음의 기록.
- `Schedule` – UTF-8 문자열입니다.

일정을 지정하는 데 사용되는 cron 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 `cron(15 12 * * ? *)`을 지정합니다.

- `Classifiers` – UTF-8 문자열의 배열입니다.

  사용자가 등록한 사용자 지정 분류자 목록. 기본적으로 모든 기본 설정 분류자는 크롤러에 포함되지만 이 사용자 지정 분류자는 항상 주어진 분류에 대한 기본 분류자를 재정의합니다.

- `TablePrefix` – 128바이트 이하 길이의 UTF-8 문자열입니다.

  생성된 카탈로그 테이블에 사용되는 테이블 접두사입니다.

- `SchemaChangePolicy` – SchemaChangePolicy 객체입니다.

  크롤러의 업데이트 및 삭제 동작 정책입니다.

- `RecrawlPolicy` – RecrawlPolicy 객체입니다.

  전체 데이터 집합을 다시 크롤링할지 아니면 마지막 크롤러 실행 이후 추가된 폴더만 크롤링할지 지정하는 정책입니다.

- `LineageConfiguration` – LineageConfiguration 객체입니다.

  크롤러에 대한 데이터 계보 구성 설정을 지정합니다.

- `LakeFormationConfiguration` – LakeFormationConfiguration 객체입니다.

  크롤러에 대한 AWS Lake Formation 구성 설정을 지정합니다.

- `Configuration` – UTF-8 문자열입니다.

  크롤러 구성 정보. 이 버전의 JSON 문자열은 사용자가 크롤러 동작을 지정할 수 있게 만듭니다. 자세한 내용을 알아보려면 크롤러 구성 옵션 설정을 참조하세요.

- `CrawlerSecurityConfiguration` – 128바이트 이하 길이의 UTF-8 문자열입니다.

  이 크롤러가 사용할 `SecurityConfiguration` 구조의 이름입니다.

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이 크롤러 요청에서 사용할 태그입니다. 태그를 사용하여 크롤러에 대한 액세스를 제한할 수 있습니다. AWS Glue의 태그에 대한 자세한 내용은 개발자 안내서의 AWS Glue의 AWS 태그를 참조하세요.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

DeleteCrawler 작업(Python: delete_crawler)

크롤러 상태가 `RUNNING`이 아닌 한, AWS Glue Data Catalog에서 지정한 크롤러를 제거합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  제거할 크롤러의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `CrawlerRunningException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

GetCrawler 작업(Python: get_crawler)

지정한 크롤러의 메타데이터 가져오기

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  메타데이터를 검색할 크롤러 이름입니다.

응답

- Crawler – 크롤러 객체입니다.

  지정한 크롤러의 메타데이터

오류

- EntityNotFoundException
- OperationTimeoutException

GetCrawlers 작업(Python: get_crawler)

사용자 계정에 정의된 모든 크롤러의 메타데이터를 가져옵니다.

요청

- MaxResults – 1~1,000의 숫자(정수)입니다.

  각 호출에 따라 반환할 크롤러의 수입니다.
- NextToken – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.

응답

- Crawlers – 크롤러 객체의 배열입니다.

  크롤러 메타데이터의 목록.
- NextToken – UTF-8 문자열입니다.

  이 사용자 계정에 정의된 것들의 끝에 반환된 목록이 도달하지 못한 경우, 지속적인 토큰입니다.

오류

- `OperationTimeoutException`

GetCrawlerMetrics Action(Python: get_crawler_metrics)

지정한 크롤러의 지표 가져오기

요청

- `CrawlerNameList` – 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  지표를 가져올 크롤러의 이름 목록.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 최대 크기.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `CrawlerMetricsList` – CrawlerMetrics 객체의 배열입니다.

  지정한 크롤러에 대한 지표 목록.
- `NextToken` – UTF-8 문자열입니다.

  반환된 목록이 사용가능한 마지막 지표를 포함하지 경우의 연속 토큰입니다.

오류

- `OperationTimeoutException`

UpdateCrawler 작업(Python: update_crawler)

크롤러 업데이트. 크롤러가 실행 중이면 업데이트하기 전에는 `StopCrawler`를 사용하여 중지해야 합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  새로운 크롤러의 이름.

- Role – UTF-8 문자열입니다.

  새로운 크롤러를 사용하여 고객 리소스에 액세스하는 IAM 역할 또는 IAM 역할의 Amazon 리소스 이름(ARN)입니다.

- DatabaseName – UTF-8 문자열입니다.

  arn:aws:daylight:us-east-1::database/sometable/*와 같은 결과가 저장되는 AWS Glue 데이터베이스입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  새로운 크롤러에 대한 설명.

- Targets – CrawlerTargets 객체입니다.

  크롤할 대상 목록.

- Schedule – UTF-8 문자열입니다.

  일정을 지정하는 데 사용되는 cron 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 cron(15 12 * * ? *)을 지정합니다.

- Classifiers – UTF-8 문자열의 배열입니다.

  사용자가 등록한 사용자 지정 분류자 목록. 기본적으로 모든 기본 설정 분류자는 크롤러에 포함되지만 이 사용자 지정 분류자는 항상 주어진 분류에 대한 기본 분류자를 재정의합니다.

- TablePrefix – 128바이트 이하 길이의 UTF-8 문자열입니다.

  생성된 카탈로그 테이블에 사용되는 테이블 접두사입니다.

- SchemaChangePolicy – SchemaChangePolicy 객체입니다.

  크롤러의 업데이트 및 삭제 동작 정책입니다.

- RecrawlPolicy – RecrawlPolicy 객체입니다.

전체 데이터 집합을 다시 크롤링할지 아니면 마지막 크롤러 실행 이후 추가된 폴더만 크롤링할지 지정하는 정책입니다.

- `LineageConfiguration` – LineageConfiguration 객체입니다.

  크롤러에 대한 데이터 계보 구성 설정을 지정합니다.

- `LakeFormationConfiguration` – LakeFormationConfiguration 객체입니다.

  크롤러에 대한 AWS Lake Formation 구성 설정을 지정합니다.

- `Configuration` – UTF-8 문자열입니다.

  크롤러 구성 정보. 이 버전의 JSON 문자열은 사용자가 크롤러 동작을 지정할 수 있게 만듭니다. 자세한 내용을 알아보려면 크롤러 구성 옵션 설정을 참조하세요.

- `CrawlerSecurityConfiguration` – 128바이트 이하 길이의 UTF-8 문자열입니다.

  이 크롤러가 사용할 `SecurityConfiguration` 구조의 이름입니다.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `VersionMismatchException`
- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StartCrawler 작업(Python: start_crawler)

어떤 일정이든지 지정된 크롤러를 사용하여 크롤러를 시작합니다. 크롤러가 이미 실행 중이면 CrawlerRunningException을 반환합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  시작할 크롤러의 이름.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StopCrawler 작업(Python: stop_crawler)

지정된 크롤러가 실행 중이면 크롤러를 중지합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  중지할 크롤러의 이름.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `CrawlerNotRunningException`
- `CrawlerStoppingException`

- OperationTimeoutException

BatchGetCrawlers 작업(Python: batch_get_crawlers)

주어진 크롤러 이름 목록에 대한 리소스 메타데이터 목록을 반환합니다. ListCrawlers 작업을 호출한 후에는 권한이 부여된 데이터에 액세스하기 위해 이 작업을 호출할 수 있습니다. 이 작업은 태그를 사용하는 권한 조건을 포함해 모든 IAM 권한을 지원합니다.

요청

- CrawlerNames – 필수(Required): 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  크롤러 이름(ListCrawlers 작업에서 반환된 이름일 수 있음)의 목록입니다.

응답

- Crawlers – 크롤러 객체의 배열입니다.

  크롤러 정의 목록.
- CrawlersNotFound – 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  찾을 수 없는 크롤러의 이름 목록입니다.

오류

- InvalidInputException
- OperationTimeoutException

ListCrawlers 작업(Python: list_crawlers)

이 AWS 계정의 모든 크롤러 리소스의 이름 또는 지정된 태그를 가진 리소스를 검색합니다. 이 작업을 통해 계정에서 사용 가능한 리소스와 그 이름을 확인할 수 있습니다.

이 작업을 수행하면 응답에서 필터로 사용할 수 있는 선택 사항인 Tags 필드가 검색되기 때문에 태그가 지정된 리소스를 하나의 그룹으로 검색할 수 있습니다. 태그 필터링을 사용하기로 선택하면 태그가 포함된 리소스만 검색됩니다.

요청

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 최대 크기.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이렇게 태그가 지정된 리소스만 반환하도록 지정합니다.

응답

- `CrawlerNames` – 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  계정의 모든 크롤러 또는 지정된 태그를 가진 크롤러의 이름.
- `NextToken` – UTF-8 문자열입니다.

  반환된 목록이 사용가능한 마지막 지표를 포함하지 경우의 연속 토큰입니다.

오류

- `OperationTimeoutException`

ListCrawls 작업(Python: list_crawls)

지정된 크롤러에 대한 모든 크롤이 반환됩니다. 크롤러 기록 기능의 시작 날짜 이후 발생한 크롤만 반환되고 최대 12개월의 크롤만 유지됩니다. 이전의 크롤은 반환되지 않습니다.

이 API를 사용하여 다음을 수행할 수 있습니다.

- 지정된 크롤러의 모든 크롤을 검색합니다.
- 제한된 수 내에서 지정된 크롤러의 모든 크롤을 검색합니다.
- 특정 시간 범위에서 지정된 크롤러의 모든 크롤을 검색합니다.

- 특정 상태, 크롤 ID 또는 DPU 시간 값을 사용하여 지정된 크롤러의 모든 크롤을 검색합니다.

요청

- `CrawlerName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행을 검색할 크롤러의 이름입니다.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다. 기본값은 20이고 최대값은 100입니다.
- `Filters` – CrawlsFilter 객체의 배열입니다.

  `CrawlsFilter` 객체 목록에서 지정하는 기준에 따라 크롤을 필터링합니다.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `Crawls` – CrawlerHistory 객체의 배열입니다.

  기준을 충족하는 크롤 실행을 나타내는 `CrawlerHistory` 객체의 목록입니다.
- `NextToken` – UTF-8 문자열입니다.

  목록의 현재 세그먼트가 마지막이 아니면 반환된 토큰 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

오류

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`

열 통계의 API

열 통계 AWS Glue API는 테이블의 열에 대한 통계를 반환하기 위한 API를 설명합니다.

데이터 타입

- ColumnStatisticsTaskRun 구조
- ColumnStatisticsTaskSettings 구조
- ExecutionAttempt 구조

ColumnStatisticsTaskRun 구조

열 통계 실행의 세부 정보를 표시하는 객체입니다.

필드

- `CustomerId` – UTF-8 문자열입니다(12바이트 이하).

  AWS 계정 ID입니다.
- `ColumnStatisticsTaskRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  특정 열 통계 작업 실행의 식별자입니다.
- `DatabaseName` – UTF-8 문자열입니다.

  테이블이 상주하는 데이터베이스.
- `TableName` – UTF-8 문자열입니다.

  열 통계가 생성되는 테이블의 이름입니다.
- `ColumnNameList` – UTF-8 문자열의 배열입니다.

  열 이름의 목록입니다. 제공되지 않은 경우 기본적으로 테이블의 모든 열 이름이 사용됩니다.
- `CatalogID` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.
- `Role` – UTF-8 문자열입니다.

  서비스가 통계를 생성하기 위해 맡을 IAM 역할.
- `SampleSize` – 100 이하의 숫자(실수)입니다.

통계 생성에 사용된 행의 비율. 제공되지 않은 경우 전체 테이블을 사용하여 통계를 생성합니다.

- `SecurityConfiguration` – 128바이트 이하 길이의 UTF-8 문자열입니다.

  열 통계 작업 실행의 CloudWatch 로그를 암호화하는 데 사용되는 보안 구성의 이름입니다.

- `NumberOfWorkers` – 1 이상의 숫자(정수)입니다.

  열 통계를 생성하는 데 사용된 작업자 수입니다. 작업은 인스턴스 25개까지 자동 확장되도록 사전 구성되어 있습니다.

- `WorkerType` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 생성에 사용되는 작업자 유형. 기본값은 `g.1x`입니다.

- `ComputationType` – UTF-8 문자열입니다(유효한 값: `FULL` | `INCREMENTAL`).

  열 통계 계산의 유형입니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `STARTING` | `RUNNING` | `SUCCEEDED` | `FAILED` | `STOPPED`).

  실행된 작업의 상태입니다.

- `CreationTime` – 타임스탬프입니다.

  이 작업이 생성된 시각입니다.

- `LastUpdated` – 타임스탬프입니다.

  이 작업이 마지막으로 수정된 시점.

- `StartTime` – 타임스탬프입니다.

  이벤트의 시작 시각.

- `EndTime` – 타임스탬프입니다.

  작업의 종료 시각.

- `ErrorMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  작업에 대한 오류 메시지입니다.

- `DPUSeconds` – None 이하의 숫자(실수)입니다.

  자동 크기 조정된 모든 작업자의 계산된 DPU 사용량(초 단위).

ColumnStatisticsTaskSettings 구조

열 통계 작업의 설정입니다.

필드

- `DatabaseName` – UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.

- `TableName` – UTF-8 문자열입니다.

  열 통계를 생성할 테이블의 이름입니다.

- `Schedule` – 일정 객체입니다.

  CRON 구문에 지정된 열 통계 실행 일정입니다.

- `ColumnNameList` – UTF-8 문자열의 배열입니다.

  통계를 실행할 열 이름의 목록입니다.

- `CatalogID` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 카탈로그 ID 문자열입니다.

  데이터베이스가 있는 데이터 카탈로그의 ID입니다.

- `Role` – UTF-8 문자열입니다.

  열 통계를 실행하는 데 사용되는 역할입니다.

- `SampleSize` – 100 이하의 숫자(실수)입니다.

  샘플링할 데이터의 백분율입니다.

- `SecurityConfiguration` – 128바이트 이하 길이의 UTF-8 문자열입니다.

  CloudWatch 로그를 암호화하는 데 사용되는 보안 구성의 이름입니다.

- `ScheduleType` – UTF-8 문자열입니다(유효한 값: `CRON` | `AUTO`).

  열 통계 작업의 일정 유형입니다. 가능한 값은 `CRON` 또는 `AUTO`입니다.

- `SettingSource` – UTF-8 문자열입니다(유효한 값: `CATALOG` | `TABLE`).

  열 통계 작업 설정의 소스입니다. 가능한 값은 `CATALOG` 또는 `TABLE`입니다.

- `LastExecutionAttempt` – ExecutionAttempt 객체입니다.

열 통계 작업 실행의 마지막 ExecutionAttempt입니다.

ExecutionAttempt 구조

열 통계 작업 실행의 실행 시도입니다.

필드

- Status – UTF-8 문자열입니다(유효한 값: FAILED | STARTED).

  마지막 열 통계 작업 실행의 상태입니다.

- ColumnStatisticsTaskRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  마지막 열 통계 작업 실행의 작업 실행 ID입니다.

- ExecutionTimestamp – 타임스탬프입니다.

  마지막 열 통계 작업 실행이 발생한 타임스탬프입니다.

- ErrorMessage – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  마지막 열 통계 작업 실행과 관련한 오류 메시지입니다.

운영

- StartColumnStatisticsTaskRun 작업 (Python: start_column_statistics_task_run)
- GetColumnStatisticsTaskRun 작업 (Python: get_column_statistics_task_run)
- GetColumnStatisticsTaskRuns 작업 (Python: get_column_statistics_task_runs)
- ListColumnStatisticsTaskRuns 작업 (Python: list_column_statistics_task_runs)
- StopColumnStatisticsTaskRun 작업 (Python: stop_column_statistics_task_run)
- CreateColumnStatisticsTaskSettings 작업(Python: create_column_statistics_task_settings)
- UpdateColumnStatisticsTaskSettings 작업(Python: update_column_statistics_task_settings)
- GetColumnStatisticsTaskSettings 작업(Python: get_column_statistics_task_settings)
- DeleteColumnStatisticsTaskSettings 작업(Python: delete_column_statistics_task_settings)
- StartColumnStatisticsTaskRunSchedule 작업(Python: start_column_statistics_task_run_schedule)

- StopColumnStatisticsTaskRunSchedule 작업(Python: stop_column_statistics_task_run_schedule)

StartColumnStatisticsTaskRun 작업 (Python: start_column_statistics_task_run)

지정된 테이블 및 열에 대해 열 통계 작업 실행을 시작합니다.

요청

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계를 생성할 테이블의 이름입니다.

- `ColumnNameList` – UTF-8 문자열의 배열입니다.

  통계를 생성할 열 이름의 목록입니다. 제공되지 않은 경우 기본적으로 테이블의 모든 열 이름이 사용됩니다.

- `Role` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  서비스가 통계를 생성하기 위해 맡을 IAM 역할.

- `SampleSize` – 100 이하의 숫자(실수)입니다.

  통계 생성에 사용된 행의 비율. 제공되지 않은 경우 전체 테이블을 사용하여 통계를 생성합니다.

- `CatalogID` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 존재하는 데이터 카탈로그의 ID. 제공되지 않은 경우 기본적으로 AWS 계정 ID가 사용됩니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계 작업 실행의 CloudWatch 로그를 암호화하는 데 사용되는 보안 구성의 이름입니다.

응답

- `ColumnStatisticsTaskRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계 작업 실행의 식별자입니다.

오류

- `AccessDeniedException`
- `EntityNotFoundException`
- `ColumnStatisticsTaskRunningException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `InvalidInputException`

GetColumnStatisticsTaskRun 작업 (Python: get_column_statistics_task_run)

작업 실행 ID가 주어지면 작업 실행과 관련된 메타데이터/정보를 가져옵니다.

요청

- `ColumnStatisticsTaskRunId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  특정 열 통계 작업 실행의 식별자입니다.

응답

- `ColumnStatisticsTaskRun` – ColumnStatisticsTaskRun 객체입니다.

  열 통계 실행의 세부 정보를 나타내는 `ColumnStatisticsTaskRun` 객체입니다.

오류

- `EntityNotFoundException`
- `OperationTimeoutException`

- `InvalidInputException`

GetColumnStatisticsTaskRuns 작업 (Python: get_column_statistics_task_runs)

지정된 테이블과 관련된 모든 실행에 대한 정보를 검색합니다.

요청

- `DatabaseName` – 필수: UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  응답의 최대 크기입니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `ColumnStatisticsTaskRuns` – ColumnStatisticsTaskRun 객체의 배열입니다.

  실행된 작업의 열 통계 목록입니다.

- `NextToken` – UTF-8 문자열입니다.

  실행된 작업이 아직 모두 반환되지 않은 경우의 지속 토큰입니다.

오류

- `OperationTimeoutException`

ListColumnStatisticsTaskRuns 작업 (Python: list_column_statistics_task_runs)

특정 계정에 대해 실행된 모든 작업을 나열합니다.

요청

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  응답의 최대 크기입니다.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `ColumnStatisticsTaskRunIds` – 100개 이하의 문자열로 구성된 UTF-8 문자열입니다.

  열 통계 작업 실행 ID 목록.
- `NextToken` – UTF-8 문자열입니다.

  실행된 작업 ID가 아직 전부 반환되지 않은 경우의 지속 토큰입니다.

오류

- `OperationTimeoutException`

StopColumnStatisticsTaskRun 작업 (Python: stop_column_statistics_task_run)

지정된 테이블에 대한 작업 실행을 중지합니다.

요청

- `DatabaseName` – 필수: UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블의 이름

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `ColumnStatisticsTaskNotRunningException`
- `ColumnStatisticsTaskStoppingException`
- `OperationTimeoutException`

CreateColumnStatisticsTaskSettings 작업(Python: create_column_statistics_task_settings)

열 통계 작업에 대한 설정을 생성합니다.

요청

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계를 생성할 테이블의 이름입니다.
- `Role` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계를 실행하는 데 사용되는 역할입니다.
- `Schedule` – UTF-8 문자열입니다.

  CRON 구문에 지정된 열 통계 실행 일정입니다.
- `ColumnNameList` – UTF-8 문자열의 배열입니다.

  통계를 실행할 열 이름의 목록입니다.
- `SampleSize` – 100 이하의 숫자(실수)입니다.

  샘플링할 데이터의 백분율입니다.
- `CatalogID` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터베이스가 있는 데이터 카탈로그의 ID입니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  CloudWatch 로그를 암호화하는 데 사용되는 보안 구성의 이름입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  태그의 맵입니다.

응답

- 무응답 파라미터.

오류

- `AlreadyExistsException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ColumnStatisticsTaskRunningException`

UpdateColumnStatisticsTaskSettings 작업(Python: update_column_statistics_task_settings)

열 통계 작업의 설정을 업데이트합니다.

요청

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계를 생성할 테이블의 이름입니다.

- `Role` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계를 실행하는 데 사용되는 역할입니다.

- `Schedule` – UTF-8 문자열입니다.

  CRON 구문에 지정된 열 통계 실행 일정입니다.

- `ColumnNameList` – UTF-8 문자열의 배열입니다.

  통계를 실행할 열 이름의 목록입니다.

- `SampleSize` – 100 이하의 숫자(실수)입니다.

  샘플링할 데이터의 백분율입니다.

- `CatalogID` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터베이스가 있는 데이터 카탈로그의 ID입니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  CloudWatch 로그를 암호화하는 데 사용되는 보안 구성의 이름입니다.

응답

- 무응답 파라미터.

오류

- `AccessDeniedException`
- `EntityNotFoundException`
- `InvalidInputException`
- `VersionMismatchException`
- `OperationTimeoutException`

GetColumnStatisticsTaskSettings 작업(Python: get_column_statistics_task_settings)

열 통계 작업의 설정을 가져옵니다.

요청

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계를 검색할 테이블의 이름입니다.

응답

- `ColumnStatisticsTaskSettings` – ColumnStatisticsTaskSettings 객체입니다.

  열 통계 작업의 설정을 나타내는 `ColumnStatisticsTaskSettings` 객체입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`

DeleteColumnStatisticsTaskSettings 작업(Python: delete_column_statistics_task_settings)

열 통계 작업의 설정을 삭제합니다.

요청

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계를 삭제할 테이블의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`

StartColumnStatisticsTaskRunSchedule 작업(Python: start_column_statistics_task_run_schedule)

열 통계 작업 실행 일정을 시작합니다.

요청

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계 작업 실행 일정을 시작할 테이블의 이름입니다.

응답

- 무응답 파라미터.

오류

- `AccessDeniedException`
- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`

StopColumnStatisticsTaskRunSchedule 작업(Python: stop_column_statistics_task_run_schedule)

열 통계 작업 실행 일정을 중지합니다.

요청

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블이 있는 데이터베이스의 이름입니다.

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열 통계 작업 실행 일정을 중지할 테이블의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`

예외

- ColumnStatisticsTaskRunningException 구조

- ColumnStatisticsTaskNotRunningException 구조
- ColumnStatisticsTaskStoppingException 구조
- ColumnStatisticsTaskAutoConcurrencyLimitException 구조
- InvalidCatalogSettingException 구조

ColumnStatisticsTaskRunningException 구조

열 통계 생성 작업을 실행하는 동안 다른 작업을 시작하려고 하면 예외가 발생합니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ColumnStatisticsTaskNotRunningException 구조

실행 중인 작업이 없을 때 작업 실행을 중지하려고 하면 예외가 발생합니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ColumnStatisticsTaskStoppingException 구조

작업 실행을 중지하려고 할 때 발생하는 예외입니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ColumnStatisticsTaskAutoConcurrencyLimitException 구조

동시 자동 통계 작업의 한도에 이미 도달했을 때 발생하는 예외입니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

InvalidCatalogSettingException 구조

카탈로그 설정에 문제가 있을 때 발생하는 예외입니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

크롤러 스케줄러 API

크롤러 스케줄러 API는 크롤러를 생성, 삭제, 업데이트 및 나열하기 위한 API와 함께 AWS Glue 크롤러 데이터 유형에 대해 설명합니다.

데이터 타입

- 일정 구조

일정 구조

cron을 사용하여 객체의 일정을 정하여 이벤트의 일정을 정합니다.

필드

- ScheduleExpression – UTF-8 문자열입니다.

  일정을 지정하는 데 사용되는 cron 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 cron(15 12 * * ? *)을 지정합니다.
- State – UTF-8 문자열입니다(유효 값: SCHEDULED | NOT_SCHEDULED | TRANSITIONING).

  일정 상태

운영

- UpdateCrawlerSchedule 작업(Python: start_crawler_schedule)
- StartCrawlerSchedule 작업(Python: start_crawler_schedule)
- StopCrawlerSchedule 작업(Python: stop_crawler_schedule)

UpdateCrawlerSchedule 작업(Python: start_crawler_schedule)

`cron` 표현식을 사용하여 크롤러의 일정을 업데이트합니다.

요청

- `CrawlerName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트할 일정이 있는 크롤러 이름입니다.
- `Schedule` – UTF-8 문자열입니다.

  일정을 지정하는 데 사용되는 업데이트된 `cron` 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 `cron(15 12 * * ? *)`을 지정합니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `VersionMismatchException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

StartCrawlerSchedule 작업(Python: start_crawler_schedule)

크롤러가 실행 중이거나 일정 상태가 이미 `SCHEDULED`로 되어있지 않는 한 지정된 크롤러의 일정 상태를 `SCHEDULED`로 변경합니다.

요청

- `CrawlerName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  일정을 잡을 크롤러의 이름.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `SchedulerRunningException`
- `SchedulerTransitioningException`
- `NoScheduleException`
- `OperationTimeoutException`

StopCrawlerSchedule 작업(Python: stop_crawler_schedule)

지정된 크롤러의 일정 상태를 `NOT_SCHEDULED`로 설정하지만 크롤러가 실행되고 있으면 크롤러를 중지하지 않습니다.

요청

- `CrawlerName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  설정할 일정이 있는 크롤러 이름.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `SchedulerNotRunningException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

ETL 스크립트 API 자동 생성

ETL 스크립트 생성 API는 AWS Glue에서 ETL 스크립트를 생성하는 API와 데이터 유형에 대해 설명합니다.

데이터 타입

- CodeGenNode 구조
- CodeGenNodeArg 구조
- CodeGenEdge 구조
- 위치 구조
- CatalogEntry 구조
- MappingEntry 구조

CodeGenNode 구조

DAG(방향성 비순환 그래프)로 노드를 표시합니다

필드

- `Id` – 필수(Required): Identifier string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  노드 그래프 내 고유한 노드 식별자입니다.
- `NodeType` – 필수(Required): UTF-8 문자열입니다.

  노드 유형입니다.
- `Args` – 필수(Required): CodeGenNodeArg 객체의 배열이며 구조는 50개 이하입니다.

이름값 페어 형식인 노드의 속성

- `LineNumber` - 숫자(정수)입니다.

  노드의 행 수입니다.

CodeGenNodeArg 구조

인수 또는 노드의 속성.

필드

- `Name` – 필수(Required): UTF-8 문자열입니다.

  인수 또는 속성의 이름입니다.

- `Value` – 필수(Required): UTF-8 문자열입니다.

  인수 또는 속성의 값입니다.

- `Param` – 부울입니다.

  값이 파라미터로 사용되었다면 true입니다.

CodeGenEdge 구조

DAG(방향성 비순환 그래프)로 방향성 엣지를 표시합니다

필드

- `Source` – 필수(Required): Identifier string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  엣지가 시작하는 노드의 ID입니다.

- `Target` – 필수(Required): Identifier string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  엣지가 끝나는 노드의 ID입니다.

- `TargetParameter` – UTF-8 문자열입니다.

  엣지의 대상입니다.

위치 구조

리소스의 위치입니다.

필드

- `Jdbc` – CodeGenNodeArg 객체의 배열이며 구조는 50개 이하입니다.

  JDBC 위치
- `S3` – CodeGenNodeArg 객체의 배열이며 구조는 50개 이하입니다.

  Amazon Simple Storage Service(Amazon S3) 위치입니다.
- `DynamoDB` – CodeGenNodeArg 객체의 배열이며 구조는 50개 이하입니다.

  Amazon DynamoDB 테이블 위치입니다.

CatalogEntry 구조

AWS Glue Data Catalog에서 테이블 정의를 지정합니다.

필드

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  테이블 메타데이터가 상주하는 데이터베이스.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 테이블 이름입니다.

MappingEntry 구조

매핑 정의

필드

- `SourceTable` – UTF-8 문자열입니다.

  원본 테이블의 이름.

- `SourcePath` – UTF-8 문자열입니다.

  소스 경로입니다.
- `SourceType` – UTF-8 문자열입니다.

  소스 유형
- `TargetTable` – UTF-8 문자열입니다.

  대상 테이블은 입니다.
- `TargetPath` – UTF-8 문자열입니다.

  대상 경로
- `TargetType` – UTF-8 문자열입니다.

  대상 유형입니다.

운영

- CreateScript 작업(Python: create_script)
- GetDataflowGraph 작업(Python: get_dataflow_graph)
- GetMapping 작업(Python: get_mapping)
- GetPlan 작업(Python: get_plan)

CreateScript 작업(Python: create_script)

DAG(방향성 비순환 그래프)를 코드로 변환합니다.

요청

- `DagNodes` – CodeGenNode 객체의 배열입니다.

  DAG 노드 목록.
- `DagEdges` – CodeGenEdge 객체의 배열입니다.

  DAG 엣지 목록.
- `Language` – UTF-8 문자열입니다(유효 값: `PYTHON` | `SCALA`).

  DAG의 결과 코드 프로그래밍 언어.

응답

- `PythonScript` – UTF-8 문자열입니다.

  DAG에서 생성된 Python 스크립트입니다.
- `ScalaCode` – UTF-8 문자열입니다.

  DAG에서 생성된 Scala 코드입니다.

Errors

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetDataflowGraph 작업(Python: get_dataflow_graph)

DAG(방향성 비순환 그래프)로 Python 스크립트를 변환합니다.

요청

- `PythonScript` – UTF-8 문자열입니다.

  변환할 Python 스크립트

응답

- `DagNodes` – CodeGenNode 객체의 배열입니다.

  결과 DAG 노드 목록.
- `DagEdges` – CodeGenEdge 객체의 배열입니다.

  결과 DAG 엣지 목록.

Errors

- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`

GetMapping 작업(Python: get_mapping)

매핑 생성

요청

- `Source` – 필수(Required): CatalogEntry 객체입니다.

  원본 테입을 지정합니다.
- `Sinks` – CatalogEntry 객체의 배열입니다.

  대상 테이블 목록
- `Location` – 위치 객체입니다.

  매핑 파라미터입니다.

응답

- `Mapping` – 필수(Required): MappingEntry 객체의 배열입니다.

  지정한 대상 매핑 목록.

Errors

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

GetPlan 작업(Python: get_plan)

지정한 매핑을 실행할 코드 얻기.

요청

- `Mapping` – 필수(Required): MappingEntry 객체의 배열입니다.

원본 테이블에서 대상 테이블로 매핑하는 목록

- `Source` – 필수(Required): CatalogEntry 객체입니다.

  원본 테이블.

- `Sinks` – CatalogEntry 객체의 배열입니다.

  대상 테이블

- `Location` – 위치 객체입니다.

  매핑 파라미터입니다.

- `Language` – UTF-8 문자열입니다(유효 값: `PYTHON` | `SCALA`).

  매핑을 실행할 코드 프로그래밍 언어.

- `AdditionalPlanOptionsMap` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  추가 선택적 키-값 파라미터를 보유하는 맵입니다.

  현재 다음 키-값 페어가 지원됩니다.

  - `inferSchema` - AWS Glue 작업에 의해 생성된 기본 스크립트에 대해 `inferSchema`를 true로 설정할지 아니면 false로 설정할지 지정합니다. 예를 들어 `inferSchema`를 true로 설정하려면 다음 키 값 페어를 전달합니다.

    ```
    --additional-plan-options-map '{"inferSchema":"true"}'
    ```

응답

- `PythonScript` – UTF-8 문자열입니다.

  매핑할 Python 스크립트

- `ScalaCode` – UTF-8 문자열입니다.

  매핑을 실행할 Scala 코드.

Errors

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

시각적 작업 API

Visual Job API를 사용하면 AWS Glue 작업의 시각적 구성을 나타내는 JSON 객체에서 AWS Glue API를 사용하여 데이터 통합 작업을 생성할 수 있습니다.

생성된 작업에 대한 DAG를 AWS Glue Studio에 등록하고 연관된 코드를 생성하기 위해 생성 또는 업데이트 작업 API에 `CodeGenConfigurationNodes` 목록이 제공됩니다.

데이터 타입

- CodeGenConfigurationNode 구조
- JDBCConnectorOptions 구조
- StreamingDataPreviewOptions 구조
- AthenaConnectorSource 구조
- JDBCConnectorSource 구조
- SparkConnectorSource 구조
- CatalogSource 구조
- MySQLCatalogSource 구조
- PostgreSQLCatalogSource 구조
- OracleSQLCatalogSource 구조
- MicrosoftSQLServerCatalogSource 구조
- CatalogKinesisSource 구조
- DirectKinesisSource 구조
- KinesisStreamingSourceOptions 구조
- CatalogKafkaSource 구조
- DirectKafkaSource 구조

- KafkaStreamingSourceOptions 구조
- RedshiftSource 구조
- AmazonRedshiftSource 구조
- AmazonRedshiftNodeData 구조
- AmazonRedshiftAdvancedOption 구조
- 옵션 구조
- S3CatalogSource 구조
- S3SourceAdditionalOptions 구조
- S3CsvSource 구조
- DirectJDBCSource 구조
- S3DirectSourceAdditionalOptions 구조
- S3JsonSource 구조
- S3ParquetSource 구조
- S3DeltaSource 구조
- S3CatalogDeltaSource 구조
- CatalogDeltaSource 구조
- S3HudiSource 구조
- S3CatalogHudiSource 구조
- S3ExcelSource 구조
- CatalogHudiSource 구조
- DynamoDBCatalogSource 구조
- RelationalCatalogSource 구조
- JDBCConnectorTarget 구조
- SparkConnectorTarget 구조
- BasicCatalogTarget 구조
- MySQLCatalogTarget 구조
- PostgreSQLCatalogTarget 구조
- OracleSQLCatalogTarget 구조

- MicrosoftSQLServerCatalogTarget 구조
- RedshiftTarget 구조
- AmazonRedshiftTarget 구조
- UpsertRedshiftTargetOptions 구조
- S3CatalogTarget 구조
- S3GlueParquetTarget 구조
- CatalogSchemaChangePolicy 구조
- S3DirectTarget 구조
- S3HudiCatalogTarget 구조
- S3HudiDirectTarget 구조
- S3DeltaCatalogTarget 구조
- S3DeltaDirectTarget 구조
- S3HyperDirectTarget 구조
- S3IcebergDirectTarget 구조
- DirectSchemaChangePolicy 구조
- ApplyMapping 구조
- Mapping 구조
- SelectFields 구조
- DropFields 구조
- RenameField 구조
- Spigot 구조
- 조인 구조
- JoinColumn 구조
- SplitFields 구조
- SelectFromCollection 구조
- FillMissingValues 구조
- Filter 구조
- FilterExpression 구조

- FilterValue 구조
- CustomCode 구조
- SparkSQL 구조
- SqlAlias 구조
- DropNullFields 구조
- NullCheckBoxList 구조
- NullValueField 구조
- 데이터 형식 구조
- 병합 구조
- 결합 구조
- PIIDetection 구조
- 집계 구조
- DropDuplicates 구조
- GovernedCatalogTarget 구조
- GovernedCatalogSource 구조
- AggregateOperation 구조
- GlueSchema 구조
- GlueStudioSchemaColumn 구조
- GlueStudioColumn 구조
- DynamicTransform 구조
- TransformConfigParameter 구조
- EvaluateDataQuality 구조
- DQResultsPublishingOptions 구조
- DQStopJobOnFailureOptions 구조
- EvaluateDataQualityMultiFrame 구조
- 레시피 구조
- RecipeReference 구조
- SnowflakeNodeData 구조
- SnowflakeSource 구조

- SnowflakeTarget 구조
- ConnectorDataSource 구조
- ConnectorDataTarget 구조
- RecipeStep 구조
- RecipeAction 구조
- ConditionExpression 구조

CodeGenConfigurationNode 구조

`CodeGenConfigurationNode`는 유효한 모든 노드 유형을 열거합니다. 멤버 변수 중 하나만 채울 수 있습니다.

필드

- `AthenaConnectorSource` – AthenaConnectorSource 객체입니다.

  Amazon Athena 데이터 원본에 대한 커넥터를 지정합니다.
- `JDBCConnectorSource` – JDBCConnectorSource 객체입니다.

  JDBC 데이터 원본에 대한 커넥터를 지정합니다.
- `SparkConnectorSource` – SparkConnectorSource 객체입니다.

  Apache Spark 데이터 원본에 대한 커넥터를 지정합니다.
- `CatalogSource` – CatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 데이터 스토어를 지정합니다.
- `RedshiftSource` – RedshiftSource 객체입니다.

  Amazon Redshift 데이터 스토어를 지정합니다.
- `S3CatalogSource` – S3CatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 Amazon S3 데이터 스토어를 지정합니다.
- `S3CsvSource` – S3CsvSource 객체입니다.

  Amazon S3에 저장된 CSV(쉼표로 구분된 값) 데이터 스토어를 지정합니다.
- `S3ExcelSource` – S3ExcelSource 객체입니다.

Amazon S3에서 Excel 파일을 읽기 위한 구성 파라미터를 정의합니다.

- `S3JsonSource` – S3JsonSource 객체입니다.

  Amazon S3에 저장된 JSON 데이터 스토어를 지정합니다.
- `S3ParquetSource` – S3ParquetSource 객체입니다.

  Amazon S3에 저장된 Apache Parquet 데이터 스토어를 지정합니다.
- `RelationalCatalogSource` – RelationalCatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 관계형 카탈로그 데이터 소스를 지정합니다.
- `DynamoDBCatalogSource` – DynamoDBCatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 DynamoDBC 카탈로그 데이터 스토어를 지정합니다.
- `JDBCConnectorTarget` – JDBCConnectorTarget 객체입니다.

  Apache Parquet 열 형식 스토리지의 Amazon S3에 쓰는 데이터 대상을 지정합니다.
- `SparkConnectorTarget` – SparkConnectorTarget 객체입니다.

  Apache Spark 커넥터를 사용하는 대상을 지정합니다.
- `CatalogTarget` – BasicCatalogTarget 객체입니다.

  AWS Glue 데이터 카탈로그 테이블을 사용하는 대상을 지정합니다.
- `RedshiftTarget` – RedshiftTarget 객체입니다.

  Amazon Redshift를 사용하는 대상을 지정합니다.
- `S3CatalogTarget` – S3CatalogTarget 객체입니다.

  AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에 쓰는 데이터 대상을 지정합니다.
- `S3GlueParquetTarget` – S3GlueParquetTarget 객체입니다.

  Apache Parquet 열 형식 스토리지의 Amazon S3에 쓰는 데이터 대상을 지정합니다.
- `S3HyperDirectTarget` – S3HyperDirectTarget 객체입니다.

  HyperDirect 최적화를 사용하여 데이터를 Amazon S3에 쓰기 위한 구성 파라미터를 정의합니다.
- `S3DirectTarget` – S3DirectTarget 객체입니다.

  Amazon S3에 쓰는 데이터 대상을 지정합니다.

- `S3IcebergDirectTarget` – S3IcebergDirectTarget 객체입니다.

  데이터를 Amazon S3에 Apache Iceberg 테이블로 쓰기 위한 구성 파라미터를 정의합니다.
- `ApplyMapping` – ApplyMapping 객체입니다.

  데이터 원본의 데이터 속성 키를 데이터 대상의 데이터 속성 키에 매핑하는 변환을 지정합니다. 키의 이름을 바꾸고 키의 데이터 유형을 수정하고 데이터 집합에서 삭제할 키를 선택할 수 있습니다.
- `SelectFields` – SelectFields 객체입니다.

  유지할 데이터 속성 키를 선택하는 변환을 지정합니다.
- `DropFields` – DropFields 객체입니다.

  삭제할 데이터 속성 키를 선택하는 변환을 지정합니다.
- `RenameField` – RenameField 객체입니다.

  단일 데이터 속성 키의 이름을 바꾸는 변환을 지정합니다.
- `Spigot` – Spigot 객체입니다.

  Amazon S3 버킷에 데이터 샘플을 쓰는 변환을 지정합니다.
- `Join` – 조인 객체입니다.

  지정된 데이터 속성 키의 비교 구문을 사용하여 두 데이터 집합을 하나의 데이터 집합으로 조인하는 변환을 지정합니다. 내부, 외부, 왼쪽, 오른쪽, 왼쪽 반 및 왼쪽 안티 조인을 사용할 수 있습니다.
- `SplitFields` – SplitFields 객체입니다.

  데이터 속성 키를 두 개의 `DynamicFrames`로 분할하는 변환을 지정합니다. 출력은 `DynamicFrames` 컬렉션입니다. 하나에는 선택한 데이터 속성 키가 있고 다른 하나에는 나머지 데이터 속성 키가 있습니다.
- `SelectFromCollection` – SelectFromCollection 객체입니다.

  `DynamicFrames` 컬렉션에서 하나의 `DynamicFrame`을 선택하는 변환을 지정합니다. 출력은 선택한 `DynamicFrame`입니다.
- `FillMissingValues` – FillMissingValues 객체입니다.

  데이터 집합에서 누락된 값이 있는 레코드를 찾고 대체를 통해 결정된 값으로 새 필드를 추가하는 변환을 지정합니다. 입력 데이터 집합은 누락 값을 결정하는 기계 학습 모델을 훈련하는 데 사용됩니다.
- `Filter` – Filter 객체입니다.

필터 조건에 따라 하나의 데이터 집합을 두 개로 분할하는 변환을 지정합니다.

- `CustomCode` – CustomCode 객체입니다.

  제공한 사용자 지정 코드를 사용하여 데이터 변환을 수행하는 변환을 지정합니다. 출력은 DynamicFrames의 컬렉션입니다.

- `SparkSQL` – SparkSQL 객체입니다.

  데이터를 변환하기 위해 Spark SQL 구문을 사용하여 SQL 쿼리를 입력하는 변환을 지정합니다. 출력은 단일 `DynamicFrame`입니다.

- `DirectKinesisSource` – DirectKinesisSource 객체입니다.

  직접적인 Amazon Kinesis 데이터 원본을 지정합니다.

- `DirectKafkaSource` – DirectKafkaSource 객체입니다.

  Apache Kafka 데이터 스토어를 지정합니다.

- `CatalogKinesisSource` – CatalogKinesisSource 객체입니다.

  AWS Glue 데이터 카탈로그의 Kinesis 데이터 원본을 지정합니다.

- `CatalogKafkaSource` – CatalogKafkaSource 객체입니다.

  데이터 카탈로그의 Apache Kafka 데이터 스토어를 지정합니다.

- `DropNullFields` – DropNullFields 객체입니다.

  열의 모든 값이 'null'인 경우 데이터 집합에서 열을 제거하는 변환을 지정합니다. 기본값으로 AWS Glue Studio는 Null 객체를 인식하지만 빈 문자열, 'null'인 문자열, -1 정수 또는 0과 같은 다른 자리 표시자 등의 일부 값은 자동으로 Null로 인식되지 않습니다.

- `Merge` – 병합 객체입니다.

  레코드를 식별하기 위해 지정된 기본 키를 기준으로 `DynamicFrame`을 스테이징 `DynamicFrame`과 병합하는 변환을 지정합니다. 중복 레코드(기본 키가 동일한 레코드)는 중복 제거되지 않습니다.

- `Union` – Union 객체입니다.

  둘 이상 데이터 집합의 행을 단일 결과로 결합하는 변환을 지정합니다.

- `PIIDetection` – PIIDetection 객체입니다.

  PII 데이터를 식별, 제거 또는 마스킹하는 변환을 지정합니다.

- `Aggregate` – Aggregate 객체입니다.

선택한 필드별로 행을 그룹화하고 지정된 함수에 의해 집계된 값을 계산하는 변환을 지정합니다.

- `DropDuplicates` – DropDuplicates 객체입니다.

  데이터세트에서 반복 데이터의 행을 제거하는 변환을 지정합니다.

- `GovernedCatalogTarget` – GovernedCatalogTarget 객체입니다.

  관리 카탈로그에 작성하는 데이터 대상을 지정합니다.

- `GovernedCatalogSource` – GovernedCatalogSource 객체입니다.

  관리 데이터 카탈로그의 데이터 소스를 지정합니다.

- `MicrosoftSQLServerCatalogSource` – MicrosoftSQLServerCatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 Microsoft SQL 서버 데이터 소스를 지정합니다.

- `MySQLCatalogSource` – MySQLCatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 MySQL 데이터 소스를 지정합니다.

- `OracleSQLCatalogSource` – OracleSQLCatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 Oracle 데이터 소스를 지정합니다.

- `PostgreSQLCatalogSource` – PostgreSQLCatalogSource 객체입니다.

  AWS Glue 데이터 카탈로그의 PostgresSQL 데이터 소스를 지정합니다.

- `MicrosoftSQLServerCatalogTarget` – MicrosoftSQLServerCatalogTarget 객체입니다.

  Microsoft SQL을 사용하는 대상을 지정합니다.

- `MySQLCatalogTarget` – MySQLCatalogTarget 객체입니다.

  MySQL을 사용하는 대상을 지정합니다.

- `OracleSQLCatalogTarget` – OracleSQLCatalogTarget 객체입니다.

  Oracle SQL을 사용하는 대상을 지정합니다.

- `PostgreSQLCatalogTarget` – PostgreSQLCatalogTarget 객체입니다.

  Postgres SQL을 사용하는 대상을 지정합니다.

- `DynamicTransform` – DynamicTransform 객체입니다.

  사용자가 생성한 사용자 지정 시각적 변환을 지정합니다.

- `EvaluateDataQuality` – EvaluateDataQuality 객체입니다.

  데이터 품질 평가 기준을 지정합니다.

- `S3CatalogHudiSource` – S3CatalogHudiSource 객체입니다.

  AWS Glue 데이터 카탈로그에 등록된 Hudi 데이터 소스를 지정합니다. 데이터 소스를 Amazon S3에 저장해야 합니다.

- `CatalogHudiSource` – CatalogHudiSource 객체입니다.

  AWS Glue 데이터 카탈로그에 등록된 Hudi 데이터 소스를 지정합니다.

- `S3HudiSource` – S3HudiSource 객체입니다.

  Amazon S3에 저장된 Hudi 데이터 소스를 지정합니다.

- `S3HudiCatalogTarget` – S3HudiCatalogTarget 객체입니다.

  AWS Glue 데이터 카탈로그의 Hudi 데이터 소스에 작성하는 데이터 대상을 지정합니다.

- `S3HudiDirectTarget` – S3HudiDirectTarget 객체입니다.

  Amazon S3에서 Hudi 데이터 소스에 작성하는 대상을 지정합니다.

- `S3CatalogDeltaSource` – S3CatalogDeltaSource 객체입니다.

  AWS Glue 데이터 카탈로그에 등록된 Delta Lake 데이터 소스를 지정합니다. 데이터 소스를 Amazon S3에 저장해야 합니다.

- `CatalogDeltaSource` – CatalogDeltaSource 객체입니다.

  AWS Glue 데이터 카탈로그에 등록된 Delta Lake 데이터 소스를 지정합니다.

- `S3DeltaSource` – S3DeltaSource 객체입니다.

  Amazon S3에 저장된 Delta Lake 데이터 소스를 지정합니다.

- `S3DeltaCatalogTarget` – S3DeltaCatalogTarget 객체입니다.

  AWS Glue 데이터 카탈로그의 Delta Lake 데이터 소스에 작성하는 데이터 대상을 지정합니다.

- `S3DeltaDirectTarget` – S3DeltaDirectTarget 객체입니다.

  Amazon S3에서 Delta Lake 데이터 소스에 작성하는 대상을 지정합니다.

- `AmazonRedshiftSource` – AmazonRedshiftSource 객체입니다.

  Amazon Redshift에서 데이터 소스에 작성하는 대상을 지정합니다.

- `AmazonRedshiftTarget` – AmazonRedshiftTarget 객체입니다.

  Amazon Redshift에서 데이터 대상에 작성하는 대상을 지정합니다.

- `EvaluateDataQualityMultiFrame` – EvaluateDataQualityMultiFrame 객체입니다.

  데이터 품질 평가 기준을 지정합니다. 여러 입력 데이터를 허용하고 동적 프레임 컬렉션을 반환합니다.

- `Recipe` – 레시피 객체입니다.

  AWS Glue DataBrew 레시피 노드를 지정합니다.

- `SnowflakeSource` – SnowflakeSource 객체입니다.

  Snowflake 데이터 소스를 지정합니다.

- `SnowflakeTarget` – SnowflakeTarget 객체입니다.

  Snowflake 데이터 소스에 작성하는 대상을 지정합니다.

- `ConnectorDataSource` – ConnectorDataSource 객체입니다.

  표준 연결 옵션으로 생성된 소스를 지정합니다.

- `ConnectorDataTarget` – ConnectorDataTarget 객체입니다.

  표준 연결 옵션으로 생성된 대상을 지정합니다.

JDBCConnectorOptions 구조

커넥터에 대한 추가 연결 옵션입니다.

필드

- `FilterPredicate` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  소스에서 데이터를 필터링하기 위한 추가 조건 절입니다. 예:

  ```
  BillingCity='Mountain View'
  ```

  테이블 이름 대신 쿼리를 사용하는 경우 쿼리가 지정된 `filterPredicate`에서 작동하는지 검증해야 합니다.

- `PartitionColumn` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

분할에 사용되는 정수 열의 이름입니다. 이 옵션은 lowerBound, upperBound 및 numPartitions에 포함되는 경우에만 작동합니다. 이 옵션은 Spark SQL JDBC 리더에서와 같은 방식으로 작동합니다.

- LowerBound – None 이하의 숫자(정수)입니다.

  파티션 스트라이드를 결정하는 데 사용되는 partitionColumn의 최소값입니다.

- UpperBound – None 이하의 숫자(정수)입니다.

  파티션 스트라이드를 결정하는 데 사용되는 partitionColumn의 최대값입니다.

- NumPartitions – None 이하의 숫자(정수)입니다.

  파티션 수입니다. 이 값은 lowerBound(포함) 및 upperBound(배타)와 함께 partitionColumn을 분할하는 데 사용되는 생성된 WHERE 절 표현에 대한 파티션 스트라이드를 형성합니다.

- JobBookmarkKeys – UTF-8 문자열의 배열입니다.

  정렬할 작업 북마크 키의 이름입니다.

- JobBookmarkKeysSortOrder – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  오름차순 또는 내림차순 정렬 순서를 지정합니다.

- DataTypeMapping – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다(유효한 값: ARRAY | BIGINT | BINARY | BIT | BLOB | BOOLEAN | CHAR | CLOB | DATALINK | DATE | DECIMAL | DISTINCT | DOUBLE | FLOAT | INTEGER | JAVA_OBJECT | LONGNVARCHAR | LONGVARBINARY | LONGVARCHAR | NCHAR | NCLOB | NULL | NUMERIC | NVARCHAR | OTHER | REAL | REF | REF_CURSOR | ROWID | SMALLINT | SQLXML | STRUCT | TIME | TIME_WITH_TIMEZONE | TIMESTAMP | TIMESTAMP_WITH_TIMEZONE | TINYINT | VARBINARY | VARCHAR).

  각 값은 UTF-8 문자열입니다(유효한 값: DATE | STRING | TIMESTAMP | INT | FLOAT | LONG | BIGDECIMAL | BYTE | SHORT | DOUBLE).

  JDBC 데이터 유형에서 AWS Glue 데이터 유형으로의 매핑을 구축하는 사용자 지정 데이터 유형 매핑입니다. 예를 들어 "dataTypeMapping":{"FLOAT":"STRING"} 옵션은 드라이버의 ResultSet.getString() 메서드를 호출하여 JDBC 유형 FLOAT의 데이터 필드를 Java String 유형으로 매핑하고 이를 AWS Glue 레코드를 구축하는 데 사용합니다. ResultSet 객체는 각 드라이버에 의해 구현되므로 동작은 사용하는 드라이버에 따라 다릅니다. 드라이버가 변환을 수행하는 방법을 이해하려면 JDBC 드라이버에 대한 설명서를 참조하세요.

StreamingDataPreviewOptions 구조

데이터 샘플을 보기 위한 데이터 미리 보기와 관련된 옵션을 지정합니다.

필드

- `PollingTime` - 최소 10 이상의 숫자(long)입니다.

  밀리초 단위의 폴링 시간입니다.
- `RecordPollingLimit` - 최소 1 이상의 숫자(long)입니다.

  폴링되는 레코드 수에 대한 제한입니다.

AthenaConnectorSource 구조

Amazon Athena 데이터 원본에 대한 커넥터를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.
- `ConnectionName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터와 연관된 연결 이름입니다.
- `ConnectorName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  AWS Glue Studio에서 데이터 스토어에 액세스하는 데 도움이 되는 커넥터의 이름입니다.
- `ConnectionType` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Amazon Athena 데이터 스토어에 대한 연결을 지정하는 marketplace.athena 또는 custom.athena와 같은 연결 유형입니다.
- `ConnectionTable` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본에 있는 테이블의 이름입니다.
- `SchemaName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 CloudWatch 로그 그룹의 이름입니다. 예: `/aws-glue/jobs/output`.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

사용자 지정 Athena 소스에 대한 데이터 스키마를 지정합니다.

JDBCConnectorSource 구조

JDBC 데이터 원본에 대한 커넥터를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.

- `ConnectionName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터와 연관된 연결 이름입니다.

- `ConnectorName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  AWS Glue Studio에서 데이터 스토어에 액세스하는 데 도움이 되는 커넥터의 이름입니다.

- `ConnectionType` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  JDBC 데이터 스토어에 대한 연결을 지정하는 marketplace.jdbc 또는 custom.jdbc와 같은 연결 유형입니다.

- `AdditionalOptions` – JDBCConnectorOptions 객체입니다.

  커넥터에 대한 추가 연결 옵션입니다.

- `ConnectionTable` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본에 있는 테이블의 이름입니다.

- `Query` – Custom string pattern #60과(와) 일치하는 UTF-8 문자열입니다.

  데이터를 가져올 테이블 또는 SQL 쿼리입니다. `ConnectionTable` 또는 `query`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  사용자 지정 JDBC 소스에 대한 데이터 스키마를 지정합니다.

SparkConnectorSource 구조

Apache Spark 데이터 원본에 대한 커넥터를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.
- `ConnectionName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터와 연관된 연결 이름입니다.
- `ConnectorName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  AWS Glue Studio에서 데이터 스토어에 액세스하는 데 도움이 되는 커넥터의 이름입니다.
- `ConnectionType` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Apache Spark 데이터 스토어에 대한 연결을 지정하는 marketplace.spark 또는 custom.spark와 같은 연결 유형입니다.
- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션입니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  사용자 지정 Spark 소스에 대한 데이터 스키마를 지정합니다.

CatalogSource 구조

AWS Glue 데이터 카탈로그의 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

MySQLCatalogSource 구조

AWS Glue 데이터 카탈로그의 MySQL 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

PostgreSQLCatalogSource 구조

AWS Glue 데이터 카탈로그의 PostgresSQL 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

OracleSQLCatalogSource 구조

AWS Glue 데이터 카탈로그의 Oracle 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

MicrosoftSQLServerCatalogSource 구조

AWS Glue 데이터 카탈로그의 Microsoft SQL 서버 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

CatalogKinesisSource 구조

AWS Glue 데이터 카탈로그의 Kinesis 데이터 원본을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

데이터 원본의 이름입니다.

- `WindowSize` – None 이하의 숫자(정수)입니다.

  각 마이크로 배치를 처리하는 데 사용할 시간입니다.

- `DetectSchema` – 부울입니다.

  들어오는 데이터의 스키마를 자동으로 결정할지 여부입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.

- `StreamingOptions` – KinesisStreamingSourceOptions 객체입니다.

  Kinesis 스트리밍 데이터 원본에 대한 추가 옵션입니다.

- `DataPreviewOptions` – StreamingDataPreviewOptions 객체입니다.

  데이터 미리 보기에 대한 추가 옵션입니다.

DirectKinesisSource 구조

직접적인 Amazon Kinesis 데이터 원본을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.

- `WindowSize` – None 이하의 숫자(정수)입니다.

  각 마이크로 배치를 처리하는 데 사용할 시간입니다.

- `DetectSchema` – 부울입니다.

  들어오는 데이터의 스키마를 자동으로 결정할지 여부입니다.

- `StreamingOptions` – KinesisStreamingSourceOptions 객체입니다.

  Kinesis 스트리밍 데이터 원본에 대한 추가 옵션입니다.

- `DataPreviewOptions` – StreamingDataPreviewOptions 객체입니다.

  데이터 미리 보기에 대한 추가 옵션입니다.

KinesisStreamingSourceOptions 구조

Amazon Kinesis 스트리밍 데이터 원본에 대한 추가 옵션입니다.

필드

- `EndpointUrl` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Kinesis 엔드포인트의 URL입니다.

- `StreamName` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Kinesis 데이터 스트림의 이름입니다.

- `Classification` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  선택적 분류입니다.

- `Delimiter` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  구분 기호 문자열을 지정합니다.

- `StartingPosition` – UTF-8 문자열입니다(유효한 값: `latest="LATEST"` | `trim_horizon="TRIM_HORIZON"` | `earliest="EARLIEST"` | `timestamp="TIMESTAMP"`).

  데이터를 읽을 Kinesis 데이터 스트림의 시작 위치입니다. 가능한 값은 `"latest"`, `"trim_horizon"`, `"earliest"` 또는 `yyyy-mm-ddTHH:MM:SSZ` 패턴에서 UTC 형식의 타임스탬프 문자열입니다(여기서, `Z`는 UTC 시간대 오프셋(+/-)임, 예: '2023-04-04T08:00:00-04:00'). 기본값은 `"latest"`입니다.

  참고: 'startingPosition'에서 UTC 형식의 타임스탬프 문자열 값을 사용하는 것은 AWS Glue 버전 4.0 이상에서만 지원됩니다.

- `MaxFetchTimeInMs` – None 이하의 숫자(정수)입니다.

  작업 실행기가 Kinesis 데이터 스트림에서 현재 배치에 대한 레코드를 읽는 데 걸리는 최대 시간(밀리초(ms) 단위로 지정)입니다. 이 시간 내에 여러 개의 `GetRecords` API 호출을 할 수 있습니다. 기본값은 `1000`입니다.

- `MaxFetchRecordsPerShard` – None 이하의 숫자(정수)입니다.

마이크로 배치에 따라 Kinesis 데이터 스트림에서 샤드당 가져올 최대 레코드 수입니다. 참고: 스트리밍 작업이 이미 Kinesis의 동일한 get-records 호출에서 추가 레코드를 읽은 경우 클라이언트가 이 제한을 초과할 수 있습니다. `MaxFetchRecordsPerShard`가 엄격해야 한다면 `MaxRecordPerRead`의 배수여야 합니다. 기본값은 `100000`입니다.

- `MaxRecordPerRead` – None 이하의 숫자(정수)입니다.

  각 getRecords 작업에서 Kinesis 데이터 스트림으로부터 가져올 최대 레코드 수입니다. 기본값은 `10000`입니다.

- `AddIdleTimeBetweenReads` – 부울입니다.

  두 개의 연속 getRecords 작업 사이에 시간 지연을 추가합니다. 기본값은 `"False"`입니다. 이 옵션은 AWS Glue 버전 2.0 이상에서만 구성할 수 있습니다.

- `IdleTimeBetweenReadsInMs` – None 이하의 숫자(정수)입니다.

  두 개의 연속 getRecords 작업 사이의 최소 시간 지연으로, ms 단위로 지정됩니다. 기본값은 `1000`입니다. 이 옵션은 AWS Glue 버전 2.0 이상에서만 구성할 수 있습니다.

- `DescribeShardInterval` – None 이하의 숫자(정수)입니다.

  스크립트가 리샤딩을 고려하기 위한 두 개의 ListShards API 호출 사이의 최소 시간 간격입니다. 기본값은 `1s`입니다.

- `NumRetries` – None 이하의 숫자(정수)입니다.

  Kinesis Data Streams API 요청의 최대 재시도 횟수입니다. 기본값은 `3`입니다.

- `RetryIntervalMs` – None 이하의 숫자(정수)입니다.

  Kinesis Data Streams API 호출을 재시도하기 전의 휴지 기간(ms 단위로 지정)입니다. 기본값은 `1000`입니다.

- `MaxRetryIntervalMs` – None 이하의 숫자(정수)입니다.

  Kinesis Data Streams API 호출을 두 번 재시도하는 사이의 최대 휴지 시간(ms 단위로 지정)입니다. 기본값은 `10000`입니다.

- `AvoidEmptyBatches` – 부울입니다.

  배치가 시작되기 전에 Kinesis 데이터 스트림에서 읽지 않은 데이터를 확인하여 빈 마이크로 배치 작업 생성을 방지합니다. 기본값은 `"False"`입니다.

- `StreamArn` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

Kinesis 데이터 스트림의 Amazon 리소스 이름(ARN)입니다.

- `RoleArn` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  AWS Security Token Service(AWS STS)를 사용하여 맡을 역할의 Amazon 리소스 이름(ARN)입니다. 이 역할에는 Kinesis 데이터 스트림에 대한 레코드 작업을 설명하거나 읽을 수 있는 권한이 있어야 합니다. 다른 계정의 데이터 스트림에 액세스할 때 이 파라미터를 사용해야 합니다. `"awsSTSSessionName"`과(와) 함께 사용합니다.

- `RoleSessionName` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  AWS STS를 사용하여 역할을 맡는 세션의 식별자입니다. 다른 계정의 데이터 스트림에 액세스할 때 이 파라미터를 사용해야 합니다. `"awsSTSRoleARN"`과(와) 함께 사용합니다.

- `AddRecordTimestamp` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  이 옵션이 'true'로 설정되면 데이터 출력에는 이름이 '__src_timestamp'라는 추가 열이 포함됩니다. 이 열은 스트림에서 해당 레코드를 수신한 시간을 나타냅니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.

- `EmitConsumerLagMetrics` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  이 옵션을 'true'로 설정하면 각 배치에 대해 스트림에서 수신한 가장 오래된 레코드와 AWS Glue에 도착한 시간 사이의 지표를 CloudWatch로 내보냅니다. 지표의 이름은 'glue.driver.streaming.maxConsumerLagInMs'입니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.

- `StartingTimestamp` – UTF-8 문자열입니다.

  Kinesis 데이터 스트림에서 데이터 읽기를 시작하는 레코드의 타임스탬프입니다. 가능한 값은 `yyyy-mm-ddTHH:MM:SSZ` 패턴에서 UTC 형식의 타임스탬프 문자열입니다(여기서, Z는 UTC 시간대 오프셋(+/-)임, 예: '2023-04-04T08:00:00+08:00').

CatalogKafkaSource 구조

데이터 카탈로그의 Apache Kafka 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.

- `WindowSize` – None 이하의 숫자(정수)입니다.

  각 마이크로 배치를 처리하는 데 사용할 시간입니다.

- `DetectSchema` – 부울입니다.

  들어오는 데이터의 스키마를 자동으로 결정할지 여부입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.

- `StreamingOptions` – KafkaStreamingSourceOptions 객체입니다.

  스트리밍 옵션을 지정합니다.

- `DataPreviewOptions` – StreamingDataPreviewOptions 객체입니다.

  데이터 샘플을 보기 위한 데이터 미리 보기와 관련된 옵션을 지정합니다.

DirectKafkaSource 구조

Apache Kafka 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.

- `StreamingOptions` – KafkaStreamingSourceOptions 객체입니다.

  스트리밍 옵션을 지정합니다.

- `WindowSize` – None 이하의 숫자(정수)입니다.

  각 마이크로 배치를 처리하는 데 사용할 시간입니다.

- `DetectSchema` – 부울입니다.

  들어오는 데이터의 스키마를 자동으로 결정할지 여부입니다.

- `DataPreviewOptions` – StreamingDataPreviewOptions 객체입니다.

데이터 샘플을 보기 위한 데이터 미리 보기와 관련된 옵션을 지정합니다.

KafkaStreamingSourceOptions 구조

스트리밍에 대한 추가 옵션입니다.

필드

- `BootstrapServers` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  부트스트랩 서버 URL 목록입니다(예: `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`). 이 옵션은 API 호출에 지정하거나 데이터 카탈로그의 테이블 메타데이터에 정의해야 합니다.
- `SecurityProtocol` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  브로커와 통신하는 데 사용되는 프로토콜입니다. 가능한 값은 `"SSL"` 또는 `"PLAINTEXT"`입니다.
- `ConnectionName` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  연결의 이름입니다.
- `TopicName` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Apache Kafka에 지정된 주제 이름입니다. `"topicName"`, `"assign"` 또는 `"subscribePattern"` 중 하나 이상을 지정해야 합니다.
- `Assign` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  사용할 특정 `TopicPartitions`입니다. `"topicName"`, `"assign"` 또는 `"subscribePattern"` 중 하나 이상을 지정해야 합니다.
- `SubscribePattern` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  구독할 주제 목록을 식별하는 Java 정규식 문자열입니다. `"topicName"`, `"assign"` 또는 `"subscribePattern"` 중 하나 이상을 지정해야 합니다.
- `Classification` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  선택적 분류입니다.
- `Delimiter` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  구분 기호 문자열을 지정합니다.

- `StartingOffsets` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터를 읽을 Kafka 주제의 시작 위치입니다. 가능한 값은 `"earliest"` 또는 `"latest"`입니다. 기본값은 `"latest"`입니다.
- `EndingOffsets` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  배치 쿼리가 종료되는 엔드포인트입니다. 가능한 값은 `"latest"` 또는 각 `TopicPartition`의 끝 오프셋을 지정하는 JSON 문자열입니다.
- `PollTimeoutMs` – None 이하의 숫자(정수)입니다.

  Spark 작업 실행기에서 Kafka의 데이터를 폴링하는 시간 제한(밀리초)입니다. 기본값은 `512`입니다.
- `NumRetries` – None 이하의 숫자(정수)입니다.

  Kafka 오프셋 가져오기에 실패하기 전에 재시도할 횟수입니다. 기본값은 `3`입니다.
- `RetryIntervalMs` – None 이하의 숫자(정수)입니다.

  Kafka 오프셋 가져오기를 재시도하기 전에 대기할 시간(밀리초)입니다. 기본값은 `10`입니다.
- `MaxOffsetsPerTrigger` – None 이하의 숫자(정수)입니다.

  트리거 간격당 처리되는 최대 오프셋 수에 대한 속도 제한입니다. 지정된 총 오프셋 수는 서로 다른 볼륨의 `topicPartitions`에 비례하여 분할됩니다. 기본값은 null입니다. 즉, 소비자가 알려진 최신 오프셋까지 모든 오프셋을 읽습니다.
- `MinPartitions` – None 이하의 숫자(정수)입니다.

  Kafka에서 읽을 원하는 최소 파티션 수입니다. 기본값은 null이며 이는 Spark 파티션의 수가 Kafka 파티션의 수와 동일함을 의미합니다.
- `IncludeHeaders` – 부울입니다.

  Kafka 헤더를 포함할지 여부입니다. 옵션이 "true"로 설정되면 데이터 출력에는 유형이 `Array[Struct(key: String, value: String)]`인 "glue_streaming_kafka_headers"라는 추가 열이 포함됩니다. 기본값은 "false"입니다. 이 옵션은 AWS Glue 버전 3.0 이상에서만 사용할 수 있습니다.
- `AddRecordTimestamp` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  이 옵션이 'true'로 설정되면 데이터 출력에는 이름이 '__src_timestamp'라는 추가 열이 포함됩니다. 이 열은 주제에서 해당 레코드를 수신한 시간을 나타냅니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.
- `EmitConsumerLagMetrics` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

이 옵션을 'true'로 설정하면 각 배치에 대해 주제에서 수신한 가장 오래된 레코드와 AWS Glue에 도착한 시간 사이의 지표를 CloudWatch로 내보냅니다. 지표의 이름은 'glue.driver.streaming.maxConsumerLagInMs'입니다. 기본값은 'false'입니다. 이 옵션은 AWS Glue 버전 4.0 이상에서 지원됩니다.

- `StartingTimestamp` – UTF-8 문자열입니다.

  Kafka 주제에서 데이터 읽기를 시작하는 레코드의 타임스탬프입니다. 가능한 값은 `yyyy-mm-ddTHH:MM:SSZ` 패턴에서 UTC 형식의 타임스탬프 문자열입니다(여기서, Z는 UTC 시간대 오프셋(+/-)임, 예: '2023-04-04T08:00:00+08:00').

  `StartingTimestamp` 또는 `StartingOffsets` 중 하나만 설정해야 합니다.

RedshiftSource 구조

Amazon Redshift 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Amazon Redshift 데이터 스토어의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽어야 할 데이터베이스입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽어야 할 데이터베이스 테이블입니다.

- `RedshiftTmpDir` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터베이스 외부에서 복사할 때 임시 데이터를 스테이징할 수 있는 Amazon S3 경로입니다.

- `TmpDirIAMRole` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  권한이 있는 IAM 역할입니다.

AmazonRedshiftSource 구조

Amazon Redshift 소스를 지정합니다.

필드

- `Name` – Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Amazon Redshift 소스의 이름입니다.
- `Data` – AmazonRedshiftNodeData 객체입니다.

  Amazon Resshift 소스 노드의 데이터를 지정합니다.

AmazonRedshiftNodeData 구조

Amazon Redshift 노드를 지정합니다.

필드

- `AccessType` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  Redshift 연결을 위한 액세스 유형입니다. 직접 연결 또는 카탈로그 연결일 수 있습니다.
- `SourceType` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  특정 테이블이 소스인지 또는 사용자 지정 쿼리인지를 지정하기 위한 소스 유형입니다.
- `Connection` – 옵션 객체입니다.

  Redshift 클러스터에 대한 AWS Glue 연결입니다.
- `Schema` – 옵션 객체입니다.

  직접 연결로 작업하는 경우 Redshift 스키마 이름입니다.
- `Table` – 옵션 객체입니다.

  직접 연결로 작업하는 경우 Redshift 테이블 이름입니다.
- `CatalogDatabase` – 옵션 객체입니다.

  데이터 카탈로그로 작업하는 경우 AWS Glue 데이터 카탈로그 데이터베이스의 이름입니다.
- `CatalogTable` – 옵션 객체입니다.

  데이터 카탈로그로 작업하는 경우 AWS Glue 데이터 카탈로그 테이블의 이름입니다.
- `CatalogRedshiftSchema` – UTF-8 문자열입니다.

  데이터 카탈로그로 작업하는 경우 Redshift 스키마 이름입니다.

- `CatalogRedshiftTable` – UTF-8 문자열입니다.

  읽어야 할 데이터베이스 테이블입니다.

- `TempDir` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터베이스 외부에서 복사할 때 임시 데이터를 스테이징할 수 있는 Amazon S3 경로입니다.

- `IamRole` – 옵션 객체입니다.

  선택 사항. S3에 연결할 때 사용하는 역할 이름입니다. 비어 있는 경우 IAM 역할은 기본적으로 작업의 역할을 사용합니다.

- `AdvancedOptions` – AmazonRedshiftAdvancedOption 객체의 배열입니다.

  Redshift 클러스터에 연결하는 경우 선택적 값입니다.

- `SampleQuery` – UTF-8 문자열입니다.

  SourceType이 '쿼리'인 경우 Redshift 소스에서 데이터를 가져오는 데 사용되는 SQL입니다.

- `PreAction` – UTF-8 문자열입니다.

  업서트와 함께 MERGE 또는 APPEND를 실행하기 전에 사용되는 SQL입니다.

- `PostAction` – UTF-8 문자열입니다.

  업서트와 함께 MERGE 또는 APPEND를 실행하기 전에 사용되는 SQL입니다.

- `Action` – UTF-8 문자열입니다.

  Redshift 클러스터에 쓰는 방법을 지정합니다.

- `TablePrefix` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  테이블의 접두사를 지정합니다.

- `Upsert` – 부울입니다.

  APPEND를 수행하는 경우 Redshift 싱크에서 사용되는 작업입니다.

- `MergeAction` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  Redshift 싱크에서 MERGE 처리 방식을 결정할 때 사용되는 작업입니다.

- `MergeWhenMatched` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  기존 레코드가 새 레코드와 일치하는 경우 Redshift 싱크에서 MERGE 처리 방식을 결정할 때 사용되는 작업입니다.

- `MergeWhenNotMatched` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  기존 레코드가 새 레코드와 일치하지 않는 경우 Redshift 싱크에서 MERGE 처리 방식을 결정할 때 사용되는 작업입니다.

- `MergeClause` – UTF-8 문자열입니다.

  일치하는 레코드를 처리하기 위해 사용자 지정 병합에 사용되는 SQL입니다.

- `CrawlerConnection` – UTF-8 문자열입니다.

  사용된 카탈로그 테이블과 연관된 연결 이름을 지정합니다.

- `TableSchema` – 옵션 객체의 배열입니다.

  지정된 노드에 대한 스키마 출력 배열입니다.

- `StagingTable` – UTF-8 문자열입니다.

  업서트와 함께 MERGE 또는 APPEND를 수행할 때 사용되는 임시 스테이징 테이블의 이름입니다.

- `SelectedColumns` – 옵션 객체의 배열입니다.

  업서트와 함께 MERGE 또는 APPEND를 수행할 때 일치하는 레코드를 결정하는 데 사용되는 열 이름 목록입니다.

AmazonRedshiftAdvancedOption 구조

Redshift 클러스터에 연결할 때 선택적 값을 지정합니다.

필드

- `Key` – UTF-8 문자열입니다.

  추가 연결 옵션의 키입니다.

- `Value` – UTF-8 문자열입니다.

  추가 연결 옵션의 값입니다.

옵션 구조

옵션 값을 지정합니다.

필드

- `Value` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  옵션 값을 지정합니다.
- `Label` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  옵션의 레이블을 지정합니다.
- `Description` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  옵션에 대한 설명을 지정합니다.

S3CatalogSource 구조

AWS Glue 데이터 카탈로그의 Amazon S3 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽어야 할 데이터베이스입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽어야 할 데이터베이스 테이블입니다.
- `PartitionPredicate` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  이 조건자를 충족하는 파티션이 삭제됩니다. 이러한 파티션에서 보존 기간 내에 있는 파일은 삭제되지 않습니다. 기본적으로 ""(비움)로 설정합니다.
- `AdditionalOptions` – S3SourceAdditionalOptions 객체입니다.

  추가 연결 옵션을 지정합니다.

S3SourceAdditionalOptions 구조

Amazon S3 데이터 스토어에 대한 추가 연결 옵션을 지정합니다.

필드

- `BoundedSize` - 숫자(정수)입니다.

  처리될 데이터 집합의 대상 크기에 대한 상한을 바이트 단위로 설정합니다.
- `BoundedFiles` - 숫자(정수)입니다.

  처리될 대상 파일 수에 대한 상한을 설정합니다.

S3CsvSource 구조

Amazon S3에 저장된 CSV(쉼표로 구분된 값) 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.
- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  읽을 Amazon S3 경로 목록입니다.
- `CompressionType` – UTF-8 문자열입니다(유효 값: `gzip="GZIP"` | `bzip2="BZIP2"`).

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).
- `Exclusions` – UTF-8 문자열의 배열입니다.

  제외할 Unix 스타일 glob 패턴의 JSON 목록이 포함된 문자열입니다. 예를 들어 "[\"**.pdf\"]"는 모든 PDF 파일을 배제합니다.
- `GroupSize` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  대상 그룹 크기(바이트)입니다. 입력 데이터 크기와 클러스터 크기에 따라 기본값을 계산합니다. 입력 파일이 50,000개 미만일 때는 `"groupFiles"`을 `"inPartition"`으로 설정해야 적용됩니다.
- `GroupFiles` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  입력에 50,000개 이상의 파일이 포함된 경우 기본값으로 파일 그룹화가 설정됩니다. 50,000개 미만인 파일로 그룹화를 설정하려면 이 파라미터를 "inPartition"으로 설정합니다. 파일이 50,000개 이상일 때 그룹화를 비활성화하려면 이 파라미터를 `"none"`으로 설정합니다.
- `Recurse` – 부울입니다.

true로 설정할 경우 지정된 경로의 모든 하위 디렉터리에 있는 파일을 재귀적으로 읽습니다.

- `MaxBand` – None 이하의 숫자(정수)입니다.

  이 옵션은 s3 목록이 일정하게 유지되기 시작할 가능성이 있는 기간(밀리초)을 제어합니다. JobBookmarks를 사용하여 Amazon S3 최종 일관성을 처리할 때 수정 타임스탬프가 마지막 maxBand 밀리초에 속하는 파일은 특별히 추적됩니다. 대부분의 사용자는 이 옵션을 설정할 필요가 없습니다. 기본값은 900,000밀리초 또는 15분입니다.
- `MaxFilesInBand` – None 이하의 숫자(정수)입니다.

  이 옵션은 마지막 maxBand초부터 저장할 최대 파일 수를 지정합니다. 이 수를 초과할 경우 추가 파일은 건너뛰고 다음 작업 실행에서만 처리됩니다.
- `AdditionalOptions` – S3DirectSourceAdditionalOptions 객체입니다.

  추가 연결 옵션을 지정합니다.
- `Separator` – 필수: UTF-8 문자열입니다(유효한 값: `comma="COMMA"` | `ctrla="CTRLA"` | `pipe="PIPE"` | `semicolon="SEMICOLON"` | `tab="TAB"`).

  구분 기호 문자열을 지정합니다. 기본값은 쉼표(",")지만 다른 문자도 지정할 수 있습니다.
- `Escaper` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  이스케이프에 사용할 문자를 지정합니다. 이 옵션은 CSV 파일을 읽을 때만 사용됩니다. 기본값은 none입니다. 활성화된 경우 바로 다음에 나오는 문자가 잘 알려진 이스케이프 세트(`\n`, `\r`, `\t` 및 `\0`)를 제외하고는 있는 그대로 사용됩니다.
- `QuoteChar` – 필수: UTF-8 문자열입니다(유효한 값: `quote="QUOTE"` | `quillemet="QUILLEMET"` | `single_quote="SINGLE_QUOTE"` | `disabled="DISABLED"`).

  인용에 사용할 문자를 지정합니다. 기본 문자는 큰 따옴표(")입니다: `'"'`. 전체 인용을 해제하려면 이 값을 `-1`로 설정합니다.
- `Multiline` – 부울입니다.

  단일 기록이 다양한 라인을 포괄할 수 있는지 여부를 지정하는 부울 값입니다. 필드가 인용된 새로운 라인 문자를 포함할 때 발생합니다. 레코드가 여러 줄에 걸쳐 있는 경우 이 옵션을 True로 설정해야 합니다. 기본값은 `False`이라서 파싱 동안 더 많은 공격적 파일 쪼개기가 가능합니다.
- `WithHeader` – 부울입니다.

  첫 번째 라인을 헤더로 취급할지 여부를 지정하는 부울 값입니다. 기본값은 `False`입니다.
- `WriteHeader` – 부울입니다.

헤더를 작성하여 출력할지 여부를 지정하는 부울 값입니다. 기본값은 `True`입니다.

- `SkipFirst` – 부울입니다.

  첫 번째 데이터 라인을 건너뛸지 여부를 지정하는 부울 값입니다. 기본값은 `False`입니다.

- `OptimizePerformance` – 부울입니다.

  Apache Arrow 기반 열 형식 메모리 포맷과 함께 고급 SIMD CSV 리더를 사용할지 여부를 지정하는 부울 값입니다. AWS Glue 버전 3.0에서만 사용할 수 있습니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  S3 CSV 소스에 대한 데이터 스키마를 지정합니다.

DirectJDBCSource 구조

직접 JDBC 소스 연결을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  JDBC 소스 연결의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  JDBC 소스 연결의 데이터베이스입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  JDBC 소스 연결의 테이블입니다.

- `ConnectionName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  JDBC 소스의 연결 이름입니다.

- `ConnectionType` – 필수: UTF-8 문자열입니다(유효한 값: `sqlserver` | `mysql` | `oracle` | `postgresql` | `redshift`).

  JDBC 소스의 연결 유형입니다.

- `RedshiftTmpDir` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  JDBC Redshift 소스의 임시 디렉터리입니다.

S3DirectSourceAdditionalOptions 구조

Amazon S3 데이터 스토어에 대한 추가 연결 옵션을 지정합니다.

필드

- `BoundedSize` - 숫자(정수)입니다.

  처리될 데이터 집합의 대상 크기에 대한 상한을 바이트 단위로 설정합니다.
- `BoundedFiles` - 숫자(정수)입니다.

  처리될 대상 파일 수에 대한 상한을 설정합니다.
- `EnableSamplePath` – 부울입니다.

  샘플 경로를 사용 설정하는 옵션을 설정합니다.
- `SamplePath` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  사용 설정된 경우 샘플 경로를 지정합니다.

S3JsonSource 구조

Amazon S3에 저장된 JSON 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.
- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  읽을 Amazon S3 경로 목록입니다.
- `CompressionType` – UTF-8 문자열입니다(유효 값: `gzip="GZIP"` | `bzip2="BZIP2"`).

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).
- `Exclusions` – UTF-8 문자열의 배열입니다.

  제외할 Unix 스타일 glob 패턴의 JSON 목록이 포함된 문자열입니다. 예를 들어 "[\"**.pdf\"]"는 모든 PDF 파일을 배제합니다.

- `GroupSize` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  대상 그룹 크기(바이트)입니다. 입력 데이터 크기와 클러스터 크기에 따라 기본값을 계산합니다. 입력 파일이 50,000개 미만일 때는 `"groupFiles"`을 `"inPartition"`으로 설정해야 적용됩니다.

- `GroupFiles` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  입력에 50,000개 이상의 파일이 포함된 경우 기본값으로 파일 그룹화가 설정됩니다. 50,000개 미만인 파일로 그룹화를 설정하려면 이 파라미터를 "inPartition"으로 설정합니다. 파일이 50,000개 이상일 때 그룹화를 비활성화하려면 이 파라미터를 `"none"`으로 설정합니다.

- `Recurse` – 부울입니다.

  true로 설정할 경우 지정된 경로의 모든 하위 디렉터리에 있는 파일을 재귀적으로 읽습니다.

- `MaxBand` – None 이하의 숫자(정수)입니다.

  이 옵션은 s3 목록이 일정하게 유지되기 시작할 가능성이 있는 기간(밀리초)을 제어합니다. JobBookmarks를 사용하여 Amazon S3 최종 일관성을 처리할 때 수정 타임스탬프가 마지막 maxBand 밀리초에 속하는 파일은 특별히 추적됩니다. 대부분의 사용자는 이 옵션을 설정할 필요가 없습니다. 기본값은 900,000밀리초 또는 15분입니다.

- `MaxFilesInBand` – None 이하의 숫자(정수)입니다.

  이 옵션은 마지막 maxBand초부터 저장할 최대 파일 수를 지정합니다. 이 수를 초과할 경우 추가 파일은 건너뛰고 다음 작업 실행에서만 처리됩니다.

- `AdditionalOptions` – S3DirectSourceAdditionalOptions 객체입니다.

  추가 연결 옵션을 지정합니다.

- `JsonPath` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  JSON 데이터를 정의하는 JsonPath 문자열입니다.

- `Multiline` – 부울입니다.

  단일 기록이 다양한 라인을 포괄할 수 있는지 여부를 지정하는 부울 값입니다. 필드가 인용된 새로운 라인 문자를 포함할 때 발생합니다. 레코드가 여러 줄에 걸쳐 있는 경우 이 옵션을 True로 설정해야 합니다. 기본값은 `False`이라서 파싱 동안 더 많은 공격적 파일 쪼개기가 가능합니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  S3 JSON 소스에 대한 데이터 스키마를 지정합니다.

S3ParquetSource 구조

Amazon S3에 저장된 Apache Parquet 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.
- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  읽을 Amazon S3 경로 목록입니다.
- `CompressionType` – UTF-8 문자열입니다(유효한 값: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `brotli="BROTLI"` | `lz4="LZ4"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).
- `Exclusions` – UTF-8 문자열의 배열입니다.

  제외할 Unix 스타일 glob 패턴의 JSON 목록이 포함된 문자열입니다. 예를 들어 "[\"**.pdf\"]"는 모든 PDF 파일을 배제합니다.
- `GroupSize` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  대상 그룹 크기(바이트)입니다. 입력 데이터 크기와 클러스터 크기에 따라 기본값을 계산합니다. 입력 파일이 50,000개 미만일 때는 `"groupFiles"`을 `"inPartition"`으로 설정해야 적용됩니다.
- `GroupFiles` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  입력에 50,000개 이상의 파일이 포함된 경우 기본값으로 파일 그룹화가 설정됩니다. 50,000개 미만인 파일로 그룹화를 설정하려면 이 파라미터를 "inPartition"으로 설정합니다. 파일이 50,000개 이상일 때 그룹화를 비활성화하려면 이 파라미터를 `"none"`으로 설정합니다.
- `Recurse` – 부울입니다.

  true로 설정할 경우 지정된 경로의 모든 하위 디렉터리에 있는 파일을 재귀적으로 읽습니다.
- `MaxBand` – None 이하의 숫자(정수)입니다.

  이 옵션은 s3 목록이 일정하게 유지되기 시작할 가능성이 있는 기간(밀리초)을 제어합니다. JobBookmarks를 사용하여 Amazon S3 최종 일관성을 처리할 때 수정 타임스탬프가 마지막

maxBand 밀리초에 속하는 파일은 특별히 추적됩니다. 대부분의 사용자는 이 옵션을 설정할 필요가 없습니다. 기본값은 900,000밀리초 또는 15분입니다.

- `MaxFilesInBand` – None 이하의 숫자(정수)입니다.

  이 옵션은 마지막 maxBand초부터 저장할 최대 파일 수를 지정합니다. 이 수를 초과할 경우 추가 파일은 건너뛰고 다음 작업 실행에서만 처리됩니다.

- `AdditionalOptions` – S3DirectSourceAdditionalOptions 객체입니다.

  추가 연결 옵션을 지정합니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  S3 Parquet 소스에 대한 데이터 스키마를 지정합니다.

S3DeltaSource 구조

Amazon S3에 저장된 Delta Lake 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Delta Lake 소스의 이름입니다.

- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  읽을 Amazon S3 경로 목록입니다.

- `AdditionalDeltaOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  추가 연결 옵션을 지정합니다.

- `AdditionalOptions` – S3DirectSourceAdditionalOptions 객체입니다.

  커넥터의 추가 옵션을 지정합니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  Delta Lake 소스에 대한 데이터 스키마를 지정합니다.

S3CatalogDeltaSource 구조

AWS Glue 데이터 카탈로그에 등록된 Delta Lake 데이터 소스를 지정합니다. 데이터 소스를 Amazon S3에 저장해야 합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Delta Lake 데이터 소스의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.
- `AdditionalDeltaOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  추가 연결 옵션을 지정합니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  Delta Lake 소스에 대한 데이터 스키마를 지정합니다.

CatalogDeltaSource 구조

AWS Glue 데이터 카탈로그에 등록된 Delta Lake 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Delta Lake 데이터 소스의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.
- `AdditionalDeltaOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  추가 연결 옵션을 지정합니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  Delta Lake 소스에 대한 데이터 스키마를 지정합니다.

S3HudiSource 구조

Amazon S3에 저장된 Hudi 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Hudi 테이블의 이름입니다.
- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  읽을 Amazon S3 경로 목록입니다.
- `AdditionalHudiOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  추가 연결 옵션을 지정합니다.
- `AdditionalOptions` – S3DirectSourceAdditionalOptions 객체입니다.

  커넥터의 추가 옵션을 지정합니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  Hudi 소스에 대한 데이터 스키마를 지정합니다.

S3CatalogHudiSource 구조

AWS Glue 데이터 카탈로그에 등록된 Hudi 데이터 소스를 지정합니다. Hudi 데이터 소스를 Amazon S3에 저장해야 합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Hudi 데이터 소스의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.
- `AdditionalHudiOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  추가 연결 옵션을 지정합니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  Hudi 소스에 대한 데이터 스키마를 지정합니다.

S3ExcelSource 구조

S3 Excel 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  S3 Excel 데이터 소스의 이름입니다.
- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  Excel 파일이 있는 S3 경로입니다.

- `CompressionType` – UTF-8 문자열입니다(유효한 값: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `brotli="BROTLI"` | `lz4="LZ4"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

  Excel 파일에 사용되는 압축 형식입니다.

- `Exclusions` – UTF-8 문자열의 배열입니다.

  처리에서 특정 파일 또는 경로를 제외하는 패턴입니다.

- `GroupSize` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  배치 처리에 사용할 파일 그룹의 크기를 정의합니다.

- `GroupFiles` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  처리할 때 파일을 그룹화하는 방법을 지정합니다.

- `Recurse` – 부울입니다.

  하위 디렉터리를 재귀적으로 처리할지 여부를 나타냅니다.

- `MaxBand` – None 이하의 숫자(정수)입니다.

  사용할 최대 처리 밴드 수입니다.

- `MaxFilesInBand` – None 이하의 숫자(정수)입니다.

  각 밴드에서 처리할 최대 파일 수입니다.

- `AdditionalOptions` – S3DirectSourceAdditionalOptions 객체입니다.

  S3 직접 소스 처리를 위한 추가 구성 옵션입니다.

- `NumberRows` - 숫자(정수)입니다.

  각 Excel 파일에서 처리할 행 수입니다.

- `SkipFooter` – None 이하의 숫자(정수)입니다.

  각 Excel 파일의 끝에서 건너뛸 행 수입니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  처리된 데이터에 적용할 AWS Glue 스키마입니다.

CatalogHudiSource 구조

AWS Glue 데이터 카탈로그에 등록된 Hudi 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Hudi 데이터 소스의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.
- `AdditionalHudiOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  추가 연결 옵션을 지정합니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  Hudi 소스에 대한 데이터 스키마를 지정합니다.

DynamoDBCatalogSource 구조

AWS Glue 데이터 카탈로그의 DynamoDB 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

읽을 데이터베이스 테이블의 이름입니다.

RelationalCatalogSource 구조

AWS Glue 데이터 카탈로그의 관계형 데이터베이스 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 원본의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽을 데이터베이스 테이블의 이름입니다.

JDBCConnectorTarget 구조

Apache Parquet 열 형식 스토리지의 Amazon S3에 쓰는 데이터 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.
- `ConnectionName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터와 연관된 연결 이름입니다.
- `ConnectionTable` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상에 있는 테이블의 이름입니다.
- `ConnectorName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

사용할 커넥터의 이름입니다.

- `ConnectionType` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  JDBC 데이터 대상에 대한 연결을 지정하는 marketplace.jdbc 또는 custom.jdbc와 같은 연결 유형입니다.

- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션입니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  JDBC 대상의 데이터 스키마를 지정합니다.

SparkConnectorTarget 구조

Apache Spark 커넥터를 사용하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `ConnectionName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Apache Spark 커넥터에 대한 연결 이름입니다.

- `ConnectorName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Apache Spark 커넥터의 이름입니다.

- `ConnectionType` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Apache Spark 데이터 스토어에 대한 연결을 지정하는 marketplace.spark 또는 custom.spark와 같은 연결 유형입니다.

- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션입니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  사용자 지정 Spark 대상에 대한 데이터 스키마를 지정합니다.

BasicCatalogTarget 구조

AWS Glue 데이터 카탈로그 테이블을 사용하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.
- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  특정 키 또는 키 세트를 기반으로 여러 파티션 또는 샤드에 데이터를 분산하는 데 사용되는 파티션 키.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  대상으로 사용할 테이블이 포함된 데이터베이스입니다. 이 데이터베이스가 데이터 카탈로그에 이미 존재해야 합니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  출력 데이터의 스키마를 정의하는 테이블입니다. 이 테이블이 데이터 카탈로그에 이미 존재해야 합니다.

MySQLCatalogTarget 구조

MySQL을 사용하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.

PostgreSQLCatalogTarget 구조

Postgres SQL을 사용하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.

OracleSQLCatalogTarget 구조

Oracle SQL을 사용하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.

MicrosoftSQLServerCatalogTarget 구조

Microsoft SQL을 사용하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.

RedshiftTarget 구조

Amazon Redshift를 사용하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.
- `RedshiftTmpDir` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터베이스 외부에서 복사할 때 임시 데이터를 스테이징할 수 있는 Amazon S3 경로입니다.
- `TmpDirIAMRole` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  권한이 있는 IAM 역할입니다.
- `UpsertRedshiftOptions` – UpsertRedshiftTargetOptions 객체입니다.

  Redshift 대상에 쓸 때 업서트 작업을 구성하는 옵션 세트입니다.

AmazonRedshiftTarget 구조

Amazon Redshift 대상을 지정합니다.

필드

- `Name` – Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Amazon Redshift 대상의 이름입니다.
- `Data` – AmazonRedshiftNodeData 객체입니다.

  Amazon Redshift 대상 노드의 데이터를 지정합니다.
- `Inputs` – UTF-8 문자열의 배열입니다(1개의 문자열).

  데이터 대상에 대한 입력인 노드입니다.

UpsertRedshiftTargetOptions 구조

Redshift 대상에 쓸 때 업서트 작업을 구성하는 옵션입니다.

필드

- `TableLocation` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Redshift 테이블의 물리적 위치입니다.
- `ConnectionName` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Redshift에 쓰는 데 사용할 연결 이름입니다.
- `UpsertKeys` – UTF-8 문자열의 배열입니다.

  업데이트 또는 삽입 수행 여부를 결정하는 데 사용되는 키입니다.

S3CatalogTarget 구조

AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에 쓰는 데이터 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.
- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.
- `SchemaChangePolicy` – CatalogSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

S3GlueParquetTarget 구조

Apache Parquet 열 형식 스토리지의 Amazon S3에 쓰는 데이터 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.
- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.
- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 단일 Amazon S3 경로입니다.
- `Compression` – UTF-8 문자열입니다(유효한 값: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `brotli="BROTLI"` | `lz4="LZ4"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).
- `NumberTargetPartitions` – UTF-8 문자열입니다.

  AWS Glue를 사용하여 Amazon S3에 쓸 때 Parquet 파일의 대상 파티션 수를 지정합니다.
- `SchemaChangePolicy` – DirectSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

CatalogSchemaChangePolicy 구조

크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

필드

- `EnableUpdateCatalog` – 부울입니다.

크롤러가 변경된 스키마를 찾았을 때 지정된 업데이트 동작을 사용할지 여부입니다.

- `UpdateBehavior` – UTF-8 문자열입니다(유효 값: `UPDATE_IN_DATABASE` | `LOG`).

  크롤러가 변화된 객체를 찾을 때 업데이트 동작.

S3DirectTarget 구조

Amazon S3에 쓰는 데이터 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.

- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 단일 Amazon S3 경로입니다.

- `Compression` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).

- `NumberTargetPartitions` – UTF-8 문자열입니다.

  데이터를 Amazon S3에 직접 쓸 때 대상 파티션 수를 지정합니다.

- `Format` – 필수: UTF-8 문자열(유효한 값: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"` | `iceberg="ICEBERG"` | `hyper="HYPER"` | `xml="XML"`).

  대상에 대한 데이터 출력 포맷을 지정합니다.

- `SchemaChangePolicy` – DirectSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

S3HudiCatalogTarget 구조

AWS Glue 데이터 카탈로그의 Hudi 데이터 소스에 작성하는 데이터 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.
- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.
- `AdditionalOptions` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션을 지정합니다.
- `SchemaChangePolicy` – CatalogSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

S3HudiDirectTarget 구조

Amazon S3에서 Hudi 데이터 소스에 작성하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 Hudi 데이터 소스의 Amazon S3 경로입니다.

- `Compression` – 필수: UTF-8 문자열입니다(유효한 값: `gzip="GZIP"` | `lzo="LZO"` | `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).

- `NumberTargetPartitions` – UTF-8 문자열입니다.

  Amazon S3 전체에 Hudi 데이터세트 파일을 배포할 대상 파티션 수를 지정합니다.

- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.

- `Format` – 필수: UTF-8 문자열(유효한 값: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"` | `iceberg="ICEBERG"` | `hyper="HYPER"` | `xml="XML"`).

  대상에 대한 데이터 출력 포맷을 지정합니다.

- `AdditionalOptions` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션을 지정합니다.

- `SchemaChangePolicy` – DirectSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

S3DeltaCatalogTarget 구조

AWS Glue 데이터 카탈로그의 Delta Lake 데이터 소스에 작성하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.
- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.
- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션을 지정합니다.
- `SchemaChangePolicy` – CatalogSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

S3DeltaDirectTarget 구조

Amazon S3에서 Delta Lake 데이터 소스에 작성하는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.

- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 Delta Lake 데이터 소스의 Amazon S3 경로입니다.

- `Compression` – 필수: UTF-8 문자열입니다(유효한 값: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  데이터 압축 방식을 지정합니다. 이 작업은 데이터에 표준 파일 확장자가 있는 경우에는 필요하지 않습니다. 가능한 값은 `"gzip"` 및 `"bzip"`입니다).

- `NumberTargetPartitions` – UTF-8 문자열입니다.

  Amazon S3 전체에 Delta Lake 데이터세트 파일을 배포할 대상 파티션 수를 지정합니다.

- `Format` – 필수: UTF-8 문자열(유효한 값: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"` | `iceberg="ICEBERG"` | `hyper="HYPER"` | `xml="XML"`).

  대상에 대한 데이터 출력 포맷을 지정합니다.

- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  커넥터에 대한 추가 연결 옵션을 지정합니다.

- `SchemaChangePolicy` – DirectSchemaChangePolicy 객체입니다.

  크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

S3HyperDirectTarget 구조

Amazon S3에 쓰는 HyperDirect 데이터 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  HyperDirect 대상 노드의 고유 식별자입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  HyperDirect 대상의 입력 소스를 지정합니다.
- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  출력 데이터에 대한 파티셔닝 전략을 정의합니다.
- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  출력 데이터를 기록할 S3 위치입니다.
- `Compression` – UTF-8 문자열입니다(유효한 값: `uncompressed="UNCOMPRESSED"`).

  출력 데이터에 적용할 압축 유형입니다.
- `SchemaChangePolicy` – DirectSchemaChangePolicy 객체입니다.

  쓰기 작업 중에 스키마 변경 사항을 처리하는 방법을 정의합니다.

S3IcebergDirectTarget 구조

Amazon S3에서 Iceberg 데이터 소스에 쓰는 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 파이프라인의 Iceberg 대상 노드에 대한 고유 식별자를 지정합니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  이 Iceberg 대상에 데이터를 제공하는 단일 입력 소스를 정의합니다.
- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  S3에서 Iceberg 테이블 데이터를 파티셔닝하는 데 사용되는 열을 지정합니다.
- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Iceberg 테이블 데이터가 저장될 S3 위치를 정의합니다.
- `Format` – 필수: UTF-8 문자열(유효한 값: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"` | `iceberg="ICEBERG"` | `hyper="HYPER"` | `xml="XML"`).

  Iceberg 테이블 데이터 저장에 사용되는 파일 형식(예: Parquet, ORC)을 지정합니다.

- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Iceberg 테이블 동작을 사용자 지정하기 위한 추가 구성 옵션을 제공합니다.
- `SchemaChangePolicy` – DirectSchemaChangePolicy 객체입니다.

  Iceberg 테이블에 데이터를 쓸 때 스키마 변경 사항을 처리하는 방법을 정의합니다.
- `Compression` – 필수: UTF-8 문자열입니다(유효한 값: `gzip="GZIP"` | `lzo="LZO"` | `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  S3의 Iceberg 테이블 파일에 사용되는 압축 코덱을 지정합니다.
- `NumberTargetPartitions` – UTF-8 문자열입니다.

  S3에 Iceberg 테이블 파일을 배포하기 위한 대상 파티션 수를 설정합니다.

DirectSchemaChangePolicy 구조

크롤러에 대한 업데이트 동작을 지정하는 정책입니다.

필드

- `EnableUpdateCatalog` – 부울입니다.

  크롤러가 변경된 스키마를 찾았을 때 지정된 업데이트 동작을 사용할지 여부입니다.
- `UpdateBehavior` – UTF-8 문자열입니다(유효 값: `UPDATE_IN_DATABASE` | `LOG`).

  크롤러가 변화된 객체를 찾을 때 업데이트 동작.
- `Table` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  스키마 변경 정책이 적용되는 데이터베이스의 테이블을 지정합니다.
- `Database` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  스키마 변경 정책이 적용되는 데이터베이스를 지정합니다.

ApplyMapping 구조

데이터 원본의 데이터 속성 키를 데이터 대상의 데이터 속성 키에 매핑하는 변환을 지정합니다. 키의 이름을 바꾸고 키의 데이터 유형을 수정하고 데이터 집합에서 삭제할 키를 선택할 수 있습니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.
- `Mapping` – 필수: 매핑 객체의 배열입니다.

  데이터 원본의 데이터 속성 키를 데이터 대상의 데이터 속성 키에 매핑하도록 지정합니다.

Mapping 구조

데이터 속성 키의 매핑을 지정합니다.

필드

- `ToKey` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  매핑을 적용한 후의 열 이름입니다. `FromPath`와 같을 수 있습니다.
- `FromPath` – UTF-8 문자열의 배열입니다.

  수정할 테이블 또는 열입니다.
- `FromType` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  수정할 데이터 유형입니다.
- `ToType` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  데이터가 수정되는 데이터 유형입니다.
- `Dropped` – 부울입니다.

  true인 경우 열이 제거됩니다.
- `Children` – 매핑 객체의 배열입니다.

중첩된 데이터 구조에만 적용됩니다. 상위 구조뿐만 아니라 하위 구조 중 하나도 변경하려는 경우 이 데이터 구조를 작성할 수 있습니다. 마찬가지로 Mapping이지만 해당 FromPath도 상위 구조의 FromPath와 이 구조의 FromPath가 됩니다.

하위 부분의 경우 다음과 같은 구조가 있다고 가정합니다.

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

다음과 같은 Mapping을 지정할 수 있습니다.

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

SelectFields 구조

유지할 데이터 속성 키를 선택하는 변환을 지정합니다.

필드

- Name – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- Inputs – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.
- Paths – 필수: UTF-8 문자열의 배열입니다.

  데이터 구조의 변수에 대한 JSON 경로입니다.

DropFields 구조

삭제할 데이터 속성 키를 선택하는 변환을 지정합니다.

필드

- Name – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

변환 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.

- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  데이터 구조의 변수에 대한 JSON 경로입니다.

RenameField 구조

단일 데이터 속성 키의 이름을 바꾸는 변환을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.

- `SourcePath` – 필수: UTF-8 문자열의 배열입니다.

  소스 데이터에 대한 데이터 구조의 변수에 대한 JSON 경로입니다.

- `TargetPath` – 필수: UTF-8 문자열의 배열입니다.

  대상 데이터에 대한 데이터 구조의 변수에 대한 JSON 경로입니다.

Spigot 구조

Amazon S3 버킷에 데이터 샘플을 쓰는 변환을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

노드 이름으로 식별된 데이터 입력입니다.

- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  변환이 데이터 집합의 레코드 하위 집합을 Amazon S3 버킷의 JSON 파일에 쓰는 Amazon S3의 경로입니다.

- `Topk` – 100 이하의 숫자(정수)입니다.

  데이터 집합의 시작 부분부터 쓸 레코드 수를 지정합니다.

- `Prob` – 1 이하의 숫자(double)입니다.

  지정된 레코드를 선택할 확률(최대값이 1인 소수 값)입니다. 값 1은 데이터 집합에서 읽은 각 행이 샘플 출력에 포함되어야 함을 나타냅니다.

조인 구조

지정된 데이터 속성 키의 비교 구문을 사용하여 두 데이터 집합을 하나의 데이터 집합으로 조인하는 변환을 지정합니다. 내부, 외부, 왼쪽, 오른쪽, 왼쪽 반 및 왼쪽 안티 조인을 사용할 수 있습니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(2개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.

- `JoinType` – 필수: UTF-8 문자열입니다(유효한 값: `equijoin="EQUIJOIN"` | `left="LEFT"` | `right="RIGHT"` | `outer="OUTER"` | `leftsemi="LEFT_SEMI"` | `leftanti="LEFT_ANTI"`).

  데이터 집합에서 수행할 조인 유형을 지정합니다.

- `Columns` – 필수(Required): 2개 이상의 구조로 이루어진 JoinColumn 객체의 배열입니다.

  조인할 두 열의 목록입니다.

JoinColumn 구조

조인할 열을 지정합니다.

필드

- `From` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  조인할 열입니다.

- `Keys` – 필수: UTF-8 문자열의 배열입니다.

  조인할 열의 키입니다.

SplitFields 구조

데이터 속성 키를 두 개의 `DynamicFrames`로 분할하는 변환을 지정합니다. 출력은 `DynamicFrames` 컬렉션입니다. 하나에는 선택한 데이터 속성 키가 있고 다른 하나에는 나머지 데이터 속성 키가 있습니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.

- `Paths` – 필수: UTF-8 문자열의 배열입니다.

  데이터 구조의 변수에 대한 JSON 경로입니다.

SelectFromCollection 구조

`DynamicFrames` 컬렉션에서 하나의 `DynamicFrame`을 선택하는 변환을 지정합니다. 출력은 선택한 `DynamicFrame`입니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

노드 이름으로 식별된 데이터 입력입니다.

- `Index` – 필수(Required): None 이하의 숫자(정수)입니다.

  선택할 DynamicFrame의 인덱스입니다.

FillMissingValues 구조

데이터 집합에서 누락된 값이 있는 레코드를 찾고 대체를 통해 결정된 값으로 새 필드를 추가하는 변환을 지정합니다. 입력 데이터 집합은 누락 값을 결정하는 기계 학습 모델을 훈련하는 데 사용됩니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.
- `ImputedPath` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  대체된 데이터 집합에 대한 데이터 구조의 변수에 대한 JSON 경로입니다.
- `FilledPath` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  작성된 데이터 집합에 대한 데이터 구조의 변수에 대한 JSON 경로입니다.

Filter 구조

필터 조건에 따라 하나의 데이터 집합을 두 개로 분할하는 변환을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.
- `LogicalOperator` – 필수: UTF-8 문자열입니다(유효한 값: `AND` | `OR`).

키 값을 지정된 값과 비교하여 행을 필터링하는 데 사용되는 연산자입니다.

- `Filters` – 필수: FilterExpression 객체의 배열입니다.

  필터 표현식을 지정합니다.

FilterExpression 구조

필터 표현식을 지정합니다.

필드

- `Operation` – 필수: UTF-8 문자열입니다(유효한 값: `EQ` | `LT` | `GT` | `LTE` | `GTE` | `REGEX` | `ISNULL`).

  표현식에서 수행할 작업의 유형입니다.
- `Negated` – 부울입니다.

  표현식을 부정할지 여부입니다.
- `Values` – 필수: FilterValue 객체의 배열입니다.

  필터 값 목록입니다.

FilterValue 구조

`FilterExpression`의 값 목록에 있는 단일 항목을 나타냅니다.

필드

- `Type` – 필수: UTF-8 문자열입니다(유효한 값: `COLUMNEXTRACTED` | `CONSTANT`).

  필터 값 유형입니다.
- `Value` – 필수: UTF-8 문자열의 배열입니다.

  연결할 값입니다.

CustomCode 구조

제공한 사용자 지정 코드를 사용하여 데이터 변환을 수행하는 변환을 지정합니다. 출력은 DynamicFrames의 컬렉션입니다.

필드

- Name – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- Inputs – 필수: UTF-8 문자열의 배열이며 문자열은 1개 이상입니다.

  노드 이름으로 식별된 데이터 입력입니다.

- Code – 필수: Custom string pattern #52과(와) 일치하는 UTF-8 문자열입니다.

  데이터 변환을 수행하는 데 사용되는 사용자 지정 코드입니다.

- ClassName – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  사용자 지정 코드 노드 클래스에 대해 정의된 이름입니다.

- OutputSchemas – GlueSchema 객체의 배열입니다.

  사용자 지정 코드 변환에 대한 데이터 스키마를 지정합니다.

SparkSQL 구조

데이터를 변환하기 위해 Spark SQL 구문을 사용하여 SQL 쿼리를 입력하는 변환을 지정합니다. 출력은 단일 DynamicFrame입니다.

필드

- Name – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- Inputs – 필수: UTF-8 문자열의 배열이며 문자열은 1개 이상입니다.

  노드 이름으로 식별된 데이터 입력입니다. SQL 쿼리에 사용할 각 입력 노드와 테이블 이름을 연결할 수 있습니다. 선택한 이름은 Spark SQL 이름 지정 제한을 충족해야 합니다.

- SqlQuery – 필수: Custom string pattern #60과(와) 일치하는 UTF-8 문자열입니다.

  Spark SQL 구문을 사용하고 단일 데이터 집합을 반환해야 하는 SQL 쿼리입니다.

- SqlAliases – 필수: SqlAlias 객체의 배열입니다.

별칭 목록입니다. 별칭을 사용하면 지정된 입력에 대해 SQL에서 사용할 이름을 지정할 수 있습니다. 예를 들어 'MyDataSource'라는 데이터 원본이 있습니다. `From`을 MyDataSource로, `Alias`를 SqlName으로 지정할 경우 SQL에서 다음을 수행할 수 있습니다.

```
select * from SqlName
```

그러면 MyDataSource에서 데이터를 가져옵니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  SparkSQL 변환에 대한 데이터 스키마를 지정합니다.

SqlAlias 구조

`SqlAliases`의 값 목록에 있는 단일 항목을 나타냅니다.

필드

- `From` – 필수: Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  테이블 또는 테이블의 열입니다.
- `Alias` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  테이블 또는 테이블의 열에 지정된 임시 이름입니다.

DropNullFields 구조

열의 모든 값이 'null'인 경우 데이터 집합에서 열을 제거하는 변환을 지정합니다. 기본값으로 AWS Glue Studio는 Null 객체를 인식하지만 빈 문자열, 'null'인 문자열, -1 정수 또는 0과 같은 다른 자리 표시자 등의 일부 값은 자동으로 Null로 인식되지 않습니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.

- `NullCheckBoxList` – NullCheckBoxList 객체입니다.

  특정 값을 제거하기 위해 Null 값으로 인식할지 여부를 나타내는 구조입니다.

- `NullTextList` – NullValueField 객체의 배열이며 구조는 50개 이하입니다.

  0이나 데이터 집합에 고유한 Null 자리 표시자로 사용되는 다른 값과 같은 사용자 지정 Null 값을 나타내는 NullValueField 구조 목록을 지정하는 구조입니다.

  `DropNullFields` 변환은 Null 자리 표시자의 값과 데이터 유형이 모두 데이터와 일치하는 경우에만 사용자 지정 Null 값을 제거합니다.

NullCheckBoxList 구조

제거를 위해 특정 값을 Null 값으로 인식할지 여부를 나타냅니다.

필드

- `IsEmpty` – 부울입니다.

  빈 문자열이 Null 값으로 간주되도록 지정합니다.

- `IsNullString` – 부울입니다.

  'null'이라는 단어의 철자를 사용하는 값이 Null 값으로 간주되도록 지정합니다.

- `IsNegOne` – 부울입니다.

  정수 값 -1이 Null 값으로 간주되도록 지정합니다.

NullValueField 구조

0이나 데이터 집합에 고유한 Null 자리 표시자로 사용되는 다른 값과 같은 사용자 지정 Null 값을 나타냅니다.

필드

- `Value` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  Null 자리 표시자의 값입니다.

- `Datatype` – 필수(Required): 데이터 형식 객체입니다.

값의 데이터 유형입니다.

데이터 형식 구조

값의 데이터 유형을 나타내는 구조입니다.

필드

- `Id` – 필수: Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  값의 데이터 유형입니다.
- `Label` – 필수: Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  데이터 유형에 할당된 레이블입니다.

병합 구조

레코드를 식별하기 위해 지정된 기본 키를 기준으로 `DynamicFrame`을 스테이징 `DynamicFrame`과 병합하는 변환을 지정합니다. 중복 레코드(기본 키가 동일한 레코드)는 중복 제거되지 않습니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(2개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.
- `Source` – 필수: Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  스테이징 `DynamicFrame`과 병합되는 소스 `DynamicFrame`입니다.
- `PrimaryKeys` – 필수: UTF-8 문자열의 배열입니다.

  소스 및 스테이징 동적 프레임의 레코드와 일치시킬 기본 키 필드 목록입니다.

결합 구조

둘 이상 데이터 집합의 행을 단일 결과로 결합하는 변환을 지정합니다.

필드

- Name – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- Inputs – 필수(Required): UTF-8 문자열의 배열(2개 이상)입니다.

  노드 ID가 변환에 입력됩니다.
- UnionType – 필수: UTF-8 문자열입니다(유효한 값: ALL | DISTINCT).

  Union 변환 유형을 나타냅니다.

  데이터 원본의 모든 행을 결과 DynamicFrame에 조인하려면 ALL을 지정합니다. 결과 union 구조는 중복 행을 제거하지 않습니다.

  결과 DynamicFrame에서 중복 행을 제거하려면 DISTINCT를 지정합니다.

PIIDetection 구조

PII 데이터를 식별, 제거 또는 마스킹하는 변환을 지정합니다.

필드

- Name – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- Inputs – 필수: UTF-8 문자열의 배열(1개 이상)입니다.

  노드 ID가 변환에 입력됩니다.
- PiiType – 필수: UTF-8 문자열입니다(유효한 값: RowAudit | RowMasking | ColumnAudit | ColumnMasking).

  PIIDetection 변환 유형을 나타냅니다.
- EntityTypesToDetect – 필수: UTF-8 문자열의 배열입니다.

  PIIDetection 변환이 PII 데이터로 식별할 엔터티 유형을 나타냅니다.

  PII 유형 엔터티는 다음을 포함합니다. PERSON_NAME, DATE, USA_SNN, EMAIL, USA_ITIN, USA_PASSPORT_NUMBER, PHONE_NUMBER, BANK_ACCOUNT, IP_ADDRESS, MAC_ADDRESS, USA_CPT_CODE, USA_HCPCS_CODE,

USA_NATIONAL_DRUG_CODE, USA_MEDICARE_BENEFICIARY_IDENTIFIER, USA_HEALTH_INSURANCE_CLAIM_NUMBER,CREDIT_CARD,USA_NATIONAL_PROVIDER_IDENTIFI

- `OutputColumnName` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  해당 행에서 감지된 모든 엔터티 유형을 포함할 출력 열 이름을 나타냅니다.
- `SampleFraction` – 1 이하의 숫자(실수)입니다.

  PII 엔터티를 스캔할 때 샘플링할 데이터의 비율을 나타냅니다.
- `ThresholdFraction` – 1 이하의 숫자(실수)입니다.

  열을 PII 데이터로 식별하기 위해 충족되어야 하는 데이터의 비율을 나타냅니다.
- `MaskValue` – Custom string pattern #56과(와) 일치하는 256바이트 이하 길이의 UTF-8 문자열입니다.

  감지된 개체를 대체할 값을 나타냅니다.

집계 구조

선택한 필드별로 행을 그룹화하고 지정된 함수에 의해 집계된 값을 계산하는 변환을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.
- `Inputs` – 필수: UTF-8 문자열의 배열(1개 이상)입니다.

  집계 변환에 대해 입력으로 사용할 필드와 행을 지정합니다.
- `Groups` – 필수: UTF-8 문자열의 배열입니다.

  그룹화할 필드를 지정합니다.
- `Aggs` – 필수(Required): AggregateOperation 객체의 배열이며 구조는 1~30개입니다.

  지정된 필드에서 수행할 집계 함수를 지정합니다.

DropDuplicates 구조

데이터세트에서 반복 데이터의 행을 제거하는 변환을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  변환 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  노드 이름으로 식별된 데이터 입력입니다.

- `Columns` – UTF-8 문자열의 배열입니다.

  반복될 경우 병합하거나 제거할 열의 이름입니다.

GovernedCatalogTarget 구조

AWS Glue 데이터 카탈로그를 사용하여 Amazon S3에 쓰는 데이터 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 대상의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 대상에 대한 입력인 노드입니다.

- `PartitionKeys` – UTF-8 문자열의 배열입니다.

  일련의 키를 사용하여 기본 분할을 지정합니다.

- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스 테이블의 이름입니다.

- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  쓰기를 수행할 데이터베이스의 이름입니다.

- `SchemaChangePolicy` – CatalogSchemaChangePolicy 객체입니다.

  관리 카탈로그에 대한 업데이트 동작을 지정하는 정책입니다.

GovernedCatalogSource 구조

관리 AWS Glue 데이터 카탈로그의 데이터 스토어를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 스토어의 이름입니다.
- `Database` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽어야 할 데이터베이스입니다.
- `Table` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  읽어야 할 데이터베이스 테이블입니다.
- `PartitionPredicate` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  이 조건자를 충족하는 파티션이 삭제됩니다. 이러한 파티션에서 보존 기간 내에 있는 파일은 삭제되지 않습니다. 기본적으로 ""(비움)로 설정합니다.
- `AdditionalOptions` – S3SourceAdditionalOptions 객체입니다.

  추가 연결 옵션을 지정합니다.

AggregateOperation 구조

집계 변환에서 집계를 수행하는 데 필요한 파라미터 세트를 지정합니다.

필드

- `Column` – 필수: UTF-8 문자열의 배열입니다.

  집계 함수가 적용될 데이터 세트의 열을 지정합니다.
- `AggFunc` – 필수(Required): UTF-8 문자열입니다(유효 값: `avg` | `countDistinct` | `count` | `first` | `last` | `kurtosis` | `max` | `min` | `skewness` | `stddev_samp` | `stddev_pop` | `sum` | `sumDistinct` | `var_samp` | `var_pop`).

  적용할 집계 함수를 지정합니다.

  가능한 집계 함수로는 평균 개수별, 개수, 첫 번째, 마지막, kurtosis, 최대, 최소, 왜도, stddev_samp, stddev_pop, 합계, sumDistinct, var_samp, var_pop 등이 있습니다.

GlueSchema 구조

스키마를 AWS Glue에서 결정할 수 없는 경우 사용자 정의 스키마를 지정합니다.

필드

- `Columns` – GlueStudioSchemaColumn 객체의 배열입니다.

  AWS Glue 스키마를 구성하는 열 정의를 지정합니다.

GlueStudioSchemaColumn 구조

AWS Glue 스키마 정의에서 단일 열을 지정합니다.

필드

- `Name` – 필수: Single-line string pattern과 일치하는 1,024바이트 이하 길이의 UTF-8 문자열입니다.

  AWS Glue Studio 스키마의 열 이름입니다.
- `Type` – Single-line string pattern과(와) 일치하는 131,072바이트 이하 길이의 UTF-8 문자열입니다.

  AWS Glue Studio 스키마의 이 열에 대한 하이브 유형입니다.

GlueStudioColumn 구조

AWS Glue Studio에서 단일 열을 지정합니다.

필드

- `Key` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  AWS Glue Studio에서 열의 키입니다.
- `FullPath` – 필수: UTF-8 문자열의 배열입니다.

  AWS Glue Studio에서 열의 전체 URL입니다.
- `Type` – 필수: UTF-8 문자열(유효한 값: `array="ARRAY"` | `bigint="BIGINT"` | `bigint array="BIGINT_ARRAY"` | `binary="BINARY"` | `binary array="BINARY_ARRAY"` | `boolean="BOOLEAN"` | `boolean array="BOOLEAN_ARRAY"` | `byte="BYTE"` | `byte array="BYTE_ARRAY"` | `char="CHAR"` | `char array="CHAR_ARRAY"` | `choice="CHOICE"`

| choice array="CHOICE_ARRAY" | date="DATE" | date array="DATE_ARRAY" | decimal="DECIMAL" | decimal array="DECIMAL_ARRAY" | double="DOUBLE" | double array="DOUBLE_ARRAY" | enum="ENUM" | enum array="ENUM_ARRAY" | float="FLOAT" | float array="FLOAT_ARRAY" | int="INT" | int array="INT_ARRAY" | interval="INTERVAL" | interval array="INTERVAL_ARRAY" | long="LONG" | long array="LONG_ARRAY" | object="OBJECT" | short="SHORT" | short array="SHORT_ARRAY" | smallint="SMALLINT" | smallint array="SMALLINT_ARRAY" | string="STRING" | string array="STRING_ARRAY" | timestamp="TIMESTAMP" | timestamp array="TIMESTAMP_ARRAY" | tinyint="TINYINT" | tinyint array="TINYINT_ARRAY" | varchar="VARCHAR" | varchar array="VARCHAR_ARRAY" | null="NULL" | unknown="UNKNOWN" | unknown array="UNKNOWN_ARRAY").

  AWS Glue Studio에서 열의 유형입니다.

- `Children` - 구조의 배열입니다.

  AWS Glue Studio에서 상위 열의 하위 요소입니다.

DynamicTransform 구조

동적 변환을 수행하는 데 필요한 파라미터 세트를 지정합니다.

필드

- Name – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  동적 변환의 이름을 지정합니다.

- `TransformName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  AWS Glue Studio 시각적 편집기에 표시되는 동적 변환의 이름을 지정합니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  필요한 동적 변환에 대한 입력을 지정합니다.

- `Parameters` – TransformConfigParameter 객체의 배열입니다.

  동적 변환의 파라미터를 지정합니다.

- `FunctionName` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  동적 변환의 함수 이름을 지정합니다.

- `Path` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  동적 변환 소스 및 구성 파일의 경로를 지정합니다.
- `Version` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  이 필드는 사용되지 않으며 향후 릴리스에서 사용 중단됩니다.
- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  동적 변환에 대한 데이터 스키마를 지정합니다.

TransformConfigParameter 구조

동적 변환 구성 파일의 파라미터를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  동적 변환 구성 파일의 파라미터 이름을 지정합니다.
- `Type` – 필수: UTF-8 문자열입니다(유효한 값: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  동적 변환 구성 파일의 파라미터 유형을 지정합니다.
- `ValidationRule` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  동적 변환 구성 파일의 검증 규칙을 지정합니다.
- `ValidationMessage` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  동적 변환 구성 파일의 검증 메시지를 지정합니다.
- `Value` – UTF-8 문자열의 배열입니다.

  동적 변환 구성 파일의 파라미터 값을 지정합니다.
- `ListType` – UTF-8 문자열입니다(유효한 값: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  동적 변환 구성 파일의 파라미터 목록 유형을 지정합니다.
- `IsOptional` – 부울입니다.

  파라미터가 동적 변환 구성 파일에서 선택 사항인지 여부를 지정합니다.

EvaluateDataQuality 구조

데이터 품질 평가 기준을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 품질 평가의 이름입니다.
- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  데이터 품질 평가의 입력입니다.
- `Ruleset` – 필수: Custom string pattern #57과(와) 일치하는 1~65536바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 평가를 위한 규칙 세트입니다.
- `Output` – UTF-8 문자열입니다(유효 값: `PrimaryInput` | `EvaluationResults`).

  데이터 품질 평가의 출력입니다.
- `PublishingOptions` – DQResultsPublishingOptions 객체입니다.

  결과 게시 방법을 구성하는 옵션입니다.
- `StopJobOnFailureOptions` – DQStopJobOnFailureOptions 객체입니다.

  데이터 품질 평가에 실패할 경우 작업을 중지하는 방법을 구성하는 옵션입니다.

DQResultsPublishingOptions 구조

데이터 품질 평가 결과 게시 방법을 구성하는 옵션입니다.

필드

- `EvaluationContext` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  평가의 컨텍스트입니다.
- `ResultsS3Prefix` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  결과 앞에 Amazon S3 접두사가 추가되었습니다.
- `CloudWatchMetricsEnabled` – 부울입니다.

데이터 품질 결과에 대한 지표를 활성화합니다.

- `ResultsPublishingEnabled` – 부울입니다.

  데이터 품질 결과에 대한 게시를 활성화합니다.

DQStopJobOnFailureOptions 구조

데이터 품질 평가에 실패할 경우 작업을 중지하는 방법을 구성하는 옵션입니다.

필드

- `StopJobOnFailureTiming` – UTF-8 문자열입니다(유효 값: `Immediate` | `AfterDataLoad`).

  데이터 품질 평가에 실패할 경우에 작업을 중지할 시점입니다. 옵션은 Immediate 또는 AfterDataLoad입니다.

EvaluateDataQualityMultiFrame 구조

데이터 품질 평가 기준을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  데이터 품질 평가의 이름입니다.

- `Inputs` – 필수: UTF-8 문자열의 배열이며 문자열은 1개 이상입니다.

  데이터 품질 평가의 입력입니다. 이 목록의 첫 번째 입력은 기본 데이터 소스입니다.

- `AdditionalDataSources` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  기본을 제외한 모든 데이터 소스의 별칭입니다.

- `Ruleset` – 필수: Custom string pattern #57과(와) 일치하는 1~65536바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 평가를 위한 규칙 세트입니다.

- `PublishingOptions` – DQResultsPublishingOptions 객체입니다.

  결과 게시 방법을 구성하는 옵션입니다.

- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열(유효 값: `performanceTuning.caching="CacheOption" | observations.scope="ObservationsOption"`)입니다.

  각 값은 UTF-8 문자열입니다.

  변환의 런타임 동작을 구성하는 옵션입니다.

- `StopJobOnFailureOptions` – DQStopJobOnFailureOptions 객체입니다.

  데이터 품질 평가에 실패할 경우 작업을 중지하는 방법을 구성하는 옵션입니다.

레시피 구조

AWS Glue 작업에서 AWS Glue DataBrew 레시피를 사용하는 AWS Glue 스튜디오 노드입니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  AWS Glue 스튜디오 노드의 이름입니다.

- `Inputs` – 필수(Required): UTF-8 문자열의 배열(1개 이상)입니다.

  레시피 노드에 대한 입력에 해당하는 노드로, ID로 식별됩니다.

- `RecipeReference` – RecipeReference 객체입니다.

  노드에서 사용하는 DataBrew 레시피에 대한 참조입니다.

- `RecipeSteps` – RecipeStep 객체의 배열입니다.

  레시피 노드에서 사용되는 변환 단계.

RecipeReference 구조

레시피에 대한 AWS Glue DataBrew 참조입니다.

필드

- `RecipeArn` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  DataBrew 레시피의 ARN입니다.
- `RecipeVersion` – 필수: 1~16바이트 길이의 UTF-8 문자열입니다.

  DataBrew 레시피의 RecipeVersion입니다.

SnowflakeNodeData 구조

AWS Glue Studio에서 Snowflake 노드에 대한 구성을 지정합니다.

필드

- `SourceType` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  검색된 데이터를 지정하는 방법을 지정합니다. 유효한 값: `"table"`, `"query"`.
- `Connection` – 옵션 객체입니다.

  Snowflake 엔드포인트에 대한 AWS Glue 데이터 카탈로그 연결을 지정합니다.
- `Schema` – UTF-8 문자열입니다.

  노드에서 사용할 Snowflake 데이터베이스 스키마를 지정합니다.
- `Table` – UTF-8 문자열입니다.

  노드에서 사용할 Snowflake 테이블을 지정합니다.
- `Database` – UTF-8 문자열입니다.

  노드에서 사용할 Snowflake 데이터베이스를 지정합니다.
- `TempDir` – Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  현재 사용 중이지 않습니다.
- `IamRole` – 옵션 객체입니다.

  현재 사용 중이지 않습니다.
- `AdditionalOptions` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

각 값은 Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

Snowflake 커넥터에 전달되는 추가 옵션을 지정합니다. 이 노드의 다른 위치에서 옵션이 지정된 경우 이 옵션이 우선합니다.

- `SampleQuery` – UTF-8 문자열입니다.

  `query` 소스 유형의 데이터를 검색하는 데 사용되는 SQL 문자열입니다.

- `PreAction` – UTF-8 문자열입니다.

  Snowflake 커넥터가 표준 작업을 수행하기 전에 실행되는 SQL 문자열입니다.

- `PostAction` – UTF-8 문자열입니다.

  Snowflake 커넥터가 표준 작업을 수행한 후에 실행되는 SQL 문자열입니다.

- `Action` – UTF-8 문자열입니다.

  기존 데이터가 있는 테이블에 쓸 때 수행할 작업을 지정합니다. 유효한 값: `append`, `merge`, `truncate`, `drop`.

- `Upsert` – 부울입니다.

  `append` 작업일 때 사용됩니다. 행이 이미 있는 경우 확인 동작을 지정합니다. true인 경우 기존 행이 업데이트됩니다. false인 경우 해당 행이 삽입됩니다.

- `MergeAction` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  병합 작업을 지정합니다. 유효한 값: `simple`, `custom`. simple인 경우 병합 동작은 `MergeWhenMatched` 및 `MergeWhenNotMatched`로 정의됩니다. custom인 경우 `MergeClause`로 정의됩니다.

- `MergeWhenMatched` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  병합 시 기존 데이터와 일치하는 레코드를 확인하는 방법을 지정합니다. 유효한 값: `update`, `delete`.

- `MergeWhenNotMatched` – Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  병합 시 기존 데이터와 일치하지 않는 레코드를 처리하는 방법을 지정합니다. 유효한 값: `insert`, `none`.

- `MergeClause` – UTF-8 문자열입니다.

  사용자 지정 병합 동작을 지정하는 SQL 문입니다.

- `StagingTable` – UTF-8 문자열입니다.

  `merge` 또는 업서트 `append` 작업을 수행할 때 사용되는 스테이징 테이블의 이름입니다. 데이터가 이 테이블에 기록된 후에는 생성된 사후 작업에 의해 `table`로 이동됩니다.

- `SelectedColumns` – 옵션 객체의 배열입니다.

  병합 및 업서트에서 일치하는 항목을 감지할 때 레코드를 식별하기 위해 결합된 열을 지정합니다. `value`, `label` 및 `description` 키가 있는 구조 목록입니다. 각 구조는 열을 설명합니다.

- `AutoPushdown` – 부울입니다.

  자동 쿼리 푸시다운의 활성화 여부를 지정합니다. 푸시다운이 활성화된 경우 Spark에서 쿼리를 실행할 때 쿼리의 일부를 Snowflake 서버로 '푸시다운'할 수 있으면 해당 쿼리가 푸시다운됩니다. 이렇게 하면 일부 쿼리의 성능이 향상됩니다.

- `TableSchema` – 옵션 객체의 배열입니다.

  노드의 대상 스키마를 수동으로 정의합니다. `value`, `label` 및 `description` 키가 있는 구조 목록입니다. 각 구조는 열을 정의합니다.

SnowflakeSource 구조

Snowflake 데이터 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Snowflake 데이터 소스의 이름입니다.

- `Data` – 필수(Required): SnowflakeNodeData 객체입니다.

  Snowflake 데이터 소스의 구성입니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  출력 데이터에 대한 사용자 정의 스키마를 지정합니다.

SnowflakeTarget 구조

Snowflake 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  Snowflake 대상의 이름입니다.

- `Data` – 필수(Required): SnowflakeNodeData 객체입니다.

  Snowflake 대상 노드의 데이터를 지정합니다.

- `Inputs` – UTF-8 문자열의 배열입니다(1개의 문자열).

  데이터 대상에 대한 입력인 노드입니다.

ConnectorDataSource 구조

표준 연결 옵션으로 생성된 소스를 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  이 소스 노드의 이름입니다.

- `ConnectionType` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  `connectionType`은 기본 AWS Glue 라이브러리에 제공된 값입니다. 노드 유형은 다음과 같은 연결 유형을 지원합니다.

  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`

- `Data` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

노드에 대한 연결 옵션을 지정하는 맵입니다. AWS Glue 설명서의 연결 매개변수 섹션에서 해당 연결 유형에 대한 표준 연결 옵션을 찾을 수 있습니다.

- `OutputSchemas` – GlueSchema 객체의 배열입니다.

  이 소스에 대한 데이터 스키마를 지정합니다.

ConnectorDataTarget 구조

표준 연결 옵션으로 생성된 대상을 지정합니다.

필드

- `Name` – 필수: Custom string pattern #61과(와) 일치하는 UTF-8 문자열입니다.

  이 대상 노드의 이름입니다.

- `ConnectionType` – 필수: Custom string pattern #59과(와) 일치하는 UTF-8 문자열입니다.

  `connectionType`은 기본 AWS Glue 라이브러리에 제공된 값입니다. 노드 유형은 다음과 같은 연결 유형을 지원합니다.

  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`

- `Data` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  노드에 대한 연결 옵션을 지정하는 맵입니다. AWS Glue 설명서의 연결 매개변수 섹션에서 해당 연결 유형에 대한 표준 연결 옵션을 찾을 수 있습니다.

- `Inputs` – UTF-8 문자열의 배열입니다(1개의 문자열).

데이터 대상에 대한 입력인 노드입니다.

RecipeStep 구조

AWS Glue Studio 데이터 준비 레시피 노드에서 사용되는 레시피 단계.

필드

- `Action` – 필수(Required): RecipeAction 객체입니다.

  레시피 단계의 변환 작업.

- `ConditionExpressions` – ConditionExpression 객체의 배열입니다.

  레시피 단계에 대한 조건 표현식.

RecipeAction 구조

AWS Glue Studio 데이터 준비 레시피 노드에 정의된 작업.

필드

- `Operation` – 필수(Required): Custom string pattern #54과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  레시피 작업.

- `Parameters` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #55과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  각 값은 길이가 1~32,768바이트인 UTF-8 문자열입니다.

  레시피 작업의 파라미터.

ConditionExpression 구조

AWS Glue Studio 데이터 준비 레시피 노드에 정의된 조건 표현식.

필드

- `Condition` – 필수(Required): Custom string pattern #54과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  조건 표현식의 조건.

- `Value` – UTF-8 문자열(1,024바이트 이하).

  조건 표현식의 값.

- `TargetColumn` – 필수: 1~1,024바이트 길이의 UTF-8 문자열입니다.

  조건 표현식의 대상 열.

작업 API

작업 API는 작업 데이터 유형을 설명하며 AWS Glue에서 작업, 작업 실행, 트리거로 작업하기 위한 API를 포함합니다.

주제

- 작업
- 작업 실행
- 트리거

작업

작업 API는 AWS Glue에서의 작업 생성, 업데이트, 삭제 또는 확인과 관련된 API 및 데이터 유형에 대해 설명합니다.

데이터 타입

- 작업 구조
- ExecutionProperty 구조
- NotificationProperty 구조
- JobCommand 구조
- ConnectionsList 구조

- JobUpdate 구조
- SourceControlDetails 구조

작업 구조

작업 정의를 지정합니다.

필드

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 정의에 할당하는 이름입니다.
- JobMode – UTF-8 문자열입니다(유효한 값: SCRIPT="" | VISUAL="" | NOTEBOOK="").

  작업이 생성된 방법을 설명하는 모드입니다. 유효한 값은 다음과 같습니다.
  - SCRIPT - AWS Glue Studio 스크립트 편집기를 사용하여 작업을 생성했습니다.
  - VISUAL - AWS Glue Studio 시각적 편집기를 사용하여 작업을 생성했습니다.
  - NOTEBOOK - 대화형 세션 노트북을 사용하여 작업을 생성했습니다.

  JobMode 필드가 없거나 null인 경우 기본값으로 SCRIPT가 할당됩니다.
- JobRunQueuingEnabled – 부울입니다.

  이 작업에 대한 작업 실행을 위해 작업 실행 대기열을 활성화할지 여부를 지정합니다.

  값이 true이면 해당 작업 실행에 대해 작업 실행 큐가 활성화됩니다. false이거나 채워지지 않은 경우 작업 실행은 대기열에 포함되는 것으로 간주되지 않습니다.

  이 필드가 작업 실행에 설정된 값과 일치하지 않으면 작업 실행 필드의 값이 사용됩니다.
- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  작업 설명입니다.
- LogUri – UTF-8 문자열입니다.

  이 필드는 향후 사용하기 위해 예약되어 있습니다.
- Role – UTF-8 문자열입니다.

  이 작업과 연결된 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다.

- `CreatedOn` – 타임스탬프입니다.

  작업 정의가 생성된 날짜와 시간입니다.

- `LastModifiedOn` – 타임스탬프입니다.

  이 작업 정의가 수정된 마지막 시점입니다.

- `ExecutionProperty` – ExecutionProperty 객체입니다.

  `ExecutionProperty`는 이 작업에 허용된 최대 동시 실행 수를 지정합니다.

- `Command` – JobCommand 객체입니다.

  이 작업을 실행하는 `JobCommand`입니다.

- `DefaultArguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  이 작업의 모든 실행에서 기본 인수(이름 값 페어로 지정됨)입니다.

  AWS Glue 자체가 사용하는 인수는 물론 사용자의 작업 실행 스크립트가 사용하는 인수를 지정할 수 있습니다.

  작업 인수가 로깅될 수 있습니다. 일반 텍스트 보안 암호를 인수로 전달하지 마세요. 보안 암호를 작업 내에 보관하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

  자체 작업 인수를 지정하고 사용하는 방법에 대한 자세한 내용은 개발자 가이드의 Python에서 AWS Glue Glue API 호출을 참조하세요.

  Spark 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 AWS Glue에서 사용하는 특별 파라미터 주제를 참조하세요.

  Ray 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 Ray 작업에서 작업 파라미터 사용을 참조하세요.

- `NonOverridableArguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

작업 실행 시 작업 인수를 제공할 때 재정의되지 않는 이 작업의 인수(이름 값 페어로 지정됨)입니다.

- `Connections` – ConnectionsList 객체입니다.

  이 작업에 사용된 연결입니다.

- `MaxRetries` - 숫자(정수)입니다.

  JobRun이 실패한 후 이 작업을 다시 시도할 수 있는 최대 횟수입니다.

- `AllocatedCapacity` - 숫자(정수)입니다.

  이 필드는 더 이상 사용되지 않습니다. 대신 `MaxCapacity`을 사용하세요.

  이 작업 실행에 따라 할당된 AWS Glue 데이터 처리 장치(DPU) 수입니다. 최소 2DPU를 할당할 수 있습니다. 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  작업 타임아웃(분)입니다. 작업을 실행하여 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다.

  작업의 시간 제한 값은 7일 또는 10,080분 미만이어야 합니다. 그렇지 않으면 작업에서 예외가 발생합니다.

  값을 비워 두면 제한 시간은 기본적으로 2,880분으로 설정됩니다.

  제한 시간 값이 7일을 초과하는 기존 AWS Glue 작업은 기본적으로 7일로 설정됩니다. 예를 들어 배치 작업에 20일의 제한 시간을 지정한 경우 7일째 되는 날에 작업이 중지됩니다.

  스트리밍 작업은 유지 관리 기간을 설정한 경우 7일 후 유지 관리 기간 중에 작업이 다시 시작됩니다.

- `MaxCapacity` - 숫자(double)입니다.

  Glue 버전 1.0 이전 적업의 경우 표준 작업자 유형을 사용하여 이 작업을 실행할 때 할당할 수 있는 AWS Glue 데이터 처리 장치(DPU) 수입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  Glue 버전 2.0 이상 작업의 경우 `Maximum capacity`를 지정할 수 없습니다. 그 대신 `Worker type` 및 `Number of workers`를 지정해야 합니다.

  `WorkerType` 및 `NumberOfWorkers`를 사용하는 경우, `MaxCapacity`를 설정하지 마십시오.

MaxCapacity에 할당할 수 있는 값은 Python 셸 작업을 실행하는지 또는 Apache Spark ETL 작업 또는 Apache Spark 스트리밍 ETL 작업을 실행하는지에 따라 다릅니다.

- Python 셸 작업(JobCommand.Name="pythonshell")을 지정하면 0.0625 또는 1 DPU를 할당할 수 있습니다. 기본값은 0.0625 DPU입니다.
- Apache Spark ETL 작업(JobCommand.Name="glueetl") 또는 Apache Spark 스트리밍 ETL 작업(JobCommand.Name="gluestreaming") 을 지정하면 2에서 100 DPU를 할당할 수 있습니다. 기본값은 10 DPU입니다. 이 작업 유형에는 부분적인 DPU 할당을 사용할 수 없습니다.

- WorkerType – UTF-8 문자열입니다(유효한 값: Standard="" | G.1X="" | G.2X="" | G.025X="" | G.4X="" | G.8X="" | Z.2X="").

  작업이 실행될 때 할당되는 미리 정의된 작업자 유형입니다. Spark 작업에 대해 G.1X, G.2X, G.4X, G.8X 또는 G.025X의 값을 허용합니다. Ray 작업에 대해 Z.2X 값을 허용합니다.

  - G.1X 작업자 유형의 경우, 각 작업자가 94GB의 디스크가 있는 1DPU(4개의 vCPU, 16GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
  - G.2X 작업자 유형의 경우, 각 작업자가 138GB의 디스크가 있는 2DPU(8개의 vCPU, 32GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
  - G.4X 작업자 유형의 경우, 각 작업자가 256GB의 디스크가 있는 4DPU(16개의 vCPU, 64GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 다음 AWS 리전에서 실행되는 Spark ETL 작업의 AWS Glue 버전 3.0 이상에서만 사용할 수 있습니다. 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오리건), 캐나다(중부), 아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 유럽(스페인), 유럽(스톡홀름), 남아메리카(상파울루).
  - G.8X 작업자 유형의 경우, 각 작업자가 512GB의 디스크가 있는 8DPU(32개의 vCPU, 128GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 G.4X 작업자 유형에 지원되는 동일한 AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.

  - `G.025X` 작업자 유형의 경우, 각 작업자가 84GB의 디스크가 있는 0.25DPU(2개의 vCPU, 4GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 볼륨이 낮은 스트리밍 작업에 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 AWS Glue 버전 3.0 이상 스트리밍 작업에만 사용할 수 있습니다.
  - `Z.2X` 작업자 유형의 경우, 각 작업자는 128GB 디스크에서 2개의 M-DPU(vCPU 8개, 메모리 64GB)에 매핑되고, Autoscaler에 따라 최대 8개의 Ray 작업자를 제공합니다.
- `NumberOfWorkers` - 숫자(정수)입니다.

  작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.
- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업에 사용할 `SecurityConfiguration` 구조의 이름입니다.
- `NotificationProperty` – NotificationProperty 객체입니다.

  작업 알림의 구성 속성을 지정합니다.
- `Running` – 부울입니다.

  이 필드는 향후 사용하기 위해 예약되어 있습니다.
- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Spark 작업에서 `GlueVersion`에 따라 AWS Glue에서 작업에 사용할 수 있는 Apache Spark 및 Python 버전이 결정됩니다. Python의 버전으로 Spark 유형의 작업에 대해 지원되는 버전을 확인할 수 있습니다.

  Ray 작업에서는 `GlueVersion`을 `4.0` 이상으로 설정해야 합니다. 그러나 Ray 작업에서 사용할 수 있는 Ray, Python 및 추가 라이브러리의 버전은 Job 명령의 `Runtime` 파라미터에 의해 결정됩니다.

  이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전에 대한 자세한 내용은 개발자 안내서의 Glue 버전을 참조하세요.

  Glue 버전 지정 없이 생성된 작업은 Glue 0.9로 기본 지정됩니다.
- `CodeGenConfigurationNodes` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

  각 값은 CodeGenConfigurationNode 객체입니다.

Glue Studio 시각적 구성 요소 및 Glue Studio 코드 생성의 기반이 되는 방향성 비순환 그래프의 표현입니다.

- `ExecutionClass` - 16바이트 미만의 UTF-8 문자열입니다(유효한 값: `FLEX="" | STANDARD=""`).

  작업이 표준 또는 유연한 실행 클래스로 실행되는지 여부를 나타냅니다. 표준 실행 클래스는 빠른 작업 시작 및 전용 리소스가 필요한 시간에 민감한 워크로드에 적합합니다.

  유연한 실행 클래스는 시작 및 완료 시간이 다를 수 있는 시간에 민감하지 않은 작업에 적합합니다.

  AWS Glue 버전 3.0 이상 및 명령 유형 `glueetl`을 사용하는 작업만 `ExecutionClass`가 `FLEX`로 설정됩니다. 유연한 실행 클래스는 Spark 작업에 사용할 수 있습니다.

- `SourceControlDetails` – SourceControlDetails 객체입니다.

  작업에 대한 소스 제어 구성에 대한 세부 정보로, 원격 리포지토리와의 작업 아티팩트 동기화를 허용합니다.

- `MaintenanceWindow` – Custom string pattern #34과(와) 일치하는 UTF-8 문자열입니다.

  이 필드는 스트리밍 작업에 대한 유지 관리 기간의 요일과 시간을 지정합니다. AWS Glue에서는 정기적으로 유지 관리 작업을 수행합니다. 이러한 유지 관리 기간 동안에는 AWS Glue에서 스트리밍 작업을 다시 시작해야 합니다.

  AWS Glue에서는 지정된 유지 관리 기간으로부터 3시간 이내에 작업을 다시 시작합니다. 예를 들어, 유지 관리 기간을 GMT 기준 월요일 오전 10시로 설정하면 작업이 GMT 기준 오전 10시에서 오후 1시 사이에 다시 시작됩니다.

- `ProfileName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업과 연결된 AWS Glue 사용 프로필의 이름입니다.

ExecutionProperty 구조

작업의 실행 속성입니다.

필드

- `MaxConcurrentRuns` - 숫자(정수)입니다.

작업에 허용된 최대 동시 실행 수입니다. 기본 값은 1입니다. 이 임곗값에 도달하면 오류가 반환됩니다. 지정할 수 있는 최대값은 서비스 제한에 따라 통제됩니다.

NotificationProperty 구조

알림의 구성 속성을 지정합니다.

필드

- `NotifyDelayAfter` – 1 이상의 숫자(정수)입니다.

  작업 실행 시작 후 작업 실행 대기 알림을 전송하기 전까지 대기하는 시간(분)입니다.

JobCommand 구조

작업이 실행될 때 실행되는 코드를 지정합니다.

필드

- `Name` – UTF-8 문자열입니다.

  작업 명령의 이름입니다. Apache Spark ETL 작업의 경우, `glueetl`이어야 합니다. Python 셸 작업의 경우, `pythonshell`이어야 합니다. Apache Spark 스트리밍 ETL 작업의 경우, `gluestreaming`이어야 합니다. Ray 작업의 경우 `glueray`이어야 합니다.
- `ScriptLocation` – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  작업을 실행하는 스크립트의 Amazon Simple Storage Service(Amazon S3) 경로를 지정합니다.
- `PythonVersion` – Custom string pattern #48과(와) 일치하는 UTF-8 문자열입니다.

  Python 셸 작업을 실행하는 데 사용되는 Python 버전입니다. 허용되는 값은 2 또는 3입니다.
- `Runtime` – Custom string pattern #33과 일치하는 64바이트 이하 길이의 UTF-8 문자열입니다.

  Ray 작업에서 런타임은 사용자 환경에서 사용 가능한 Ray, Python 및 추가 라이브러리의 버전을 지정하는 데 사용됩니다. 이 필드는 다른 작업 유형에서 사용되지 않습니다. 지원되는 런타임 환경 값은 AWS Glue 개발자 안내서에서 지원되는 Ray 런타임 환경을 참조하세요.

ConnectionsList 구조

작업이 사용한 연결을 지정합니다.

필드

- `Connections` – 1,000개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  작업이 사용한 연결 목록입니다.

JobUpdate 구조

기존 작업 정의 업데이트에 사용된 정보를 지정합니다. 이전 작업 정의를 이 정보로 완전히 덮어씁니다.

필드

- `JobMode` – UTF-8 문자열입니다(유효한 값: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

  작업이 생성된 방법을 설명하는 모드입니다. 유효한 값은 다음과 같습니다.
  - `SCRIPT` - AWS Glue Studio 스크립트 편집기를 사용하여 작업을 생성했습니다.
  - `VISUAL` - AWS Glue Studio 시각적 편집기를 사용하여 작업을 생성했습니다.
  - `NOTEBOOK` - 대화형 세션 노트북을 사용하여 작업을 생성했습니다.

  `JobMode` 필드가 없거나 null인 경우 기본값으로 `SCRIPT`가 할당됩니다.
- `JobRunQueuingEnabled` – 부울입니다.

  이 작업에 대한 작업 실행을 위해 작업 실행 대기열을 활성화할지 여부를 지정합니다.

  값이 true이면 해당 작업 실행에 대해 작업 실행 큐가 활성화됩니다. false이거나 채워지지 않은 경우 작업 실행은 대기열에 포함되는 것으로 간주되지 않습니다.

  이 필드가 작업 실행에 설정된 값과 일치하지 않으면 작업 실행 필드의 값이 사용됩니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  정의된 작업에 대한 설명입니다.
- `LogUri` – UTF-8 문자열입니다.

  이 필드는 향후 사용하기 위해 예약되어 있습니다.

- `Role` – UTF-8 문자열입니다.

  이 작업과 연결된 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다(필수).

- `ExecutionProperty` – ExecutionProperty 객체입니다.

  `ExecutionProperty`는 이 작업에 허용된 최대 동시 실행 수를 지정합니다.

- `Command` – JobCommand 객체입니다.

  이 작업을 실행하는 `JobCommand`입니다(필수).

- `DefaultArguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  이 작업의 모든 실행에서 기본 인수(이름 값 페어로 지정됨)입니다.

  AWS Glue 자체가 사용하는 인수는 물론 사용자의 작업 실행 스크립트가 사용하는 인수를 지정할 수 있습니다.

  작업 인수가 로깅될 수 있습니다. 일반 텍스트 보안 암호를 인수로 전달하지 마세요. 보안 암호를 작업 내에 보관하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

  자체 작업 인수를 지정하고 사용하는 방법에 대한 자세한 내용은 개발자 가이드의 Python에서 AWS Glue Glue API 호출을 참조하세요.

  Spark 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 AWS Glue에서 사용하는 특별 파라미터 주제를 참조하세요.

  Ray 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 Ray 작업에서 작업 파라미터 사용을 참조하세요.

- `NonOverridableArguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  작업 실행 시 작업 인수를 제공할 때 재정의되지 않는 이 작업의 인수(이름 값 페어로 지정됨)입니다.

- `Connections` – ConnectionsList 객체입니다.

이 작업에 사용된 연결입니다.

- `MaxRetries` - 숫자(정수)입니다.

  실패한 경우 이 작업을 다시 시도할 수 있는 최대 횟수입니다.

- `AllocatedCapacity` - 숫자(정수)입니다.

  이 필드는 더 이상 사용되지 않습니다. 대신 `MaxCapacity`을 사용하세요.

  이 작업에 할당할 AWS Glue 데이터 처리 장치(DPU) 수입니다. 최소 2DPU를 할당할 수 있습니다. 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  작업 타임아웃(분)입니다. 작업을 실행하여 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다.

  작업의 시간 제한 값은 7일 또는 10,080분 미만이어야 합니다. 그렇지 않으면 작업에서 예외가 발생합니다.

  값을 비워 두면 제한 시간은 기본적으로 2,880분으로 설정됩니다.

  제한 시간 값이 7일을 초과하는 기존 AWS Glue 작업은 기본적으로 7일로 설정됩니다. 예를 들어 배치 작업에 20일의 제한 시간을 지정한 경우 7일째 되는 날에 작업이 중지됩니다.

  스트리밍 작업은 유지 관리 기간을 설정한 경우 7일 후 유지 관리 기간 중에 작업이 다시 시작됩니다.

- `MaxCapacity` - 숫자(double)입니다.

  Glue 버전 1.0 이전 적업의 경우 표준 작업자 유형을 사용하여 이 작업을 실행할 때 할당할 수 있는 AWS Glue 데이터 처리 장치(DPU) 수입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  Glue 버전 2.0 이상 작업의 경우 `Maximum capacity`를 지정할 수 없습니다. 그 대신 `Worker type` 및 `Number of workers`를 지정해야 합니다.

  `WorkerType` 및 `NumberOfWorkers`를 사용하는 경우, `MaxCapacity`를 설정하지 마십시오.

  `MaxCapacity`에 할당할 수 있는 값은 Python 셸 작업을 실행하는지 또는 Apache Spark ETL 작업 또는 Apache Spark 스트리밍 ETL 작업을 실행하는지에 따라 다릅니다.

  - Python 셸 작업(`JobCommand.Name="pythonshell"`)을 지정하면 0.0625 또는 1 DPU를 할당할 수 있습니다. 기본값은 0.0625 DPU입니다.
  - Apache Spark ETL 작업(`JobCommand.Name="glueetl"`) 또는 Apache Spark 스트리밍 ETL 작업(`JobCommand.Name="gluestreaming"`) 을 지정하면 2에서 100 DPU를 할당할 수 있습니다. 기본값은 10 DPU입니다. 이 작업 유형에는 부분적인 DPU 할당을 사용할 수 없습니다.
- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  작업이 실행될 때 할당되는 미리 정의된 작업자 유형입니다. Spark 작업에 대해 G.1X, G.2X, G.4X, G.8X 또는 G.025X의 값을 허용합니다. Ray 작업에 대해 Z.2X 값을 허용합니다. 자세한 내용은 Spark 작업을 위한 작업 속성 정의를 참조하세요.
- `NumberOfWorkers` - 숫자(정수)입니다.

  작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.
- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업에 사용할 `SecurityConfiguration` 구조의 이름입니다.
- `NotificationProperty` – NotificationProperty 객체입니다.

  작업 알림의 구성 속성을 지정합니다.
- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Spark 작업에서 `GlueVersion`에 따라 AWS Glue에서 작업에 사용할 수 있는 Apache Spark 및 Python 버전이 결정됩니다. Python의 버전으로 Spark 유형의 작업에 대해 지원되는 버전을 확인할 수 있습니다.

  Ray 작업에서는 `GlueVersion`을 `4.0` 이상으로 설정해야 합니다. 그러나 Ray 작업에서 사용할 수 있는 Ray, Python 및 추가 라이브러리의 버전은 Job 명령의 `Runtime` 파라미터에 의해 결정됩니다.

  이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전에 대한 자세한 내용은 개발자 안내서의 Glue 버전을 참조하세요.

  Glue 버전 지정 없이 생성된 작업은 Glue 0.9로 기본 지정됩니다.
- `CodeGenConfigurationNodes` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

각 값은 CodeGenConfigurationNode 객체입니다.

Glue Studio 시각적 구성 요소 및 Glue Studio 코드 생성의 기반이 되는 방향성 비순환 그래프의 표현입니다.

- `ExecutionClass` - 16바이트 미만의 UTF-8 문자열입니다(유효한 값: `FLEX="" | STANDARD=""`).

  작업이 표준 또는 유연한 실행 클래스로 실행되는지 여부를 나타냅니다. 표준 실행 클래스는 빠른 작업 시작 및 전용 리소스가 필요한 시간에 민감한 워크로드에 적합합니다.

  유연한 실행 클래스는 시작 및 완료 시간이 다를 수 있는 시간에 민감하지 않은 작업에 적합합니다.

  AWS Glue 버전 3.0 이상 및 명령 유형 `glueetl`을 사용하는 작업만 `ExecutionClass`가 `FLEX`로 설정됩니다. 유연한 실행 클래스는 Spark 작업에 사용할 수 있습니다.

- `SourceControlDetails` – SourceControlDetails 객체입니다.

  작업에 대한 소스 제어 구성에 대한 세부 정보로, 원격 리포지토리와의 작업 아티팩트 동기화를 허용합니다.

- `MaintenanceWindow` – Custom string pattern #34과(와) 일치하는 UTF-8 문자열입니다.

  이 필드는 스트리밍 작업에 대한 유지 관리 기간의 요일과 시간을 지정합니다. AWS Glue에서는 정기적으로 유지 관리 작업을 수행합니다. 이러한 유지 관리 기간 동안에는 AWS Glue에서 스트리밍 작업을 다시 시작해야 합니다.

  AWS Glue에서는 지정된 유지 관리 기간으로부터 3시간 이내에 작업을 다시 시작합니다. 예를 들어, 유지 관리 기간을 GMT 기준 월요일 오전 10시로 설정하면 작업이 GMT 기준 오전 10시에서 오후 1시 사이에 다시 시작됩니다.

- `ProfileName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업과 연결된 AWS Glue 사용 프로필의 이름입니다.

SourceControlDetails 구조

작업에 대한 소스 제어 구성에 대한 세부 정보로, 원격 리포지토리와의 작업 아티팩트 동기화를 허용합니다.

필드

- `Provider` – UTF-8 문자열입니다(유효 값: `GITHUB` | `AWS_CODE_COMMIT`).

  원격 리포지토리의 공급자입니다.
- `Repository` – 1~512바이트 길이의 UTF-8 문자열입니다.

  작업 아티팩트가 포함된 원격 리포지토리의 이름입니다.
- `Owner` – 1~512바이트 길이의 UTF-8 문자열입니다.

  작업 아티팩트가 포함된 원격 리포지토리의 소유자입니다.
- `Branch` – 1~512바이트 길이의 UTF-8 문자열입니다.

  원격 리포지토리의 선택적 브랜치입니다.
- `Folder` – 1~512바이트 길이의 UTF-8 문자열입니다.

  원격 리포지토리의 선택적 폴더입니다.
- `LastCommitId` – 1~512바이트 길이의 UTF-8 문자열입니다.

  원격 리포지토리의 커밋에 대한 마지막 커밋 ID입니다.
- `LastSyncTimestamp` – 1~512바이트 길이의 UTF-8 문자열입니다.

  마지막 작업 동기화가 수행된 날짜와 시간입니다.
- `AuthStrategy` – UTF-8 문자열입니다(유효 값: `PERSONAL_ACCESS_TOKEN` | `AWS_SECRETS_MANAGER`).

  인증 유형으로, AWS Secrets Manager에 저장된 인증 토큰이거나 개인 액세스 토큰일 수 있습니다.
- `AuthToken` – 1~512바이트 길이의 UTF-8 문자열입니다.

  인증 토큰의 값입니다.

운영

- CreateJob 작업(Python: create_job)
- UpdateJob 작업(Python: update_job)
- GetJob 작업(Python: get_job)
- GetJobs 작업(Python: get_jobs)

- DeleteJob 작업(Python: delete_job)
- ListJobs 작업(Python: list_jobs)
- BatchGetJobs 작업(Python: batch_get_jobs)

CreateJob 작업(Python: create_job)

새로운 작업 정의를 만듭니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 정의에 할당하는 이름입니다. 계정에서 고유해야 합니다.
- JobMode – UTF-8 문자열입니다(유효한 값: SCRIPT="" | VISUAL="" | NOTEBOOK="").

  작업이 생성된 방법을 설명하는 모드입니다. 유효한 값은 다음과 같습니다.
  - SCRIPT - AWS Glue Studio 스크립트 편집기를 사용하여 작업을 생성했습니다.
  - VISUAL - AWS Glue Studio 시각적 편집기를 사용하여 작업을 생성했습니다.
  - NOTEBOOK - 대화형 세션 노트북을 사용하여 작업을 생성했습니다.

  JobMode 필드가 없거나 null인 경우 기본값으로 SCRIPT가 할당됩니다.
- JobRunQueuingEnabled – 부울입니다.

  이 작업에 대한 작업 실행을 위해 작업 실행 대기열을 활성화할지 여부를 지정합니다.

  값이 true이면 해당 작업 실행에 대해 작업 실행 큐가 활성화됩니다. false이거나 채워지지 않은 경우 작업 실행은 대기열에 포함되는 것으로 간주되지 않습니다.

  이 필드가 작업 실행에 설정된 값과 일치하지 않으면 작업 실행 필드의 값이 사용됩니다.
- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  정의된 작업에 대한 설명입니다.
- LogUri – UTF-8 문자열입니다.

  이 필드는 향후 사용하기 위해 예약되어 있습니다.
- Role – 필수(Required): UTF-8 문자열입니다.

이 작업과 연결된 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다.

- `ExecutionProperty` – ExecutionProperty 객체입니다.

  `ExecutionProperty`는 이 작업에 허용된 최대 동시 실행 수를 지정합니다.

- `Command` – 필수(Required): JobCommand 객체입니다.

  이 작업을 실행하는 `JobCommand`입니다.

- `DefaultArguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  이 작업의 모든 실행에서 기본 인수(이름 값 페어로 지정됨)입니다.

  AWS Glue 자체가 사용하는 인수는 물론 사용자의 작업 실행 스크립트가 사용하는 인수를 지정할 수 있습니다.

  작업 인수가 로깅될 수 있습니다. 일반 텍스트 보안 암호를 인수로 전달하지 마세요. 보안 암호를 작업 내에 보관하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

  자체 작업 인수를 지정하고 사용하는 방법에 대한 자세한 내용은 개발자 가이드의 Python에서 AWS Glue Glue API 호출을 참조하세요.

  Spark 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 AWS Glue에서 사용하는 특별 파라미터 주제를 참조하세요.

  Ray 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 Ray 작업에서 작업 파라미터 사용을 참조하세요.

- `NonOverridableArguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  작업 실행 시 작업 인수를 제공할 때 재정의되지 않는 이 작업의 인수(이름 값 페어로 지정됨)입니다.

- `Connections` – ConnectionsList 객체입니다.

이 작업에 사용된 연결입니다.

- `MaxRetries` - 숫자(정수)입니다.

  실패한 경우 이 작업을 다시 시도할 수 있는 최대 횟수입니다.

- `AllocatedCapacity` - 숫자(정수)입니다.

  이 파라미터는 이제 사용되지 않습니다. 대신 `MaxCapacity`을 사용하세요.

  이 작업에 할당할 AWS Glue 데이터 처리 장치(DPU) 수입니다. 최소 2DPU를 할당할 수 있습니다. 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  작업 타임아웃(분)입니다. 작업을 실행하여 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다.

  작업의 시간 제한 값은 7일 또는 10,080분 미만이어야 합니다. 그렇지 않으면 작업에서 예외가 발생합니다.

  값을 비워 두면 제한 시간은 기본적으로 2,880분으로 설정됩니다.

  제한 시간 값이 7일을 초과하는 기존 AWS Glue 작업은 기본적으로 7일로 설정됩니다. 예를 들어 배치 작업에 20일의 제한 시간을 지정한 경우 7일째 되는 날에 작업이 중지됩니다.

  스트리밍 작업은 유지 관리 기간을 설정한 경우 7일 후 유지 관리 기간 중에 작업이 다시 시작됩니다.

- `MaxCapacity` - 숫자(double)입니다.

  Glue 버전 1.0 이전 적업의 경우 표준 작업자 유형을 사용하여 이 작업을 실행할 때 할당할 수 있는 AWS Glue 데이터 처리 장치(DPU) 수입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  Glue 버전 2.0 이상 작업의 경우 `Maximum capacity`를 지정할 수 없습니다. 그 대신 `Worker type` 및 `Number of workers`를 지정해야 합니다.

  `WorkerType` 및 `NumberOfWorkers`를 사용하는 경우, `MaxCapacity`를 설정하지 마십시오.

  `MaxCapacity`에 할당할 수 있는 값은 Python 셸 작업을 실행하는지 또는 Apache Spark ETL 작업 또는 Apache Spark 스트리밍 ETL 작업을 실행하는지에 따라 다릅니다.

  - Python 셸 작업(`JobCommand.Name="pythonshell"`)을 지정하면 0.0625 또는 1 DPU를 할당할 수 있습니다. 기본값은 0.0625 DPU입니다.
  - Apache Spark ETL 작업(`JobCommand.Name="glueetl"`) 또는 Apache Spark 스트리밍 ETL 작업(`JobCommand.Name="gluestreaming"`) 을 지정하면 2에서 100 DPU를 할당할 수 있습니다. 기본값은 10 DPU입니다. 이 작업 유형에는 부분적인 DPU 할당을 사용할 수 없습니다.
- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업에 사용할 `SecurityConfiguration` 구조의 이름입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이 작업에서 사용할 태그입니다. 태그를 사용하여 작업에 대한 액세스를 제한할 수 있습니다. AWS Glue의 태그에 대한 자세한 내용은 개발자 안내서의 AWS Glue의 AWS 태그를 참조하세요.
- `NotificationProperty` – NotificationProperty 객체입니다.

  작업 알림의 구성 속성을 지정합니다.
- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Spark 작업에서 `GlueVersion`에 따라 AWS Glue에서 작업에 사용할 수 있는 Apache Spark 및 Python 버전이 결정됩니다. Python의 버전으로 Spark 유형의 작업에 대해 지원되는 버전을 확인할 수 있습니다.

  Ray 작업에서는 `GlueVersion`을 `4.0` 이상으로 설정해야 합니다. 그러나 Ray 작업에서 사용할 수 있는 Ray, Python 및 추가 라이브러리의 버전은 Job 명령의 `Runtime` 파라미터에 의해 결정됩니다.

  이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전에 대한 자세한 내용은 개발자 안내서의 Glue 버전을 참조하세요.

  Glue 버전 지정 없이 생성된 작업은 Glue 0.9로 기본 지정됩니다.
- `NumberOfWorkers` - 숫자(정수)입니다.

  작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  작업이 실행될 때 할당되는 미리 정의된 작업자 유형입니다. Spark 작업에 대해 G.1X, G.2X, G.4X, G.8X 또는 G.025X의 값을 허용합니다. Ray 작업에 대해 Z.2X 값을 허용합니다.

  - `G.1X` 작업자 유형의 경우, 각 작업자가 94GB의 디스크가 있는 1DPU(4개의 vCPU, 16GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
  - `G.2X` 작업자 유형의 경우, 각 작업자가 138GB의 디스크가 있는 2DPU(8개의 vCPU, 32GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
  - `G.4X` 작업자 유형의 경우, 각 작업자가 256GB의 디스크가 있는 4DPU(16개의 vCPU, 64GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 다음 AWS 리전에서 실행되는 Spark ETL 작업의 AWS Glue 버전 3.0 이상에서만 사용할 수 있습니다. 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오리건), 캐나다(중부), 아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 유럽(스페인), 유럽(스톡홀름), 남아메리카(상파울루).
  - `G.8X` 작업자 유형의 경우, 각 작업자가 512GB의 디스크가 있는 8DPU(32개의 vCPU, 128GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 `G.4X` 작업자 유형에 지원되는 동일한 AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
  - `G.025X` 작업자 유형의 경우, 각 작업자가 84GB의 디스크가 있는 0.25DPU(2개의 vCPU, 4GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 볼륨이 낮은 스트리밍 작업에 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 AWS Glue 버전 3.0 이상 스트리밍 작업에만 사용할 수 있습니다.
  - `Z.2X` 작업자 유형의 경우, 각 작업자는 128GB 디스크에서 2개의 M-DPU(vCPU 8개, 메모리 64GB)에 매핑되고, Autoscaler에 따라 최대 8개의 Ray 작업자를 제공합니다.
- `CodeGenConfigurationNodes` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #58과(와) 일치하는 UTF-8 문자열입니다.

각 값은 CodeGenConfigurationNode 객체입니다.

Glue Studio 시각적 구성 요소 및 Glue Studio 코드 생성의 기반이 되는 방향성 비순환 그래프의 표현입니다.

- `ExecutionClass` - 16바이트 미만의 UTF-8 문자열입니다(유효한 값: `FLEX="" | STANDARD=""`).

  작업이 표준 또는 유연한 실행 클래스로 실행되는지 여부를 나타냅니다. 표준 실행 클래스는 빠른 작업 시작 및 전용 리소스가 필요한 시간에 민감한 워크로드에 적합합니다.

  유연한 실행 클래스는 시작 및 완료 시간이 다를 수 있는 시간에 민감하지 않은 작업에 적합합니다.

  AWS Glue 버전 3.0 이상 및 명령 유형 `glueetl`을 사용하는 작업만 `ExecutionClass`가 `FLEX`로 설정됩니다. 유연한 실행 클래스는 Spark 작업에 사용할 수 있습니다.

- `SourceControlDetails` – SourceControlDetails 객체입니다.

  작업에 대한 소스 제어 구성에 대한 세부 정보로, 원격 리포지토리와의 작업 아티팩트 동기화를 허용합니다.

- `MaintenanceWindow` – Custom string pattern #34과(와) 일치하는 UTF-8 문자열입니다.

  이 필드는 스트리밍 작업에 대한 유지 관리 기간의 요일과 시간을 지정합니다. AWS Glue에서는 정기적으로 유지 관리 작업을 수행합니다. 이러한 유지 관리 기간 동안에는 AWS Glue에서 스트리밍 작업을 다시 시작해야 합니다.

  AWS Glue에서는 지정된 유지 관리 기간으로부터 3시간 이내에 작업을 다시 시작합니다. 예를 들어, 유지 관리 기간을 GMT 기준 월요일 오전 10시로 설정하면 작업이 GMT 기준 오전 10시에서 오후 1시 사이에 다시 시작됩니다.

- `ProfileName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업과 연결된 AWS Glue 사용 프로필의 이름입니다.

응답

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 정의를 위해 제공된 고유 이름입니다.

오류

- `InvalidInputException`
- `IdempotentParameterMismatchException`
- `AlreadyExistsException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

UpdateJob 작업(Python: update_job)

기존 작업 정의를 업데이트합니다. 이전 작업 정의를 이 정보로 완전히 덮어씁니다.

요청

- `JobName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트할 작업 정의 이름입니다.

- `JobUpdate` – 필수(Required): JobUpdate 객체입니다.

  작업 정의를 업데이트하여 값을 지정합니다. 지정되지 않은 구성은 제거되거나 기본값으로 재설정됩니다.

- `ProfileName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업과 연결된 AWS Glue 사용 프로필의 이름입니다.

응답

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트된 작업 정의 이름을 반환합니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

GetJob 작업(Python: get_job)

기존 작업 정의를 가져옵니다.

요청

- `JobName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  가져올 작업 정의 이름입니다.

응답

- `Job` – 작업 객체입니다.

  요청한 작업 정의.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobs 작업(Python: get_jobs)

현재 모든 작업 정의를 가져옵니다.

요청

- NextToken – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.
- MaxResults – 1~1,000의 숫자(정수)입니다.

  응답의 최대 크기입니다.

응답

- Jobs – 작업 객체의 배열입니다.

  작업 정의의 목록입니다.
- NextToken – UTF-8 문자열입니다.

  모든 작업 정의가 반환하지 않은 경우의 지속 토큰입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

DeleteJob 작업(Python: delete_job)

지정한 작업 정의를 삭제합니다. 작업 정의를 못 찾으면 어떤 예외도 없습니다.

요청

- JobName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 작업 정의 이름입니다.

응답

- JobName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제된 작업 정의 이름입니다.

오류

- InvalidInputException
- InternalServiceException
- OperationTimeoutException

ListJobs 작업(Python: list_jobs)

이 AWS 계정의 모든 작업 리소스 또는 지정된 태그를 가진 리소스를 검색합니다. 이 작업을 통해 계정에서 사용 가능한 리소스와 그 이름을 확인할 수 있습니다.

이 작업을 수행하면 응답에서 필터로 사용할 수 있는 선택 사항인 Tags 필드가 검색되기 때문에 태그가 지정된 리소스를 하나의 그룹으로 검색할 수 있습니다. 태그 필터링을 사용하기로 선택하면 태그가 포함된 리소스만 검색됩니다.

요청

- NextToken – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.
- MaxResults – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 최대 크기.
- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이렇게 태그가 지정된 리소스만 반환하도록 지정합니다.

응답

- JobNames – UTF-8 문자열의 배열입니다.

  계정의 모든 작업 또는 지정된 태그를 가진 작업의 이름입니다.

- NextToken – UTF-8 문자열입니다.

  반환된 목록이 사용가능한 마지막 지표를 포함하지 경우의 연속 토큰입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

BatchGetJobs 작업(Python: batch_get_jobs)

주어진 작업 이름 목록에 대한 리소스 메타데이터 목록을 반환합니다. ListJobs 작업을 호출한 후에는 권한이 부여된 데이터에 액세스하기 위해 이 작업을 호출할 수 있습니다. 이 작업은 태그를 사용하는 권한 조건을 포함해 모든 IAM 권한을 지원합니다.

요청

- JobNames – 필수(Required): UTF-8 문자열의 배열입니다.

  작업 이름(ListJobs 작업에서 반환된 이름일 수 있음)의 목록입니다.

응답

- Jobs – 작업 객체의 배열입니다.

  작업 정의 목록입니다.

- JobsNotFound – UTF-8 문자열의 배열입니다.

  찾을 수 없는 작업의 이름 목록입니다.

오류

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

작업 실행

작업 실행 API는 AWS Glue에서의 작업 실행 시작, 중지 또는 확인과 작업 북마크 재설정과 관련된 API 및 데이터 유형에 대해 설명합니다. 워크플로 및 작업 실행의 경우 작업 실행 기록을 90일 동안 액세스할 수 있습니다.

데이터 타입

- JobRun 구조
- 이전 구조
- JobBookmarkEntry 구조
- BatchStopJobRunSuccessfulSubmission 구조
- BatchStopJobRunError 구조
- NotificationProperty 구조

JobRun 구조

작업 실행에 관한 정보를 포함합니다.

필드

- `Id` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 실행의 ID.
- `Attempt` - 숫자(정수)입니다.

  이 작업을 실행하고자 시도했던 수입니다.
- `PreviousRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업의 이전 실행 ID입니다. 예를 들어 `StartJobRun` 작업에 지정된 `JobRunId`입니다.

- TriggerName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업을 시작한 트리거의 이름입니다.

- JobName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업에 사용된 작업 정의 이름입니다.

- JobMode – UTF-8 문자열입니다(유효한 값: SCRIPT="" | VISUAL="" | NOTEBOOK="").

  작업이 생성된 방법을 설명하는 모드입니다. 유효한 값은 다음과 같습니다.

  - SCRIPT - AWS Glue Studio 스크립트 편집기를 사용하여 작업을 생성했습니다.
  - VISUAL - AWS Glue Studio 시각적 편집기를 사용하여 작업을 생성했습니다.
  - NOTEBOOK - 대화형 세션 노트북을 사용하여 작업을 생성했습니다.

  JobMode 필드가 없거나 null인 경우 기본값으로 SCRIPT가 할당됩니다.

- JobRunQueuingEnabled – 부울입니다.

  해당 작업 실행에 대해 작업 실행 대기열을 활성화할지 여부를 지정합니다.

  값이 true이면 해당 작업 실행에 대해 작업 실행 큐가 활성화됩니다. false이거나 채워지지 않은 경우 작업 실행은 대기열에 포함되는 것으로 간주되지 않습니다.

- StartedOn – 타임스탬프입니다.

  이 작업이 시작된 날짜 및 시간.

- LastModifiedOn – 타임스탬프입니다.

  이 작업 실행이 수정된 마지막 시간입니다.

- CompletedOn – 타임스탬프입니다.

  이 작업 실행이 완료된 날짜 및 시간입니다.

- JobRunState – UTF-8 문자열입니다(유효한 값: STARTING | RUNNING | STOPPING | STOPPED | SUCCEEDED | FAILED | TIMEOUT | ERROR | WAITING | EXPIRED).

  작업 실행의 현재 상태입니다. 비정상적으로 종료된 작업의 상태에 대한 자세한 내용은 AWS Glue 작업 실행 상태를 참조하세요.

- Arguments – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

각 값은 UTF-8 문자열입니다.

작업과 연결되어 있는 작업 인수입니다. 이 작업 실행에서 작업 정의 자체에 설정된 기본 인수를 바꿉니다.

AWS Glue 자체가 사용하는 인수는 물론 사용자의 작업 실행 스크립트가 사용하는 인수를 지정할 수 있습니다.

작업 인수가 로깅될 수 있습니다. 일반 텍스트 보안 암호를 인수로 전달하지 마세요. 보안 암호를 작업 내에 보관하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

자체 작업 인수를 지정하고 사용하는 방법에 대한 자세한 내용은 개발자 가이드의 Python에서 AWS Glue Glue API 호출을 참조하세요.

Spark 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 AWS Glue에서 사용하는 특별 파라미터 주제를 참조하세요.

Ray 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 Ray 작업에서 작업 파라미터 사용을 참조하세요.

- `ErrorMessage` – UTF-8 문자열입니다.

  이 작업의 실행과 연결된 오류 메시지입니다.

- `PredecessorRuns` – 이전 기록 객체의 배열입니다.

  이 작업을 실행하기 이전의 기록입니다.

- `AllocatedCapacity` - 숫자(정수)입니다.

  이 필드는 더 이상 사용되지 않습니다. 대신 `MaxCapacity`을 사용하세요.

  이 JobRun에 할당된 AWS Glue 데이터 처리 장치(DPU) 수입니다. 2~100DPU가 할당될 수 있습니다. 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

- `ExecutionTime` - 숫자(정수)입니다.

  이 작업이 리소스를 소비한 시간(초).

- `Timeout` – 1 이상의 숫자(정수)입니다.

JobRun 제한 시간(분)입니다. 작업을 실행하여 리소스를 소비하여 중지되기 전에 TIMEOUT 상태로 들어가는 최대 시간입니다. 이 값은 상위 작업에 설정된 제한 시간 값을 재정의합니다.

작업의 시간 제한 값은 7일 또는 10,080분 미만이어야 합니다. 그렇지 않으면 작업에서 예외가 발생합니다.

값을 비워 두면 제한 시간은 기본적으로 2,880분으로 설정됩니다.

제한 시간 값이 7일을 초과하는 기존 AWS Glue 작업은 기본적으로 7일로 설정됩니다. 예를 들어 배치 작업에 20일의 제한 시간을 지정한 경우 7일째 되는 날에 작업이 중지됩니다.

스트리밍 작업은 유지 관리 기간을 설정한 경우 7일 후 유지 관리 기간 중에 작업이 다시 시작됩니다.

- MaxCapacity - 숫자(double)입니다.

  Glue 버전 1.0 이전 적업의 경우 표준 작업자 유형을 사용하여 이 작업을 실행할 때 할당할 수 있는 AWS Glue 데이터 처리 장치(DPU) 수입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  Glue 버전 2.0 이상 작업의 경우 Maximum capacity를 지정할 수 없습니다. 그 대신 Worker type 및 Number of workers를 지정해야 합니다.

  WorkerType 및 NumberOfWorkers를 사용하는 경우, MaxCapacity를 설정하지 마십시오.

  MaxCapacity에 할당할 수 있는 값은 Python 셸 작업을 실행하는지 또는 Apache Spark ETL 작업 또는 Apache Spark 스트리밍 ETL 작업을 실행하는지에 따라 다릅니다.

  - Python 셸 작업(JobCommand.Name="pythonshell")을 지정하면 0.0625 또는 1 DPU를 할당할 수 있습니다. 기본값은 0.0625 DPU입니다.
  - Apache Spark ETL 작업(JobCommand.Name="glueetl") 또는 Apache Spark 스트리밍 ETL 작업(JobCommand.Name="gluestreaming") 을 지정하면 2에서 100 DPU를 할당할 수 있습니다. 기본값은 10 DPU입니다. 이 작업 유형에는 부분적인 DPU 할당을 사용할 수 없습니다.

- WorkerType – UTF-8 문자열입니다(유효한 값: Standard="" | G.1X="" | G.2X="" | G.025X="" | G.4X="" | G.8X="" | Z.2X="").

  작업이 실행될 때 할당되는 미리 정의된 작업자 유형입니다. Spark 작업에 대해 G.1X, G.2X, G.4X, G.8X 또는 G.025X의 값을 허용합니다. Ray 작업에 대해 Z.2X 값을 허용합니다.

  - G.1X 작업자 유형의 경우, 각 작업자가 94GB의 디스크가 있는 1DPU(4개의 vCPU, 16GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고

비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.

- `G.2X` 작업자 유형의 경우, 각 작업자가 138GB의 디스크가 있는 2DPU(8개의 vCPU, 32GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
- `G.4X` 작업자 유형의 경우, 각 작업자가 256GB의 디스크가 있는 4DPU(16개의 vCPU, 64GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(스톡홀름)과 같은AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
- `G.8X` 작업자 유형의 경우, 각 작업자가 512GB의 디스크가 있는 8DPU(32개의 vCPU, 128GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 `G.4X` 작업자 유형에 지원되는 동일한 AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
- `G.025X` 작업자 유형의 경우, 각 작업자가 84GB의 디스크가 있는 0.25DPU(2개의 vCPU, 4GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 볼륨이 낮은 스트리밍 작업에 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 AWS Glue 버전 3.0 이상 스트리밍 작업에만 사용할 수 있습니다.
- `Z.2X` 작업자 유형의 경우, 각 작업자는 128GB 디스크에서 2개의 M-DPU(vCPU 8개, 메모리 64GB)에 매핑되고, Autoscaler에 따라 최대 8개의 Ray 작업자를 제공합니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 실행에 사용할 `SecurityConfiguration` 구조의 이름입니다.

- `LogGroupName` – UTF-8 문자열입니다.

  AWS KMS를 사용하여 Amazon CloudWatch에서 서버 측 암호화가 가능한 보안 로깅용 로그 그룹의 이름입니다. 이 이름은 `/aws-glue/jobs/`일 수 있으며, 그 기본 암호화는 `NONE`입니다. 역할 이름과 `SecurityConfiguration` 이름(즉, `/aws-glue/jobs-yourRoleName-`

yourSecurityConfigurationName/)을 추가하면 로그 그룹 암호화에 이 보안 구성이 사용됩니다.

- NotificationProperty – NotificationProperty 객체입니다.

  작업 실행 알림의 구성 속성을 지정합니다.

- GlueVersion – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Spark 작업에서 GlueVersion에 따라 AWS Glue에서 작업에 사용할 수 있는 Apache Spark 및 Python 버전이 결정됩니다. Python의 버전으로 Spark 유형의 작업에 대해 지원되는 버전을 확인할 수 있습니다.

  Ray 작업에서는 GlueVersion을 4.0 이상으로 설정해야 합니다. 그러나 Ray 작업에서 사용할 수 있는 Ray, Python 및 추가 라이브러리의 버전은 Job 명령의 Runtime 파라미터에 의해 결정됩니다.

  이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전에 대한 자세한 내용은 개발자 안내서의 Glue 버전을 참조하세요.

  Glue 버전 지정 없이 생성된 작업은 Glue 0.9로 기본 지정됩니다.

- DPUSeconds - 숫자(double)입니다.

  이 필드는 실행 클래스가 FLEX인 작업 실행이거나 Auto Scaling이 활성화된 경우에 설정할 수 있으며, 작업 실행 수명 주기 동안 각 실행기가 실행된 총 시간(초)에 DPU 계수(작업자는 1(G.1X), 2(G.2X) 또는 0.25(G.025X))를 곱한 값을 나타냅니다. 이 값은 Auto Scaling 작업의 경우처럼 executionEngineRuntime * MaxCapacity와 다를 수 있습니다. 지정된 시간에 실행 중인 실행기 수가 MaxCapacity보다 작을 수 있기 때문입니다. 따라서, DPUSeconds 값이 executionEngineRuntime * MaxCapacity보다 작을 수 있습니다.

- ExecutionClass - 16바이트 미만의 UTF-8 문자열입니다(유효한 값: FLEX="" | STANDARD="").

  작업이 표준 또는 유연한 실행 클래스로 실행되는지 여부를 나타냅니다. 표준 실행 클래스는 빠른 작업 시작 및 전용 리소스가 필요한 시간에 민감한 워크로드에 적합합니다.

  유연한 실행 클래스는 시작 및 완료 시간이 다를 수 있는 시간에 민감하지 않은 작업에 적합합니다.

  AWS Glue 버전 3.0 이상 및 명령 유형 glueetl을 사용하는 작업만 ExecutionClass가 FLEX로 설정됩니다. 유연한 실행 클래스는 Spark 작업에 사용할 수 있습니다.

- MaintenanceWindow – Custom string pattern #34과(와) 일치하는 UTF-8 문자열입니다.

이 필드는 스트리밍 작업에 대한 유지 관리 기간의 요일과 시간을 지정합니다. AWS Glue에서는 정기적으로 유지 관리 작업을 수행합니다. 이러한 유지 관리 기간 동안에는 AWS Glue에서 스트리밍 작업을 다시 시작해야 합니다.

AWS Glue에서는 지정된 유지 관리 기간으로부터 3시간 이내에 작업을 다시 시작합니다. 예를 들어, 유지 관리 기간을 GMT 기준 월요일 오전 10시로 설정하면 작업이 GMT 기준 오전 10시에서 오후 1시 사이에 다시 시작됩니다.

- `ProfileName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행과 연결된 AWS Glue 사용 프로필의 이름입니다.
- `StateDetail` – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  이 필드에는 작업 실행 상태와 관련된 세부 정보가 들어 있습니다. 이 필드는 null을 사용할 수 없습니다.

  예를 들어 작업 실행 큐의 결과로 작업 실행이 대기 상태인 경우 필드에는 작업 실행이 해당 상태인 이유가 표시됩니다.

이전 구조

이 작업을 실행할 수 있도록 조건적 트리거를 조건으로 한 작업입니다.

필드

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 이전 작업에 사용된 작업 정의 이름입니다.
- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업의 이전 작업 실행 ID입니다.

JobBookmarkEntry 구조

작업이 프로세싱을 다시 시작할 수 있는 포인트를 정의합니다.

필드

- `JobName` – UTF-8 문자열입니다.

문제의 작업 이름입니다.

- `Version` - 숫자(정수)입니다.

  작업의 버전입니다.

- `Run` - 숫자(정수)입니다.

  실행 ID 숫자입니다.

- `Attempt` - 숫자(정수)입니다.

  시도 ID 숫자입니다.

- `PreviousRunId` – UTF-8 문자열입니다.

  이전의 작업 실행과 연결된 고유의 실행 식별자입니다.

- `RunId` – UTF-8 문자열입니다.

  실행 ID 숫자입니다.

- `JobBookmark` – UTF-8 문자열입니다.

  그 자체를 즐겨찾습니다.

BatchStopJobRunSuccessfulSubmission 구조

지정된 `JobRun`을 중지한 성공적인 요청을 기록합니다.

필드

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업이 중지된 작업 정의 이름입니다.

- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  중지된 작업 실행의 `JobRunId`입니다.

BatchStopJobRunError 구조

지정된 작업을 중지하고자 할 때 발생하는 오류를 기록합니다.

필드

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 작업 실행에 사용된 작업 정의 이름입니다.
- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 작업 실행 `JobRunId`입니다.
- `ErrorDetail` – ErrorDetail 객체입니다.

  발생한 오류에 대한 추가 세부 정보를 지정합니다.

NotificationProperty 구조

알림의 구성 속성을 지정합니다.

필드

- `NotifyDelayAfter` – 1 이상의 숫자(정수)입니다.

  작업 실행 시작 후 작업 실행 대기 알림을 전송하기 전까지 대기하는 시간(분)입니다.

운영

- StartJobRun 작업(Python: start_job_run)
- BatchStopJobRun 작업(Python: batch_stop_job_run)
- GetJobRun 작업(Python: get_job_run)
- GetJobRuns 작업(Python: get_job_runs)
- GetJobBookmark 작업(Python: get_job_bookmark)
- GetJobBookmarks 작업(Python: get_job_bookmarks)
- ResetJobBookmark 작업(Python: reset_job_bookmark)

StartJobRun 작업(Python: start_job_run)

작업 정의를 사용한 작업 시작하기

요청

- `JobName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용할 작업 정의의 이름입니다.

- `JobRunQueuingEnabled` – 부울입니다.

  해당 작업 실행에 대해 작업 실행 대기열을 활성화할지 여부를 지정합니다.

  값이 true이면 해당 작업 실행에 대해 작업 실행 큐가 활성화됩니다. false이거나 채워지지 않은 경우 작업 실행은 대기열에 포함되는 것으로 간주되지 않습니다.

- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  재시도할 이전 `JobRun`의 ID입니다.

- `Arguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  작업과 연결되어 있는 작업 인수입니다. 이 작업 실행에서 작업 정의 자체에 설정된 기본 인수를 바꿉니다.

  AWS Glue 자체가 사용하는 인수는 물론 사용자의 작업 실행 스크립트가 사용하는 인수를 지정할 수 있습니다.

  작업 인수가 로깅될 수 있습니다. 일반 텍스트 보안 암호를 인수로 전달하지 마세요. 보안 암호를 작업 내에 보관하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

  자체 작업 인수를 지정하고 사용하는 방법에 대한 자세한 내용은 개발자 가이드의 Python에서 AWS Glue Glue API 호출을 참조하세요.

  Spark 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 AWS Glue에서 사용하는 특별 파라미터 주제를 참조하세요.

  Ray 작업을 구성할 때 이 필드에 제공할 수 있는 인수에 대한 자세한 내용은 개발자 안내서의 Ray 작업에서 작업 파라미터 사용을 참조하세요.

- `AllocatedCapacity` - 숫자(정수)입니다.

이 필드는 더 이상 사용되지 않습니다. 대신 MaxCapacity을 사용하세요.

이 JobRun에 할당할 AWS Glue 데이터 처리 장치(DPU) 수입니다. 최소 2DPU를 할당할 수 있습니다. 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

- Timeout – 1 이상의 숫자(정수)입니다.

  JobRun 제한 시간(분)입니다. 작업을 실행하여 리소스를 소비하여 중지되기 전에 TIMEOUT 상태로 들어가는 최대 시간입니다. 이 값은 상위 작업에 설정된 제한 시간 값을 재정의합니다.

  작업의 시간 제한 값은 7일 또는 10,080분 미만이어야 합니다. 그렇지 않으면 작업에서 예외가 발생합니다.

  값을 비워 두면 제한 시간은 기본적으로 2,880분으로 설정됩니다.

  제한 시간 값이 7일을 초과하는 기존 AWS Glue 작업은 기본적으로 7일로 설정됩니다. 예를 들어 배치 작업에 20일의 제한 시간을 지정한 경우 7일째 되는 날에 작업이 중지됩니다.

  스트리밍 작업은 유지 관리 기간을 설정한 경우 7일 후 유지 관리 기간 중에 작업이 다시 시작됩니다.

- MaxCapacity - 숫자(double)입니다.

  Glue 버전 1.0 이전 적업의 경우 표준 작업자 유형을 사용하여 이 작업을 실행할 때 할당할 수 있는 AWS Glue 데이터 처리 장치(DPU) 수입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  Glue 버전 2.0 이상 작업의 경우 Maximum capacity를 지정할 수 없습니다. 그 대신 Worker type 및 Number of workers를 지정해야 합니다.

  WorkerType 및 NumberOfWorkers를 사용하는 경우, MaxCapacity를 설정하지 마십시오.

  MaxCapacity에 할당할 수 있는 값은 Python 셸 작업을 실행하는지 또는 Apache Spark ETL 작업 또는 Apache Spark 스트리밍 ETL 작업을 실행하는지에 따라 다릅니다.

  - Python 셸 작업(JobCommand.Name="pythonshell")을 지정하면 0.0625 또는 1 DPU를 할당할 수 있습니다. 기본값은 0.0625 DPU입니다.
  - Apache Spark ETL 작업(JobCommand.Name="glueetl") 또는 Apache Spark 스트리밍 ETL 작업(JobCommand.Name="gluestreaming") 을 지정하면 2에서 100 DPU를 할당할 수 있습니다. 기본값은 10 DPU입니다. 이 작업 유형에는 부분적인 DPU 할당을 사용할 수 없습니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 실행에 사용할 `SecurityConfiguration` 구조의 이름입니다.

- `NotificationProperty` – NotificationProperty 객체입니다.

  작업 실행 알림의 구성 속성을 지정합니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  작업이 실행될 때 할당되는 미리 정의된 작업자 유형입니다. Spark 작업에 대해 G.1X, G.2X, G.4X, G.8X 또는 G.025X의 값을 허용합니다. Ray 작업에 대해 Z.2X 값을 허용합니다.

  - `G.1X` 작업자 유형의 경우, 각 작업자가 94GB의 디스크가 있는 1DPU(4개의 vCPU, 16GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
  - `G.2X` 작업자 유형의 경우, 각 작업자가 138GB의 디스크가 있는 2DPU(8개의 vCPU, 32GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
  - `G.4X` 작업자 유형의 경우, 각 작업자가 256GB의 디스크가 있는 4DPU(16개의 vCPU, 64GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(스톡홀름)과 같은AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
  - `G.8X` 작업자 유형의 경우, 각 작업자가 512GB의 디스크가 있는 8DPU(32개의 vCPU, 128GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 `G.4X` 작업자 유형에 지원되는 동일한 AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
  - `G.025X` 작업자 유형의 경우, 각 작업자가 84GB의 디스크가 있는 0.25DPU(2개의 vCPU, 4GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 볼륨이 낮은 스트리밍 작업에 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 AWS Glue 버전 3.0 이상 스트리밍 작업에만 사용할 수 있습니다.

  - Z.2X 작업자 유형의 경우, 각 작업자는 128GB 디스크에서 2개의 M-DPU(vCPU 8개, 메모리 64GB)에 매핑되고, Autoscaler에 따라 최대 8개의 Ray 작업자를 제공합니다.
- NumberOfWorkers - 숫자(정수)입니다.

  작업이 실행될 때 할당되는 정의된 workerType의 작업자 수입니다.
- ExecutionClass - 16바이트 미만의 UTF-8 문자열입니다(유효한 값: FLEX="" | STANDARD="").

  작업이 표준 또는 유연한 실행 클래스로 실행되는지 여부를 나타냅니다. 표준 실행 클래스는 빠른 작업 시작 및 전용 리소스가 필요한 시간에 민감한 워크로드에 적합합니다.

  유연한 실행 클래스는 시작 및 완료 시간이 다를 수 있는 시간에 민감하지 않은 작업에 적합합니다.

  AWS Glue 버전 3.0 이상 및 명령 유형 glueetl을 사용하는 작업만 ExecutionClass가 FLEX로 설정됩니다. 유연한 실행 클래스는 Spark 작업에 사용할 수 있습니다.
- ProfileName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행과 연결된 AWS Glue 사용 프로필의 이름입니다.

응답

- JobRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 실행에 할당된 ID.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- ResourceNumberLimitExceededException
- ConcurrentRunsExceededException

BatchStopJobRun 작업(Python: batch_stop_job_run)

지정된 작업을 정의하기 위해서 하나 이상의 작업을 중지합니다.

요청

- JobName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업이 실행되지 않고 중지하기 위한 작업 정의 이름입니다.
- JobRunIds – 필수(Required): 1~25개 문자열의 UTF-8 문자열의 배열입니다.

  이 작업을 정의하기 위해서 중지되어야 하는 JobRunIds의 목록입니다.

응답

- SuccessfulSubmissions – BatchStopJobRunSuccessfulSubmission 객체의 배열입니다.

  중지할 목적으로 성공적으로 제출된 JobRun입니다.
- Errors – BatchStopJobRunError 객체의 배열입니다.

  JobRuns를 중지할 때 발생하는 오류 목록은 오류에 대한 상세 정보와 함께 각 오류가 발생한 JobRunId를 포함합니다.

오류

- InvalidInputException
- InternalServiceException
- OperationTimeoutException

GetJobRun 작업(Python: get_job_run)

작업 실행 시 메타데이터 가져오기 워크플로 및 작업 실행의 경우 작업 실행 기록을 365일 동안 액세스할 수 있습니다.

요청

- JobName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행 중인 작업 정의의 이름입니다.

- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 실행의 ID.

- PredecessorsIncluded – 부울입니다.

  이전 작업 실행 목록이 반환되어야 하면 True입니다.

응답

- JobRun – JobRun 객체입니다.

  요청한 작업 실행 메타데이터.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

GetJobRuns 작업(Python: get_job_runs)

주어진 작업 정의의 모든 실행 시 메타데이터 가져오기

GetJobRuns는 작업 실행을 시간순으로 반환합니다. 이때 최신 작업을 먼저 반환합니다.

요청

- JobName – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  모든 작업을 실행하기 위한 작업 정의 이름입니다.

- NextToken – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

- MaxResults - 1 이상 200 이하의 숫자(정수)입니다.

응답의 최대 크기입니다.

응답

- `JobRuns` – JobRun 객체의 배열입니다.

  작업 실행 메타데이터 객체의 목록입니다.
- `NextToken` – UTF-8 문자열입니다.

  요청된 모든 작업 실행이 반환하지 않은 경우의 지속 토큰입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobBookmark 작업(Python: get_job_bookmark)

작업 북마크 항목에 대한 정보를 반환합니다.

작업 북마크에 대한 자세한 내용을 알아보려면 다음을 참조하세요.

- 처리된 데이터를 작업 북마크로 추적
- AWS Glue에서 사용하는 작업 파라미터
- 작업 구조

요청

- `JobName` – 필수(Required): UTF-8 문자열입니다.

  문제의 작업 이름입니다.
- `Version` - 숫자(정수)입니다.

  작업의 버전입니다.

- `RunId` – UTF-8 문자열입니다.

  이 작업 실행과 연결된 고유의 실행 식별자입니다.

응답

- `JobBookmarkEntry` – JobBookmarkEntry 객체입니다.

  작업이 처리를 다시 시작할 수 있는 포인트를 정의하는 구조입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ValidationException`

GetJobBookmarks 작업(Python: get_job_bookmarks)

작업 북마크 항목에 대한 정보를 반환합니다. 목록은 버전 번호가 감소되는 순서로 정렬됩니다.

작업 북마크에 대한 자세한 내용을 알아보려면 다음을 참조하세요.

- 처리된 데이터를 작업 북마크로 추적
- AWS Glue에서 사용하는 작업 파라미터
- 작업 구조

요청

- `JobName` – 필수(Required): UTF-8 문자열입니다.

  문제의 작업 이름입니다.
- `MaxResults` - 숫자(정수)입니다.

  응답의 최대 크기입니다.
- `NextToken` - 숫자(정수)입니다.

이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `JobBookmarkEntries` – JobBookmarkEntry 객체의 배열입니다.

  작업이 처리를 다시 시작할 수 있는 포인트를 정의하는 작업 북마크 항목의 목록입니다.
- `NextToken` - 숫자(정수)입니다.

  모든 항목이 반환된 경우에는 1, 요청된 작업 실행 중 일부가 반환되지 않은 경우 1보다 큰 값이 있는 연속 토큰입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

ResetJobBookmark 작업(Python: reset_job_bookmark)

북마크 입력 재설정

작업 북마크에 대한 자세한 내용을 알아보려면 다음을 참조하세요.

- 처리된 데이터를 작업 북마크로 추적
- AWS Glue에서 사용하는 작업 파라미터
- 작업 구조

요청

- `JobName` – 필수(Required): UTF-8 문자열입니다.

  문제의 작업 이름입니다.
- `RunId` – UTF-8 문자열입니다.

이 작업 실행과 연결된 고유의 실행 식별자입니다.

응답

- JobBookmarkEntry – JobBookmarkEntry 객체입니다.

  북마크 입력 재설정

오류

- EntityNotFoundException
- InvalidInputException
- InternalServiceException
- OperationTimeoutException

트리거

트리거 API는 AWS Glue에서의 작업 트리거 생성, 업데이트 또는 삭제 및 시작/중지와 관련된 API 및 데이터 유형에 대해 설명합니다.

데이터 타입

- 트리거 구조
- TriggerUpdate 구조
- 조건자 구조
- 조건 구조
- 작업 구조
- EventBatchingCondition 구조

트리거 구조

특정 트리거에 대한 정보

필드

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

트리거의 이름입니다.

- `WorkflowName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  트리거와 연결된 워크플로의 이름입니다.

- `Id` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  추후 사용 예약.

- `Type` – UTF-8 문자열입니다(유효 값: `SCHEDULED` | `CONDITIONAL` | `ON_DEMAND` | `EVENT`).

  트리거 유형입니다.

- `State` – UTF-8 문자열입니다(유효 값: `CREATING` | `CREATED` | `ACTIVATING` | `ACTIVATED` | `DEACTIVATING` | `DEACTIVATED` | `DELETING` | `UPDATING`).

  트리거 현재 테이블 상태

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  이 트리거에 대한 설명입니다.

- `Schedule` – UTF-8 문자열입니다.

  일정을 지정하는 데 사용되는 `cron` 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 `cron(15 12 * * ? *)`을 지정합니다.

- `Actions` – 작업 객체의 배열입니다.

  이 트리거에 의해 시작되는 작업

- `Predicate` – 조건자 객체입니다.

  트리거가 실행되는 시기를 결정하는 이 트리거의 조건자입니다.

- `EventBatchingCondition` – EventBatchingCondition 객체입니다.

  EventBridge 이벤트 트리거가 실행되기 전에 충족되어야 하는 배치 조건(수신된 이벤트의 지정된 수 또는 배치 기간이 만료됨).

TriggerUpdate 구조

트리거를 업데이트하기 위해 정보를 제공하는 데 사용되는 구조입니다. 이 객체는 이전 트리거 정의를 완전히 덮어써서 업데이트합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  추후 사용 예약.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  이 트리거에 대한 설명입니다.

- `Schedule` – UTF-8 문자열입니다.

  일정을 지정하는 데 사용되는 `cron` 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 `cron(15 12 * * ? *)`을 지정합니다.

- `Actions` – 작업 객체의 배열입니다.

  이 트리거에 의해 시작되는 작업

- `Predicate` – 조건자 객체입니다.

  트리거가 실행되는 시기를 결정하는 이 트리거의 조건자입니다.

- `EventBatchingCondition` – EventBatchingCondition 객체입니다.

  EventBridge 이벤트 트리거가 실행되기 전에 충족되어야 하는 배치 조건(수신된 이벤트의 지정된 수 또는 배치 기간이 만료됨).

조건자 구조

트리거가 실행되는 시기를 결정하는 트리거의 조건자를 정의합니다.

필드

- `Logical` – UTF-8 문자열입니다(유효한 값: `AND` | `ANY`).

  오직 하나의 조건이 기록되어 있다면 조건부 필드입니다. 다수의 조건이 있을 때 이 필드가 필요합니다.

- Conditions – Condition 객체의 배열이며 구조는 500개 이하입니다.

  트리거가 언제 발생하는지 확인하는 조건 목록입니다.

조건 구조

어떤 트리거가 시작되는지에 대한 조건을 결정합니다.

필드

- LogicalOperator – UTF-8 문자열입니다(유효한 값: EQUALS).

  논리 연산자

- JobName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  JobRuns에 조건이 적용되고 트리거가 대기하는 작업의 이름입니다.

- State – UTF-8 문자열입니다(유효한 값: STARTING | RUNNING | STOPPING | STOPPED | SUCCEEDED | FAILED | TIMEOUT | ERROR | WAITING | EXPIRED).

  조건 상태입니다. 현재 트리거가 수신할 수 있는 유일한 작업은 SUCCEEDED, STOPPED, FAILED 및 TIMEOUT입니다. 트리거가 수신할 수 있는 유일한 Crawler는 SUCCEEDED, FAILED 및 CANCELLED입니다.

- CrawlerName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 조건이 적용되는 크롤러의 이름입니다.

- CrawlState – UTF-8 문자열입니다(유효 값: RUNNING | CANCELLING | CANCELLED | SUCCEEDED | FAILED | ERROR).

  이 조건이 적용되는 크롤러의 상태입니다.

작업 구조

트리거에 의해 시작한 작업을 결정합니다.

필드

- JobName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

실행할 작업의 이름입니다.

- `Arguments` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  이 트리거가 작동할 때 사용되는 작업 인수입니다. 이 작업 실행에서 작업 정의 자체에 설정된 기본 인수를 바꿉니다.

  AWS Glue 자체가 사용하는 인수는 물론 사용자의 작업 실행 스크립트가 사용하는 인수를 지정할 수 있습니다.

  자체 작업 인수를 지정하고 사용하는 방법에 대한 자세한 내용은 개발자 가이드의 Python에서 AWS Glue Glue API 호출을 참조하세요.

  AWS Glue가 작업을 설정하는 데 사용하는 키 값 페어에 대한 자세한 내용은 개발자 가이드의 AWS Glue가 사용하는 특정 파라미터를 참조하세요.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  `JobRun` 제한 시간(분)입니다. 작업을 실행하여 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다. 상위 작업에 설정된 제한 시간 값을 재정의합니다.

  작업의 시간 제한 값은 7일 또는 10,080분 미만이어야 합니다. 그렇지 않으면 작업에서 예외가 발생합니다.

  값을 비워 두면 제한 시간은 기본적으로 2,880분으로 설정됩니다.

  제한 시간 값이 7일을 초과하는 기존 AWS Glue 작업은 기본적으로 7일로 설정됩니다. 예를 들어 배치 작업에 20일의 제한 시간을 지정한 경우 7일째 되는 날에 작업이 중지됩니다.

  스트리밍 작업은 유지 관리 기간을 설정한 경우 7일 후 유지 관리 기간 중에 작업이 다시 시작됩니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업에 사용할 `SecurityConfiguration` 구조의 이름입니다.

- `NotificationProperty` – NotificationProperty 객체입니다.

  작업 실행 알림의 구성 속성을 지정합니다.

- `CrawlerName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업에 사용할 크롤러의 이름입니다.

EventBatchingCondition 구조

EventBridge 이벤트 트리거가 실행되기 전에 충족되어야 하는 배치 조건(수신된 이벤트의 지정된 수 또는 배치 기간이 만료됨).

필드

- `BatchSize` – 필수(Required): 1~100의 숫자(정수)입니다.

  EventBridge 이벤트 트리거가 실행되기 전에 Amazon EventBridge에서 수신해야 하는 이벤트 수입니다.
- `BatchWindow` – 1~900의 숫자(정수)입니다.

  EventBridge 이벤트 트리거가 실행된 후의 기간(초)입니다. 첫 번째 이벤트가 수신되면 기간이 시작됩니다.

운영

- CreateTrigger 작업(Python: create_trigger)
- StartTrigger 작업(Python: start_trigger)
- GetTrigger 작업(Python: get_trigger)
- GetTrigger 작업(Python: get_triggers)
- UpdateTrigger 작업(Python: update_trigger)
- StopTrigger 작업(Python: stop_trigger)
- DeleteTrigger 작업(Python: delete_trigger)
- ListTriggers 작업(Python: list_triggers)
- BatchGetTriggers 작업(Python: batch_get_triggers)

CreateTrigger 작업(Python: create_trigger)

새로운 트리거를 만듭니다.

작업 인수가 로깅될 수 있습니다. 일반 텍스트 보안 암호를 인수로 전달하지 마세요. 보안 암호를 작업 내에 보관하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  트리거의 이름입니다.

- WorkflowName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  트리거와 연결된 워크플로의 이름입니다.

- Type – 필수(Required): UTF-8 문자열입니다(유효 값: SCHEDULED | CONDITIONAL | ON_DEMAND | EVENT).

  새로운 트리거의 유형입니다.

- Schedule – UTF-8 문자열입니다.

  일정을 지정하는 데 사용되는 cron 표현식입니다(작업 및 크롤러의 시간 기반 일정 참조). 예를 들어, 매일 오후 12시 15분(UTC)에 실행하려면 cron(15 12 * * ? *)을 지정합니다.

  트리거 유형이 SCHEDULED(예정)되면 필드가 필요합니다.

- Predicate – 조건자 객체입니다.

  새로운 트리거가 시작할 시기를 지정하는 조건자입니다.

  트리거 유형이 CONDITIONAL이면 이 필드는 필수입니다.

- Actions – 필수(Required): 작업 객체의 배열입니다.

  트리거가 발생하면 이 트리거가 시작되는 작업입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  새로운 트리거에 대한 설명.

- StartOnCreation – 부울입니다.

생성 시 SCHEDULED 및 CONDITIONAL 트리거를 시작하려면 true로 설정합니다. ON_DEMAND 트리거에는 True가 지원되지 않습니다.

- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이 트리거에서 사용할 태그입니다. 태그를 사용하여 트리거에 대한 액세스를 제한할 수 있습니다. AWS Glue의 태그에 대한 자세한 내용은 개발자 안내서의 AWS Glue의 AWS 태그를 참조하세요.
- EventBatchingCondition – EventBatchingCondition 객체입니다.

  EventBridge 이벤트 트리거가 실행되기 전에 충족되어야 하는 배치 조건(수신된 이벤트의 지정된 수 또는 배치 기간이 만료됨).

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  트리거의 이름입니다.

오류

- AlreadyExistsException
- EntityNotFoundException
- InvalidInputException
- IdempotentParameterMismatchException
- InternalServiceException
- OperationTimeoutException
- ResourceNumberLimitExceededException
- ConcurrentModificationException

StartTrigger 작업(Python: start_trigger)

기존 트리거 시작. 다른 유형의 트리거가 시작하는 방법에 대한 자세한 내용은 작업 트리거를 참조하십시오.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  시작할 트리거의 이름입니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  시작된 트리거의 이름입니다.

오류

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

GetTrigger 작업(Python: get_trigger)

트리거의 정의를 검색합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  가져올 트리거의 이름입니다.

응답

- `Trigger` – 트리거 객체입니다.

  요청한 트리거 정의입니다.

오류

- EntityNotFoundException
- InvalidInputException
- InternalServiceException
- OperationTimeoutException

GetTrigger 작업(Python: get_triggers)

이 작업과 연결된 모든 트리거를 얻습니다.

요청

- NextToken – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.
- DependentJobName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  트리거를 가져올 작업 이름입니다. 이 작업을 시작할 수 있는 트리거가 반환되고, 이런 트리거가 없으면 모든 트리거가 반환됩니다.
- MaxResults - 1 이상 200 이하의 숫자(정수)입니다.

  응답의 최대 크기입니다.

응답

- Triggers – 트리거 객체의 배열입니다.

  지정된 작업에 대한 트리거 목록.
- NextToken – UTF-8 문자열입니다.

  모든 요청된 트리거가 반환하지 않은 경우의 지속 토큰입니다.

오류

- EntityNotFoundException
- InvalidInputException

- InternalServiceException
- OperationTimeoutException

UpdateTrigger 작업(Python: update_trigger)

트리거 정의 업데이트

작업 인수가 로깅될 수 있습니다. 일반 텍스트 보안 암호를 인수로 전달하지 마세요. 보안 암호를 작업 내에 보관하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트할 트리거의 이름입니다.
- TriggerUpdate – 필수(Required): TriggerUpdate 객체입니다.

  트리거를 업데이트하여 얻은 새로운 값입니다.

응답

- Trigger – 트리거 객체입니다.

  결과 트리거 정의입니다.

오류

- InvalidInputException
- InternalServiceException
- EntityNotFoundException
- OperationTimeoutException
- ConcurrentModificationException

StopTrigger 작업(Python: stop_trigger)

지정된 트리거를 중지합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  중지할 트리거의 이름입니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  중지된 트리거의 이름입니다.

오류

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

DeleteTrigger 작업(Python: delete_trigger)

지정된 트리거를 삭제합니다. 트리거를 못 찾으면 어떤 예외도 없습니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 트리거의 이름입니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제된 트리거의 이름입니다.

오류

- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- ConcurrentModificationException

ListTriggers 작업(Python: list_triggers)

AWS 계정의 모든 트리거 리소스 또는 지정된 태그를 가진 리소스의 이름을 검색합니다. 이 작업을 통해 계정에서 사용 가능한 리소스와 그 이름을 확인할 수 있습니다.

이 작업을 수행하면 응답에서 필터로 사용할 수 있는 선택 사항인 Tags 필드가 검색되기 때문에 태그가 지정된 리소스를 하나의 그룹으로 검색할 수 있습니다. 태그 필터링을 사용하기로 선택하면 태그가 포함된 리소스만 검색됩니다.

요청

- NextToken – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.
- DependentJobName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  트리거를 가져올 작업 이름 이 작업을 시작할 수 있는 트리거가 반환됩니다. 그런 트리거가 없으면 모든 트리거가 반환됩니다.
- MaxResults - 1 이상 200 이하의 숫자(정수)입니다.

  반환할 목록의 최대 크기.
- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

이렇게 태그가 지정된 리소스만 반환하도록 지정합니다.

응답

- `TriggerNames` – UTF-8 문자열의 배열입니다.

  계정의 모든 트리거 또는 지정된 태그를 가진 트리거의 이름입니다.
- `NextToken` – UTF-8 문자열입니다.

  반환된 목록이 사용가능한 마지막 지표를 포함하지 경우의 연속 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchGetTriggers 작업(Python: batch_get_triggers)

주어진 트리거 이름 목록에 대한 리소스 메타데이터 목록을 반환합니다. `ListTriggers` 작업을 호출한 후에는 권한이 부여된 데이터에 액세스하기 위해 이 작업을 호출할 수 있습니다. 이 작업은 태그를 사용하는 권한 조건을 포함해 모든 IAM 권한을 지원합니다.

요청

- `TriggerNames` – 필수(Required): UTF-8 문자열의 배열입니다.

  트리거 이름(`ListTriggers` 작업에서 반환된 이름일 수 있음)의 목록입니다.

응답

- `Triggers` – 트리거 객체의 배열입니다.

  트리거 정의 목록입니다.

- `TriggersNotFound` – UTF-8 문자열의 배열입니다.

  찾을 수 없는 트리거의 이름 목록입니다.

오류

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

AWS Glue의 통합 API

데이터 타입

- 통합 구조
- IntegrationConfig 구조
- IntegrationPartition 구조
- IntegrationError 구조
- IntegrationFilter 구조
- InboundIntegration 구조
- SourceProcessingProperties 구조
- TargetProcessingProperties 구조
- SourceTableConfig 구조
- TargetTableConfig 구조

통합 구조

제로 ETL 통합에 대해 설명합니다.

필드

- `SourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 소스의 ARN입니다.

- `TargetArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 대상의 ARN입니다.

- `Description` – Custom string pattern #12과(와) 일치하는 1,000바이트 이하 길이의 UTF-8 문자열입니다.

  통합에 대한 설명입니다.

- `IntegrationName` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 고유 이름입니다.

- `IntegrationArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 Amazon 리소스 이름(ARN)입니다.

- `KmsKeyId` – UTF-8 문자열입니다(1~2,048바이트).

  채널을 암호화하는 데 사용되는 KMS 키의 ARN입니다.

- `AdditionalEncryptionContext` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  암호화에 대한 추가 컨텍스트 정보가 포함된 비밀이 아닌 선택적 키-값 페어 세트입니다. `KMSKeyId`가 제공된 경우에만 제공할 수 있습니다.

- `Tags` – 태그 객체의 배열입니다.

  키-값 페어 목록으로 구성된 리소스에 할당되는 메타데이터입니다.

- `Status` – 필수: UTF-8 문자열입니다(유효한 값: `CREATING` | `ACTIVE` | `MODIFYING` | `FAILED` | `DELETING` | `SYNCING` | `NEEDS_ATTENTION`).

  가능한 상태는 다음과 같습니다.

  - CREATING: 통합이 생성 중입니다.
  - ACTIVE: 통합 생성이 성공적으로 실행됩니다.
  - MODIFYING: 통합을 수정하는 중입니다.
  - FAILED: 통합 생성이 실패했습니다.
  - DELETING: 통합이 삭제됩니다.
  - SYNCING: 통합이 동기화 중입니다.

  - NEEDS_ATTENTION: 동기화 등으로 인해 통합과 관련하여 주의가 필요합니다.
- `CreateTime` – 필수(Required): 타임스탬프입니다.

  통합을 생성한 시간(UTC)입니다.
- `IntegrationConfig` – IntegrationConfig 객체입니다.

  통합과 관련한 속성입니다.
- `Errors` – IntegrationError 객체의 배열입니다.

  통합과 관련한 오류의 목록입니다.
- `DataFilter` – UTF-8 문자열입니다(1~2,048바이트).

  Maxwell 필터 구문을 사용하여 통합의 소스 테이블을 선택합니다.

IntegrationConfig 구조

통합과 관련한 속성입니다.

필드

- `RefreshInterval` – 1~128바이트 길이의 UTF-8 문자열입니다.

  변경 데이터 캡처(CDC) 풀 또는 증분 로드가 발생하는 빈도를 지정합니다. 이 파라미터는 특정 데이터 업데이트 패턴, 시스템 로드 고려 사항 및 성능 최적화 목표에 맞게 새로 고침 빈도를 조정할 수 있는 유연성을 제공합니다. 시간 증분은 15분~8640분(6일)으로 설정할 수 있습니다. 현재는 `RefreshInterval`의 생성만 지원합니다.
- `SourceProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  통합 소스에 대한 추가 속성을 지정하는 키-값 페어의 컬렉션입니다. 이러한 속성은 데이터 통합 작업 중 ODB 소스의 동작을 사용자 지정하는 데 사용할 수 있는 구성 옵션을 제공합니다.

IntegrationPartition 구조

대상에서 데이터를 분할하는 방법을 설명하는 구조입니다.

필드

- `FieldName` – 1~128바이트 길이의 UTF-8 문자열입니다.

  대상의 데이터를 분할하는 데 사용되는 필드 이름입니다. 'LastModifiedTimestamp' 또는 'SystemModTimeStamp'와 같이 각 행에 고유한 값이 있는 열을 파티션 열로 사용하지 마세요. 이러한 열은 많은 수의 작은 파티션을 생성하여 성능 문제가 발생할 수 있으므로 파티셔닝에 적합하지 않습니다.
- `FunctionSpec` – 1~128바이트 길이의 UTF-8 문자열입니다.

  대상의 데이터를 분할하는 데 사용되는 함수를 지정합니다. 이 파라미터에 대해 허용되는 유일한 값은 ‘identity’(문자열)입니다. 'identity' 함수는 대상의 데이터 파티셔닝이 소스와 같은 체계를 따르도록 합니다. 즉, 소스 데이터의 파티셔닝 구조가 대상에 보존됩니다.
- `ConversionSpec` – 1~128바이트 길이의 UTF-8 문자열입니다.

  소스 데이터의 타임스탬프 형식을 지정합니다. 유효한 값은 다음과 같습니다.
  - `epoch_sec` - 초 단위의 Unix 에포크 타임스탬프
  - `epoch_milli` - 밀리초 단위의 Unix 에포크 타임스탬프
  - `iso` - ISO 8601 형식의 타임스탬프

  > **Note**
  >
  > 타임스탬프 기반 파티션 함수(년, 월, 일 또는 시간)를 사용할 때만 `ConversionSpec`을 지정합니다. AWS Glue 제로 ETL은 파티셔닝 전에 이 파라미터를 사용하여 소스 데이터를 타임스탬프 형식으로 올바르게 변환합니다.
  > `identity` 파티션 함수에는 카디널리티가 높은 열을 사용하지 마세요. 카디널리티가 높은 열은 다음과 같습니다.
  > - 프라이머리 키
  > - 타임스탬프 필드(예: `LastModifiedTimestamp`, `CreatedDate`)
  > - 시스템 생성 타임스탬프
  >
  > ID 파티셔닝과 함께 높은 카디널리티 열을 사용하면 작은 파티션이 많이 생성되어 수집 성능이 크게 저하될 수 있습니다.

IntegrationError 구조

제로 ETL 통합과 관련한 오류입니다.

필드

- ErrorCode – 1~128바이트 길이의 UTF-8 문자열입니다.

  이 오류와 연결된 코드입니다.
- ErrorMessage – UTF-8 문자열입니다(1~2,048바이트).

  메시지에서 오류를 설명합니다.

IntegrationFilter 구조

DescribeIntegrations 요청을 간접적으로 호출할 때 사용할 수 있는 필터입니다.

필드

- Name – 1~128바이트 길이의 UTF-8 문자열입니다.

  필터의 이름.
- Values – UTF-8 문자열의 배열입니다.

  필터 값 목록입니다.

InboundIntegration 구조

리소스에 데이터를 쓰는 통합의 구조입니다.

필드

- SourceArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합을 위한 소스 리소스의 ARN입니다.
- TargetArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합을 위한 대상 리소스의 ARN입니다.
- IntegrationArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  제로 ETL 통합의 ARN입니다.
- Status – 필수: UTF-8 문자열입니다(유효한 값: CREATING | ACTIVE | MODIFYING | FAILED | DELETING | SYNCING | NEEDS_ATTENTION).

가능한 상태는 다음과 같습니다.

- CREATING: 통합이 생성 중입니다.
- ACTIVE: 통합 생성이 성공적으로 실행됩니다.
- MODIFYING: 통합을 수정하는 중입니다.
- FAILED: 통합 생성이 실패했습니다.
- DELETING: 통합이 삭제됩니다.
- SYNCING: 통합이 동기화 중입니다.
- NEEDS_ATTENTION: 동기화 등으로 인해 통합과 관련하여 주의가 필요합니다.

- `CreateTime` – 필수(Required): 타임스탬프입니다.

  통합을 생성한 시간(UTC)입니다.

- `IntegrationConfig` – IntegrationConfig 객체입니다.

  통합과 관련한 속성입니다.

- `Errors` – IntegrationError 객체의 배열입니다.

  통합과 관련한 오류의 목록입니다.

SourceProcessingProperties 구조

통합 소스와 관련한 리소스 속성입니다.

필드

- `RoleArn` – 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 연결에 액세스할 IAM 역할입니다.

TargetProcessingProperties 구조

통합 대상과 관련한 리소스 속성입니다.

필드

- `RoleArn` – 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 데이터베이스에 액세스할 IAM 역할입니다.

- KmsArn – UTF-8 문자열입니다(1~2,048바이트).

  암호화에 사용되는 KMS 키의 ARN입니다.
- ConnectionName – 1~128바이트 길이의 UTF-8 문자열입니다.

  고객 VPC에서 실행 중인 AWS Glue 작업을 구성하기 위한 AWS Glue 네트워크 연결입니다.
- EventBusArn – UTF-8 문자열입니다(1~2,048바이트).

  통합 상태 알림을 수신할 Eventbridge 이벤트 버스의 ARN입니다.

SourceTableConfig 구조

소스 레그에서 소스의 데이터를 처리하는 데 사용하는 속성입니다.

필드

- Fields – UTF-8 문자열의 배열입니다.

  열 수준 필터링에 사용되는 필드 목록입니다. 현재 지원되지 않습니다.
- FilterPredicate – 1~128바이트 길이의 UTF-8 문자열입니다.

  행 수준 필터링에 사용되는 조건 절입니다. 현재 지원되지 않습니다.
- PrimaryKey – UTF-8 문자열의 배열입니다.

  이 테이블의 프라이머리 키 세트를 제공합니다. 현재는 요청 시 SAP EntityOf 엔터티에 대해 지원됩니다. 이 기능을 사용하려면 AWS Support에 문의하세요.
- RecordUpdateField – 1~128바이트 길이의 UTF-8 문자열입니다.

  증분 풀 타임스탬프 기반 필드입니다. 현재 지원되지 않습니다.

TargetTableConfig 구조

대상 레그에서 대상의 데이터를 분할하는 데 사용하는 속성입니다.

필드

- UnnestSpec – UTF-8 문자열입니다(유효한 값: TOPLEVEL | FULL | NOUNNEST).

중첩된 객체를 최상위 요소로 평면화하는 방법을 지정합니다. 유효한 값은 'TOPLEVEL', 'FULL' 또는 'NOUNNEST'입니다.

- `PartitionSpec` – IntegrationPartition 객체의 배열입니다.

  대상의 파일 레이아웃을 결정합니다.

- `TargetTableName` – 1~128바이트 길이의 UTF-8 문자열입니다.

  대상 테이블의 선택적 이름입니다.

운영

- CreateIntegration 작업(Python: create_integration)
- ModifyIntegration 작업(Python: modify_integration)
- DescribeIntegrations 작업(Python: describe_integrations)
- DeleteIntegration 작업(Python: delete_integration)
- DescribeInboundIntegrations 작업(Python: describe_inbound_integrations)
- CreateIntegrationTableProperties 작업(Python: create_integration_table_properties)
- UpdateIntegrationTableProperties 작업(Python: update_integration_table_properties)
- GetIntegrationTableProperties 작업(Python: get_integration_table_properties)
- DeleteIntegrationTableProperties 작업(Python: delete_integration_table_properties)
- CreateIntegrationResourceProperty 작업(Python: create_integration_resource_property)
- UpdateIntegrationResourceProperty 작업(Python: update_integration_resource_property)
- GetIntegrationResourceProperty 작업(Python: get_integration_resource_property)
- UntagResource 작업(Python: untag_resource)
- ListTagsForResource 작업(Python: list_tags_for_resource)

CreateIntegration 작업(Python: create_integration)

Amazon 리소스 이름(ARN)이 있는 두 리소스, 즉 `SourceArn` 및 `TargetArn` 간에 제로 ETL 통합을 호출자 계정에 생성합니다.

요청

- `IntegrationName` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

AWS Glue에서 통합의 고유 이름입니다.

- SourceArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합을 위한 소스 리소스의 ARN입니다.

- TargetArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합을 위한 대상 리소스의 ARN입니다.

- Description – Custom string pattern #12과(와) 일치하는 1,000바이트 이하 길이의 UTF-8 문자열입니다.

  통합에 대한 설명입니다.

- DataFilter – UTF-8 문자열입니다(1~2,048바이트).

  Maxwell 필터 구문을 사용하여 통합의 소스 테이블을 선택합니다.

- KmsKeyId – UTF-8 문자열입니다(1~2,048바이트).

  채널을 암호화하는 데 사용되는 KMS 키의 ARN입니다.

- AdditionalEncryptionContext – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  암호화에 대한 추가 컨텍스트 정보가 포함된 비밀이 아닌 선택적 키-값 페어 세트입니다. KMSKeyId가 제공된 경우에만 제공할 수 있습니다.

- Tags – 태그 객체의 배열입니다.

  키-값 페어 목록으로 구성된 리소스에 할당되는 메타데이터입니다.

- IntegrationConfig – IntegrationConfig 객체입니다.

  구성 설정입니다.

응답

- SourceArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합을 위한 소스 리소스의 ARN입니다.

- TargetArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

통합을 위한 대상 리소스의 ARN입니다.

- `IntegrationName` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue에서 통합의 고유 이름입니다.

- `Description` – Custom string pattern #12과(와) 일치하는 1,000바이트 이하 길이의 UTF-8 문자열입니다.

  통합에 대한 설명입니다.

- `IntegrationArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  생성된 통합의 Amazon 리소스 이름(ARN)입니다.

- `KmsKeyId` – UTF-8 문자열입니다(1~2,048바이트).

  채널을 암호화하는 데 사용되는 KMS 키의 ARN입니다.

- `AdditionalEncryptionContext` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  암호화에 대한 추가 컨텍스트 정보가 포함된 비밀이 아닌 선택적 키-값 페어 세트입니다.

- `Tags` – 태그 객체의 배열입니다.

  키-값 페어 목록으로 구성된 리소스에 할당되는 메타데이터입니다.

- `Status` – 필수: UTF-8 문자열입니다(유효한 값: `CREATING` | `ACTIVE` | `MODIFYING` | `FAILED` | `DELETING` | `SYNCING` | `NEEDS_ATTENTION`).

  생성 중인 통합의 상태입니다.

  가능한 상태는 다음과 같습니다.

  - CREATING: 통합이 생성 중입니다.
  - ACTIVE: 통합 생성이 성공적으로 실행됩니다.
  - MODIFYING: 통합을 수정하는 중입니다.
  - FAILED: 통합 생성이 실패했습니다.
  - DELETING: 통합이 삭제됩니다.
  - SYNCING: 통합이 동기화 중입니다.

  - NEEDS_ATTENTION: 동기화 등으로 인해 통합과 관련하여 주의가 필요합니다.
- `CreateTime` – 필수(Required): 타임스탬프입니다.

  통합이 생성된 시점의 시각(UTC)입니다.
- `Errors` – IntegrationError 객체의 배열입니다.

  통합 생성과 관련된 오류의 목록입니다.
- `DataFilter` – UTF-8 문자열입니다(1~2,048바이트).

  Maxwell 필터 구문을 사용하여 통합의 소스 테이블을 선택합니다.
- `IntegrationConfig` – IntegrationConfig 객체입니다.

  구성 설정입니다.

오류

- `ValidationException`
- `AccessDeniedException`
- `ResourceNotFoundException`
- `InternalServerException`
- `IntegrationConflictOperationFault`
- `IntegrationQuotaExceededFault`
- `KMSKeyNotAccessibleFault`
- `EntityNotFoundException`
- `InternalServiceException`
- `ConflictException`
- `ResourceNumberLimitExceededException`
- `InvalidInputException`

ModifyIntegration 작업(Python: modify_integration)

호출자의 계정에서 제로 ETL 통합을 수정합니다.

요청

- `IntegrationIdentifier` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 Amazon 리소스 이름(ARN)입니다.

- `Description` – Custom string pattern #12과(와) 일치하는 1,000바이트 이하 길이의 UTF-8 문자열입니다.

  통합에 대한 설명입니다.

- `DataFilter` – UTF-8 문자열입니다(1~2,048바이트).

  Maxwell 필터 구문을 사용하여 통합의 소스 테이블을 선택합니다.

- `IntegrationName` – 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue에서 통합의 고유 이름입니다.

응답

- `SourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 소스의 ARN입니다.

- `TargetArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 대상의 ARN입니다.

- `IntegrationName` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue에서 통합의 고유 이름입니다.

- `Description` – Custom string pattern #12과(와) 일치하는 1,000바이트 이하 길이의 UTF-8 문자열입니다.

  통합에 대한 설명입니다.

- `IntegrationArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 Amazon 리소스 이름(ARN)입니다.

- `KmsKeyId` – UTF-8 문자열입니다(1~2,048바이트).

  채널을 암호화하는 데 사용되는 KMS 키의 ARN입니다.

- `AdditionalEncryptionContext` – 키-값 페어의 맵 배열입니다.

각 키는 UTF-8 문자열입니다.

각 값은 UTF-8 문자열입니다.

암호화에 대한 추가 컨텍스트 정보가 포함된 비밀이 아닌 선택적 키-값 페어 세트입니다.

- `Tags` – 태그 객체의 배열입니다.

  키-값 페어 목록으로 구성된 리소스에 할당되는 메타데이터입니다.

- `Status` – 필수: UTF-8 문자열입니다(유효한 값: `CREATING` | `ACTIVE` | `MODIFYING` | `FAILED` | `DELETING` | `SYNCING` | `NEEDS_ATTENTION`).

  수정 중인 통합의 상태입니다.

  가능한 상태는 다음과 같습니다.
  - CREATING: 통합이 생성 중입니다.
  - ACTIVE: 통합 생성이 성공적으로 실행됩니다.
  - MODIFYING: 통합을 수정하는 중입니다.
  - FAILED: 통합 생성이 실패했습니다.
  - DELETING: 통합이 삭제됩니다.
  - SYNCING: 통합이 동기화 중입니다.
  - NEEDS_ATTENTION: 동기화 등으로 인해 통합과 관련하여 주의가 필요합니다.

- `CreateTime` – 필수(Required): 타임스탬프입니다.

  통합이 생성된 시점의 시각(UTC)입니다.

- `Errors` – IntegrationError 객체의 배열입니다.

  통합 수정과 관련된 오류의 목록입니다.

- `DataFilter` – UTF-8 문자열입니다(1~2,048바이트).

  Maxwell 필터 구문을 사용하여 통합의 소스 테이블을 선택합니다.

오류

- `ValidationException`
- `AccessDeniedException`
- `InternalServerException`

- IntegrationNotFoundFault
- IntegrationConflictOperationFault
- InvalidIntegrationStateFault
- EntityNotFoundException
- InternalServiceException
- ConflictException
- InvalidStateException
- InvalidInputException

DescribeIntegrations 작업(Python: describe_integrations)

이 API는 통합의 목록을 검색하는 데 사용됩니다.

요청

- IntegrationIdentifier – 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 Amazon 리소스 이름(ARN)입니다.
- Marker – 1~128바이트 길이의 UTF-8 문자열입니다.

  후속 요청에서 다음 응답 레코드 세트의 시작점을 나타내는 값입니다.
- MaxRecords - 숫자(정수)입니다.

  출력에서 반환되는 항목의 총 수입니다.
- Filters – IntegrationFilter 객체의 배열입니다.

  결과를 필터링하기 위한 키와 값의 목록입니다. 지원되는 키는 'Status', 'IntegrationName' 및 'SourceArn'입니다. IntegrationName은 하나의 값으로만 제한됩니다.

응답

- Integrations – 통합 객체의 배열입니다.

  제로 ETL 통합의 목록입니다.
- Marker – 1~128바이트 길이의 UTF-8 문자열입니다.

후속 요청에서 다음 응답 레코드 세트의 시작점을 나타내는 값입니다.

오류

- `ValidationException`
- `AccessDeniedException`
- `InternalServerException`
- `IntegrationNotFoundFault`
- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`

DeleteIntegration 작업(Python: delete_integration)

지정된 제로 ETL 통합을 삭제합니다.

요청

- `IntegrationIdentifier` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 Amazon 리소스 이름(ARN)입니다.

응답

- `SourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 소스의 ARN입니다.
- `TargetArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 대상의 ARN입니다.
- `IntegrationName` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  AWS Glue에서 통합의 고유 이름입니다.
- `Description` – Custom string pattern #12과(와) 일치하는 1,000바이트 이하 길이의 UTF-8 문자열입니다.

통합에 대한 설명입니다.

- `IntegrationArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 Amazon 리소스 이름(ARN)입니다.

- `KmsKeyId` – UTF-8 문자열입니다(1~2,048바이트).

  채널을 암호화하는 데 사용되는 KMS 키의 ARN입니다.

- `AdditionalEncryptionContext` – 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  암호화에 대한 추가 컨텍스트 정보가 포함된 비밀이 아닌 선택적 키-값 페어 세트입니다.

- `Tags` – 태그 객체의 배열입니다.

  키-값 페어 목록으로 구성된 리소스에 할당되는 메타데이터입니다.

- `Status` – 필수: UTF-8 문자열입니다(유효한 값: `CREATING` | `ACTIVE` | `MODIFYING` | `FAILED` | `DELETING` | `SYNCING` | `NEEDS_ATTENTION`).

  삭제 중인 통합의 상태입니다.

  가능한 상태는 다음과 같습니다.

  - CREATING: 통합이 생성 중입니다.
  - ACTIVE: 통합 생성이 성공적으로 실행됩니다.
  - MODIFYING: 통합을 수정하는 중입니다.
  - FAILED: 통합 생성이 실패했습니다.
  - DELETING: 통합이 삭제됩니다.
  - SYNCING: 통합이 동기화 중입니다.
  - NEEDS_ATTENTION: 동기화 등으로 인해 통합과 관련하여 주의가 필요합니다.

- `CreateTime` – 필수(Required): 타임스탬프입니다.

  통합이 생성된 시점의 시각(UTC)입니다.

- `Errors` – IntegrationError 객체의 배열입니다.

  통합과 관련한 오류의 목록입니다.

- DataFilter – UTF-8 문자열입니다(1~2,048바이트).

  Maxwell 필터 구문을 사용하여 통합의 소스 테이블을 선택합니다.

오류

- ValidationException
- AccessDeniedException
- InternalServerException
- IntegrationNotFoundFault
- IntegrationConflictOperationFault
- InvalidIntegrationStateFault
- EntityNotFoundException
- InternalServiceException
- ConflictException
- InvalidStateException
- InvalidInputException

DescribeInboundIntegrations 작업(Python: describe_inbound_integrations)

지정된 통합의 인바운드 통합 목록을 반환합니다.

요청

- IntegrationArn – 1~128바이트 길이의 UTF-8 문자열입니다.

  통합의 Amazon 리소스 이름(ARN)입니다.
- Marker – 1~128바이트 길이의 UTF-8 문자열입니다.

  페이지 매김을 시작할 위치를 지정하기 위한 토큰입니다. 이는 이전에 잘린 응답에서 도출된 마커입니다.
- MaxRecords - 숫자(정수)입니다.

  출력에서 반환되는 항목의 총 수입니다.
- TargetArn – 1~128바이트 길이의 UTF-8 문자열입니다.

통합에 포함된 대상 리소스의 Amazon 리소스 이름(ARN)입니다.

응답

- `InboundIntegrations` – InboundIntegration 객체의 배열입니다.

  인바운드 통합의 목록입니다.
- `Marker` – 1~128바이트 길이의 UTF-8 문자열입니다.

  후속 요청에서 다음 응답 레코드 세트의 시작점을 나타내는 값입니다.

오류

- `ValidationException`
- `AccessDeniedException`
- `InternalServerException`
- `IntegrationNotFoundFault`
- `TargetResourceNotFound`
- `OperationNotSupportedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`

CreateIntegrationTableProperties 작업(Python: create_integration_table_properties)

이 API는 복제할 테이블에 대한 선택적 재정의 속성을 제공하는 데 사용됩니다. 이러한 속성에는 소스 및 대상 테이블에 대한 필터링 및 파티셔닝 속성이 포함될 수 있습니다. 소스 속성과 대상 속성을 모두 설정하려면 각각 `SourceTableConfig`에서 `ResourceArn`을 AWS Glue 연결 ARN으로 사용하고, `TargetTableConfig`에서 `ResourceArn`을 AWS Glue 데이터베이스 ARN으로 사용하여 동일한 API를 간접적으로 호출해야 합니다.

요청

- `ResourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

통합 테이블 속성을 생성할 대상 테이블의 Amazon 리소스 이름(ARN)입니다. 현재 이 API는 대상 테이블에 대한 통합 테이블 속성 생성만 지원하며 제공된 ARN은 AWS Glue 데이터 카탈로그에 있는 대상 테이블의 ARN이어야 합니다. 소스 연결(연결 ARN 사용)에 대한 통합 테이블 속성 생성 기능 지원은 아직 구현되지 않았으며 향후 릴리스에서 추가될 예정입니다.

- `TableName` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  복제할 테이블의 이름입니다.

- `SourceTableConfig` – SourceTableConfig 객체입니다.

  소스 테이블 구성의 구조입니다. 지원되는 소스 속성 목록을 보려면 `SourceTableConfig` 구조를 참조하세요.

- `TargetTableConfig` – TargetTableConfig 객체입니다.

  대상 테이블 구성의 구조입니다.

응답

- 무응답 파라미터.

오류

- `ValidationException`
- `AccessDeniedException`
- `ResourceNotFoundException`
- `InternalServerException`
- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`

UpdateIntegrationTableProperties 작업(Python: update_integration_table_properties)

이 API는 복제할 테이블에 대한 선택적 재정의 속성을 제공하는 데 사용됩니다. 이러한 속성에는 소스 및 대상 테이블에 대한 필터링 및 파티셔닝 속성이 포함될 수 있습니다. 소스 속성과 대상 속성을 모

두 설정하려면 각각 SourceTableConfig에서 ResourceArn을 AWS Glue 연결 ARN으로 사용하고, TargetTableConfig에서 ResourceArn을 AWS Glue 데이터베이스 ARN으로 사용하여 동일한 API를 간접적으로 호출해야 합니다.

재정의는 동일한 ResourceArn 및 소스 테이블을 사용하여 모든 통합에 반영됩니다.

요청

- ResourceArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.
- TableName – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  복제할 테이블의 이름입니다.
- SourceTableConfig – SourceTableConfig 객체입니다.

  소스 테이블 구성의 구조입니다.
- TargetTableConfig – TargetTableConfig 객체입니다.

  대상 테이블 구성의 구조입니다.

응답

- 무응답 파라미터.

오류

- ValidationException
- AccessDeniedException
- ResourceNotFoundException
- InternalServerException
- EntityNotFoundException
- InternalServiceException
- InvalidInputException

GetIntegrationTableProperties 작업(Python: get_integration_table_properties)

이 API는 복제할 테이블에 대한 선택적 재정의 속성을 검색하는 데 사용됩니다. 이러한 속성에는 소스 및 대상 테이블에 대한 필터링 및 파티션 속성이 포함될 수 있습니다.

요청

- `ResourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 테이블 속성을 검색할 대상 테이블의 Amazon 리소스 이름(ARN)입니다. 현재 이 API는 대상 테이블에 대한 속성 검색만 지원하며 제공된 ARN은 AWS Glue 데이터 카탈로그에 있는 대상 테이블의 ARN이어야 합니다. 소스 연결(연결 ARN 사용)에 대한 통합 테이블 속성 검색 기능 지원은 아직 구현되지 않았으며 향후 릴리스에서 추가될 예정입니다.

- `TableName` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  복제할 테이블의 이름입니다.

응답

- `ResourceArn` – 1~128바이트 길이의 UTF-8 문자열입니다.

  통합 테이블 속성을 검색할 대상 테이블의 Amazon 리소스 이름(ARN)입니다. 현재 이 API는 대상 테이블에 대한 속성 검색만 지원하며 제공된 ARN은 AWS Glue 데이터 카탈로그에 있는 대상 테이블의 ARN이어야 합니다. 소스 연결(연결 ARN 사용)에 대한 통합 테이블 속성 검색 기능 지원은 아직 구현되지 않았으며 향후 릴리스에서 추가될 예정입니다.

- `TableName` – 1~128바이트 길이의 UTF-8 문자열입니다.

  복제할 테이블의 이름입니다.

- `SourceTableConfig` – SourceTableConfig 객체입니다.

  소스 테이블 구성의 구조입니다.

- `TargetTableConfig` – TargetTableConfig 객체입니다.

  대상 테이블 구성의 구조입니다.

오류

- ValidationException
- AccessDeniedException
- ResourceNotFoundException
- InternalServerException
- EntityNotFoundException
- InternalServiceException
- InvalidInputException

DeleteIntegrationTableProperties 작업(Python: delete_integration_table_properties)

복제할 테이블에 대해 생성된 테이블 속성을 삭제합니다.

요청

- ResourceArn – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.
- TableName – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  복제할 테이블의 이름입니다.

응답

- 무응답 파라미터.

오류

- ValidationException
- AccessDeniedException
- ResourceNotFoundException
- InternalServerException
- EntityNotFoundException

- `InternalServiceException`
- `InvalidInputException`

CreateIntegrationResourceProperty 작업(Python: create_integration_resource_property)

이 API는 AWS Glue 연결(소스의 경우)의 `ResourceProperty` 또는 AWS Glue 데이터베이스 ARN(대상의 경우)을 설정하는 데 사용할 수 있습니다. 이러한 속성에는 연결 또는 데이터베이스에 액세스하는 역할이 포함될 수 있습니다. 소스 속성과 대상 속성을 모두 설정하려면 각각 `SourceProcessingProperties`에서 `ResourceArn`을 AWS Glue 연결 ARN으로 사용하고, `TargetProcessingProperties`에서 `ResourceArn`을 AWS Glue 데이터베이스 ARN으로 사용하여 동일한 API를 간접적으로 호출해야 합니다.

요청

- `ResourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.
- `SourceProcessingProperties` – SourceProcessingProperties 객체입니다.

  통합 소스와 관련한 리소스 속성입니다.
- `TargetProcessingProperties` – TargetProcessingProperties 객체입니다.

  통합 대상과 관련한 리소스 속성입니다.

응답

- `ResourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.
- `SourceProcessingProperties` – SourceProcessingProperties 객체입니다.

  통합 소스와 관련한 리소스 속성입니다.
- `TargetProcessingProperties` – TargetProcessingProperties 객체입니다.

  통합 대상과 관련한 리소스 속성입니다.

오류

- `ValidationException`
- `AccessDeniedException`
- `ConflictException`
- `InternalServerException`
- `ResourceNotFoundException`
- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`

UpdateIntegrationResourceProperty 작업(Python: update_integration_resource_property)

이 API는 AWS Glue 연결(소스의 경우)의 `ResourceProperty` 또는 AWS Glue 데이터베이스 ARN(대상의 경우)을 업데이트하는 데 사용할 수 있습니다. 이러한 속성에는 연결 또는 데이터베이스에 액세스하는 역할이 포함될 수 있습니다. 동일한 리소스를 여러 통합에서 사용할 수 있으므로 리소스 속성을 업데이트하면 이를 사용하는 모든 통합에 영향을 미칩니다.

요청

- `ResourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.
- `SourceProcessingProperties` – SourceProcessingProperties 객체입니다.

  통합 소스와 관련한 리소스 속성입니다.
- `TargetProcessingProperties` – TargetProcessingProperties 객체입니다.

  통합 대상과 관련한 리소스 속성입니다.

응답

- `ResourceArn` – 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.

- `SourceProcessingProperties` – SourceProcessingProperties 객체입니다.

  통합 소스와 관련한 리소스 속성입니다.
- `TargetProcessingProperties` – TargetProcessingProperties 객체입니다.

  통합 대상과 관련한 리소스 속성입니다.

오류

- `ValidationException`
- `AccessDeniedException`
- `InternalServerException`
- `ResourceNotFoundException`
- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`

GetIntegrationResourceProperty 작업(Python: get_integration_resource_property)

이 API는 AWS Glue 연결(소스의 경우) 또는 AWS Glue 데이터베이스 ARN(대상의 경우)의 `ResourceProperty`를 가져오는 데 사용됩니다.

요청

- `ResourceArn` – 필수: 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.

응답

- `ResourceArn` – 1~128바이트 길이의 UTF-8 문자열입니다.

  소스의 연결 ARN 또는 대상의 데이터베이스 ARN입니다.
- `SourceProcessingProperties` – SourceProcessingProperties 객체입니다.

  통합 소스와 관련한 리소스 속성입니다.

- `TargetProcessingProperties` – TargetProcessingProperties 객체입니다.

  통합 대상과 관련한 리소스 속성입니다.

오류

- `ValidationException`
- `AccessDeniedException`
- `InternalServerException`
- `ResourceNotFoundException`
- `EntityNotFoundException`
- `InternalServiceException`
- `InvalidInputException`

UntagResource 작업(Python: untag_resource)

통합 리소스에서 지정된 태그를 제거합니다.

요청

- `ResourceArn` – 필수(Required): Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  통합 리소스의 Amazon 리소스 이름(ARN)입니다.
- `TagsToRemove` – 필수(Required): 50개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  리소스에서 제거할 메타데이터 태그의 목록입니다.

응답

- 무응답 파라미터.

오류

- `ResourceNotFoundException`

ListTagsForResource 작업(Python: list_tags_for_resource)

지정된 리소스에 할당한 메타데이터 태그를 나열합니다.

요청

- `ResourceARN` – 필수(Required): Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  리소스의 리소스 ARN입니다.

응답

- `Tags` – 태그 객체의 배열이며 구조는 10개 이하입니다.

  태그의 목록입니다.

오류

- `ResourceNotFoundException`

예외

- ResourceNotFoundException 구조
- InternalServerException 구조
- IntegrationAlreadyExistsFault 구조
- IntegrationConflictOperationFault 구조
- IntegrationQuotaExceededFault 구조
- KMSKeyNotAccessibleFault 구조
- IntegrationNotFoundFault 구조
- TargetResourceNotFound 구조
- InvalidIntegrationStateFault 구조

ResourceNotFoundException 구조

리소스를 찾을 수 없습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

InternalServerException 구조

내부 서버 오류가 발생했습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

IntegrationAlreadyExistsFault 구조

지정된 통합이 이미 있습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

IntegrationConflictOperationFault 구조

요청된 작업이 다른 작업과 충돌합니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

IntegrationQuotaExceededFault 구조

통합을 통해 처리된 데이터가 할당량을 초과했습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

KMSKeyNotAccessibleFault 구조

지정된 KMS 키에 액세스할 수 없습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

IntegrationNotFoundFault 구조

지정된 통합을 찾을 수 없습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

TargetResourceNotFound 구조

대상 리소스를 찾을 수 없습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

InvalidIntegrationStateFault 구조

통합 상태가 잘못되었습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

대화형 세션 API

대화형 세션 API는 AWS Glue 대화형 세션을 사용하여 데이터 통합을 위한 추출, 변환, 로드 스크립트를 빌드하고 테스트하는 것과 관련된 AWS Glue API를 설명합니다.

데이터 타입

- 세션 구조
- SessionCommand 구조
- 명령문 구조
- StatementOutput 구조
- StatementOutputData 구조
- ConnectionsList 구조

세션 구조

원격 Spark 런타임 환경이 실행되는 기간입니다.

필드

- `Id` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  세션의 ID입니다.

- `CreatedOn` – 타임스탬프입니다.

  세션이 생성된 시간 및 날짜입니다.

- `Status` – UTF-8 문자열입니다(유효 값: `PROVISIONING` | `READY` | `FAILED` | `TIMEOUT` | `STOPPING` | `STOPPED`).

  세션 상태입니다.

- `ErrorMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  세션 중 표시되는 오류 메시지입니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  세션에 대한 설명입니다.

- `Role` – Custom string pattern #30과 일치하는 UTF-8 문자열입니다(20~2,048바이트).

  세션과 연결된 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다.

- `Command` – SessionCommand 객체입니다.

  명령 객체입니다. SessionCommand를 참조하세요.

- `DefaultArguments` – 75개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  각 값은 URI address multi-line string pattern와 일치하는 UTF-8 문자열(4,096바이트 이하)입니다.

  키-값 페어의 맵 배열입니다. 최대 75페어입니다.

- `Connections` – ConnectionsList 객체입니다.

  세션에 사용되는 연결 수입니다.

- `Progress` - 숫자(double)입니다.

  세션의 코드 실행 진행률입니다.

- `MaxCapacity` - 숫자(double)입니다.

  작업이 실행될 때 할당할 수 있는 AWS Glue 데이터 처리 단위(DPU) 수입니다. DPU는 4 vCPU의 컴퓨팅 용량과 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  세션과 함께 사용할 SecurityConfiguration 구조의 이름입니다.

- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

AWS Glue 버전은 AWS Glue가 지원하는 Apache Spark 및 Python 버전을 결정합니다. GlueVersion은 2.0보다 커야 합니다.

- `DataAccessId` – UTF-8 문자열입니다(1~36바이트 이하)

  세션의 데이터 액세스 ID입니다.

- `PartitionId` – UTF-8 문자열입니다(1~36바이트 이하)

  세션의 파티션 ID입니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  세션에 사용할 정의된 `WorkerType`의 작업자 수입니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  세션이 실행될 때 할당되는 미리 정의된 작업자 유형입니다. Spark 세션에 대해 `G.1X`, `G.2X`, `G.4X` 또는 `G.8X`의 값을 허용합니다. Ray 세션에 대해 `Z.2X`의 값을 허용합니다.

- `CompletedOn` – 타임스탬프입니다.

  이 세션이 완료된 날짜 및 시간입니다.

- `ExecutionTime` - 숫자(double)입니다.

  세션이 실행된 총 시간입니다.

- `DPUSeconds` - 숫자(double)입니다.

  세션에서 소비된 DPU입니다(공식: ExecutionTime * MaxCapacity).

- `IdleTimeout` - 숫자(정수)입니다.

  세션 시간이 초과되기까지 유휴 상태의 시간(분)입니다.

- `ProfileName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  세션과 연결된 AWS Glue 사용 프로필의 이름입니다.

SessionCommand 구조

작업을 실행하는 `SessionCommand`입니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  SessionCommand의 이름을 지정합니다. 'glueetl' 또는 'gluestreaming'일 수 있습니다.
- `PythonVersion` – Custom string pattern #48과(와) 일치하는 UTF-8 문자열입니다.

  Python 버전을 지정합니다. Python의 버전으로 Spark 유형의 작업에 대해 지원되는 버전을 확인할 수 있습니다.

명령문 구조

세션에서 발생할 특정 작업에 대한 명령문 또는 요청입니다.

필드

- `Id` - 숫자(정수)입니다.

  문의 ID입니다.
- `Code` – UTF-8 문자열입니다.

  문의 실행 코드입니다.
- `State` – UTF-8 문자열입니다(유효 값: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

  요청이 실행되는 동안의 상태입니다.
- `Output` – StatementOutput 객체입니다.

  JSON의 출력입니다.
- `Progress` - 숫자(double)입니다.

  코드 실행 진행률입니다.
- `StartedOn` - 숫자(정수)입니다.

  작업 정의가 시작된 Unix 시간 및 날짜입니다.
- `CompletedOn` - 숫자(정수)입니다.

  작업 정의가 완료된 Unix 시간 및 날짜입니다.

StatementOutput 구조

JSON 형식의 코드 실행 출력입니다.

필드

- `Data` – StatementOutputData 객체입니다.

  코드 실행 출력입니다.
- `ExecutionCount` - 숫자(정수)입니다.

  출력의 실행 수입니다.
- `Status` – UTF-8 문자열입니다(유효 값: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

  코드 실행 출력의 상태입니다.
- `ErrorName` – UTF-8 문자열입니다.

  출력의 오류 이름입니다.
- `ErrorValue` – UTF-8 문자열입니다.

  출력의 오류 값입니다.
- `Traceback` – UTF-8 문자열의 배열입니다.

  출력의 트레이스백입니다.

StatementOutputData 구조

JSON 형식의 코드 실행 출력입니다.

필드

- `TextPlain` – UTF-8 문자열입니다.

  텍스트 형식의 코드 실행 출력입니다.

ConnectionsList 구조

작업이 사용한 연결을 지정합니다.

필드

- `Connections` – 1,000개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  작업이 사용한 연결 목록입니다.

운영

- CreateSession 작업(Python: create_session)
- StopSession 작업(Python: stop_session)
- DeleteSession 작업(Python: delete_session)
- GetSession 작업(Python: get_session)
- ListSessions 작업(Python: list_sessions)
- RunStatement 작업(Python: run_statement)
- CancelStatement 작업(Python: cancel_statement)
- GetStatement 작업(Python: get_statement)
- ListStatements 작업(Python: list_statements)

CreateSession 작업(Python: create_session)

새 세션을 생성합니다.

요청

새 세션 생성을 요청합니다.

- `Id` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  세션 요청의 ID입니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  세션에 대한 설명입니다.

- `Role` – 필수: Custom string pattern #30과 일치하는 UTF-8 문자열입니다(20~2,048바이트).

IAM 역할 ARN

- `Command` – 필수(Required): SessionCommand 객체입니다.

  작업을 실행하는 `SessionCommand`입니다.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  세션 시간이 초과되기 전 경과되는 시간(분)입니다. Spark ETL 작업의 기본값은 48시간(2,880분)입니다. 다른 작업 유형에 대해서는 설명서를 참조하세요.

- `IdleTimeout` – 1 이상의 숫자(정수)입니다.

  세션 시간이 초과되기까지 유휴 상태의 시간(초)입니다. Spark ETL 작업의 기본값은 시간 제한 값입니다. 다른 작업 유형에 대해서는 설명서를 참조하세요.

- `DefaultArguments` – 75개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  각 값은 URI address multi-line string pattern와 일치하는 UTF-8 문자열(4,096바이트 이하)입니다.

  키-값 페어의 맵 배열입니다. 최대 75페어입니다.

- `Connections` – ConnectionsList 객체입니다.

  세션에 사용할 연결 수입니다.

- `MaxCapacity` - 숫자(double)입니다.

  작업이 실행될 때 할당할 수 있는 AWS Glue 데이터 처리 단위(DPU) 수입니다. DPU는 4 vCPU의 컴퓨팅 용량과 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  세션에 사용할 정의된 `WorkerType`의 작업자 수입니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  작업이 실행될 때 할당되는 미리 정의된 작업자 유형입니다. Spark 작업에 대해 G.1X, G.2X, G.4X 또는 G.8X의 값을 허용합니다. Ray 노트북에 대해 Z.2X 값을 허용합니다.

  - `G.1X` 작업자 유형의 경우, 각 작업자가 94GB의 디스크가 있는 1DPU(4개의 vCPU, 16GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고

비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.

- `G.2X` 작업자 유형의 경우, 각 작업자가 138GB의 디스크가 있는 2DPU(8개의 vCPU, 32GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 대부분의 작업을 실행할 수 있는 확장 가능하고 비용 효율적인 방법을 제공하기 위해 데이터 변환, 조인, 쿼리와 같은 워크로드에서 이 작업자 유형을 사용하는 것이 좋습니다.
- `G.4X` 작업자 유형의 경우, 각 작업자가 256GB의 디스크가 있는 4DPU(16개의 vCPU, 64GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(스톡홀름)과 같은AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
- `G.8X` 작업자 유형의 경우, 각 작업자가 512GB의 디스크가 있는 8DPU(32개의 vCPU, 128GB 메모리)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에서 이 작업자 유형을 사용하는 것이 좋습니다. 이 작업자 유형은 `G.4X` 작업자 유형에 지원되는 동일한 AWS 리전에서 AWS Glue 버전 3.0 이상 Spark ETL 작업에 대해서만 사용할 수 있습니다.
- `Z.2X` 작업자 유형의 경우, 각 작업자는 128GB 디스크에서 2개의 M-DPU(vCPU 8개, 메모리 64GB)에 매핑되고, Autoscaler에 따라 최대 8개의 Ray 작업자를 제공합니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  세션과 함께 사용할 SecurityConfiguration 구조의 이름입니다.

- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 버전은 AWS Glue가 지원하는 Apache Spark 및 Python 버전을 결정합니다. GlueVersion은 2.0보다 커야 합니다.

- `DataAccessId` – UTF-8 문자열입니다(1~36바이트 이하)

  세션의 데이터 액세스 ID입니다.

- `PartitionId` – UTF-8 문자열입니다(1~36바이트 이하)

  세션의 파티션 ID입니다.

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

세션에 속하는 키 값 페어(태그)의 맵입니다.

- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  요청의 오리진입니다.

- `ProfileName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  세션과 연결된 AWS Glue 사용 프로필의 이름입니다.

응답

- `Session` – 세션 객체입니다.

  응답으로 세션 객체를 반환합니다.

오류

- `AccessDeniedException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`

StopSession 작업(Python: stop_session)

세션을 중지합니다.

요청

- `Id` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  중지할 세션의 ID입니다.

- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  요청의 오리진입니다.

응답

- `Id` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  중지된 세션의 ID를 반환합니다.

오류

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`
- `ConcurrentModificationException`

DeleteSession 작업(Python: delete_session)

세션을 삭제합니다.

요청

- `Id` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제되는 세션의 ID입니다.

- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  세션 삭제 요청의 오리진 이름입니다.

응답

- `Id` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제된 세션의 ID를 반환합니다.

오류

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`
- `ConcurrentModificationException`

GetSession 작업(Python: get_session)

세션을 검색합니다.

요청

- `Id` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  세션의 ID입니다.

- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  요청의 오리진입니다.

응답

- `Session` – 세션 객체입니다.

  세션 객체가 응답으로 반환됩니다.

오류

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

ListSessions 작업(Python: list_sessions)

세션 목록을 검색합니다.

요청

- `NextToken` – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  다음 결과 세트를 가져오기 위한 토큰이지만 결과가 더 없는 경우에는 null 값을 갖습니다.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  최대 결과 수입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  세션에 속한 태그입니다.
- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  요청의 오리진입니다.

응답

- `Ids` – UTF-8 문자열의 배열입니다.

  세션의 ID를 반환합니다.

- `Sessions` – 세션 객체의 배열입니다.

  세션 객체를 반환합니다.

- `NextToken` – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  다음 결과 세트를 가져오기 위한 토큰이지만 결과가 더 없는 경우에는 null 값을 갖습니다.

오류

- `AccessDeniedException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

RunStatement 작업(Python: run_statement)

문을 실행합니다.

요청

- `SessionId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행할 문의 세션 ID입니다.

- `Code` – 필수: UTF-8 문자열입니다(68,000바이트 이하).

  실행할 문 코드입니다.

- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  요청의 오리진입니다.

응답

- `Id` - 숫자(정수)입니다.

  실행된 문의 ID를 반환합니다.

오류

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`
- `IllegalSessionStateException`

CancelStatement 작업(Python: cancel_statement)

문을 취소합니다.

요청

- `SessionId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  취소할 문의 세션 ID입니다.

- `Id` – 필수(Required): 숫자(정수)입니다.

  취소할 문의 ID입니다.

- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  문 취소 요청의 오리진입니다.

응답

- 무응답 파라미터.

오류

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

GetStatement 작업(Python: get_statement)

문을 검색합니다.

요청

- `SessionId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문의 세션 ID입니다.

- `Id` – 필수(Required): 숫자(정수)입니다.

  문의 ID입니다.

- `RequestOrigin` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  요청의 오리진입니다.

응답

- `Statement` – 문 객체입니다.

  문을 반환합니다.

오류

- AccessDeniedException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- InvalidInputException
- IllegalSessionStateException

ListStatements 작업(Python: list_statements)

세션에 대한 문을 나열합니다.

요청

- SessionId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문의 세션 ID입니다.

- RequestOrigin – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  문 나열 요청의 오리진입니다.

- NextToken – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- Statements – 문 객체의 배열입니다.

  문 목록을 반환합니다.

- NextToken – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  모든 문이 아직 반환되지 않은 경우의 지속 토큰입니다.

오류

- AccessDeniedException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- InvalidInputException
- IllegalSessionStateException

개발 엔드포인트 API

개발 엔드포인트 API는 사용자 지정 DevEndpoint를 사용한 테스트와 관련된 AWS Glue API에 대해 설명합니다.

데이터 타입

- DevEndpoint 구조
- DevEndpointCustomLibraries 구조

DevEndpoint 구조

개발자가 원격으로 ETL(추출, 변환 및 로드) 스크립트를 디버그할 수 있는 개발 엔드포인트입니다.

필드

- EndpointName – UTF-8 문자열입니다.

  DevEndpoint의 이름입니다.
- RoleArn – AWS IAM ARN string pattern과(와) 일치하는 UTF-8 문자열입니다.

  이 DevEndpoint에 사용되는 IAM 역할의 Amazon 리소스 이름(ARN)입니다.
- SecurityGroupIds – UTF-8 문자열의 배열입니다.

  이 DevEndpoint에 사용된 보안 그룹 식별자 목록입니다.
- SubnetId – UTF-8 문자열입니다.

이 DevEndpoint에 대한 서브넷 ID입니다.

- YarnEndpointAddress – UTF-8 문자열입니다.

  이 DevEndpoint가 사용하는 YARN 엔드포인트 주소입니다.

- PrivateAddress – UTF-8 문자열입니다.

  DevEndpoint가 하나의 VPC에 생성되었다면 VPC 내 DevEndpoint에 액세스할 수 있는 프라이빗 IP 주소입니다. PrivateAddress 필드는 VPC 내에 DevEndpoint를 생성할 때만 표시됩니다.

- ZeppelinRemoteSparkInterpreterPort - 숫자(정수)입니다.

  원격 Apache Spark 인터프리터용 Apache Zeppelin 포트입니다.

- PublicAddress – UTF-8 문자열입니다.

  이 DevEndpoint가 사용하는 퍼블릭 IP 주소입니다. PublicAddress 필드는 비Virtual Private Cloud(VPC) DevEndpoint를 생성할 때만 표시됩니다.

- Status – UTF-8 문자열입니다.

  이 DevEndpoint의 현재 상태입니다.

- WorkerType – UTF-8 문자열입니다(유효한 값: Standard="" | G.1X="" | G.2X="" | G.025X="" | G.4X="" | G.8X="" | Z.2X="").

  개발 엔드포인트로 할당되는 미리 정의된 작업자 유형입니다. Standard, G.1X 또는 G.2X 값을 허용합니다.

  - Standard 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 50GB 디스크와, 작업자당 실행기 2개를 제공합니다.
  - G.1X 작업자 유형의 경우, 각 작업자가 1DPU(4vCPU, 16GB 메모리, 64GB 디스크)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 메모리 집약적인 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다.
  - G.2X 작업자 유형의 경우, 각 작업자가 2DPU(8vCPU, 32GB 메모리, 128GB 디스크)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 메모리 집약적인 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다.

  알려진 문제: 개발 엔드포인트가 G.2X WorkerType 구성으로 만들어질 경우 개발 엔드포인트의 Spark 드라이버가 4 vCPU, 16GB 메모리 및 64GB 디스크에서 실행됩니다.

- GlueVersion – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

Glue 버전에 따라AWS Glue에서 지원하는 Apache Spark와 Python의 버전이 정해집니다. Python 버전은 개발 엔드포인트에서 ETL 스크립트를 실행하기 위해 지원되는 버전을 나타냅니다.

이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전에 대한 자세한 내용은 개발자 안내서의 Glue 버전을 참조하세요.

Glue 버전 지정 없이 생성된 개발 엔드포인트는 Glue 0.9로 기본 지정됩니다.

`CreateDevEndpoint` 또는 `UpdateDevEndpoint` API의 `Arguments` 파라미터를 사용하여 개발 엔드포인트에 지원되는 Python 버전을 지정할 수 있습니다. 인수가 제공되지 않으면 버전은 기본적으로 Python 2입니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  개발 엔드포인트로 할당되는 정의된 `workerType`의 작업자 수입니다.

  정의할 수 있는 최대 작업자 수는 `G.1X`의 경우 299개, `G.2X`의 경우 149개입니다.

- `NumberOfNodes` - 숫자(정수)입니다.

  이 `DevEndpoint`에 할당된 AWS Glue Glue 데이터 처리 장치(DPU) 수입니다.

- `AvailabilityZone` – UTF-8 문자열입니다.

  이 `DevEndpoint`가 위치한 AWS 가용 영역입니다.

- `VpcId` – UTF-8 문자열입니다.

  이 `DevEndpoint`가 사용한 Virtual Private Cloud(VPC)의 ID입니다.

- `ExtraPythonLibsS3Path` – UTF-8 문자열입니다.

  `DevEndpoint`에서 로드되어야 할 Amazon S3 버킷에 있는 하나 이상의 Python 라이브러리에 대한 경로입니다. 여러 값은 쉼표(,)로 구분된 완전한 경로여야 합니다.

  > **Note**
  >
  > `DevEndpoint`에서는 순수 Python 라이브러리만 사용할 수 있습니다. pandas Python 데이터 분석 라이브러리 등 C 확장을 활용하는 라이브러리는 현재 지원되지 않습니다.

- `ExtraJarsS3Path` – UTF-8 문자열입니다.

  `DevEndpoint`에서 로드되어야 할 S3 버킷에 있는 하나 이상의 Java `.jar` 파일에 대한 경로입니다.

> **Note**
>
> DevEndpoint에서는 순수 Java/Scala 라이브러리만 사용할 수 있습니다.

- FailureReason – UTF-8 문자열입니다.

  이 DevEndpoint의 현재 실패 이유입니다.
- LastUpdateStatus – UTF-8 문자열입니다.

  마지막 업데이트의 상태입니다.
- CreatedTimestamp – 타임스탬프입니다.

  이 DevEndpoint가 생성된 시점.
- LastModifiedTimestamp – 타임스탬프입니다.

  이 DevEndpoint가 마지막으로 수정된 시점.
- PublicKey – UTF-8 문자열입니다.

  인증용으로 이 DevEndpoint에서 사용될 퍼블릭 키입니다. 사용할 권장 속성이 퍼블릭 키이므로 이전 버전과의 호환성을 위해 이 속성이 제공됩니다.
- PublicKeys – 5개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  인증용으로 DevEndpoints에서 사용될 퍼블릭 키 목록입니다. 퍼블릭 키를 사용하면 클라이언트마다 다른 프라이빗 키를 지정할 수 있으므로 이 속성의 사용이 단일 퍼블릭 키보다 우선됩니다.

> **Note**
>
> 이전에 퍼블릭 키로 엔드포인트를 생성한 경우, 퍼블릭 키 목록을 설정할 수 있으려면 해당 키를 제거해야 합니다. deletePublicKeys 속성의 퍼블릭 키 콘텐츠와 addPublicKeys 속성의 새 키 목록을 사용하여 UpdateDevEndpoint API를 호출하십시오.

- SecurityConfiguration – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 DevEndpoint에 사용할 SecurityConfiguration 구조의 이름입니다.
- Arguments – 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

각 값은 UTF-8 문자열입니다.

`DevEndpoint` 구성에 사용되는 인수의 맵입니다.

유효한 인수는 다음과 같습니다.

- `"--enable-glue-datacatalog": ""`

`CreateDevEndpoint` 또는 `UpdateDevEndpoint` API의 `Arguments` 파라미터를 사용하여 개발 엔드포인트에 지원되는 Python 버전을 지정할 수 있습니다. 인수가 제공되지 않으면 버전은 기본적으로 Python 2입니다.

DevEndpointCustomLibraries 구조

개발 엔드포인트에 로드될 사용자 지정 라이브러리입니다.

필드

- `ExtraPythonLibsS3Path` – UTF-8 문자열입니다.

  `DevEndpoint`에서 로드되어야 할 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 하나 이상의 Python 라이브러리에 대한 경로입니다. 여러 값은 쉼표(,)로 구분된 완전한 경로여야 합니다.

  > **Note**
  >
  > `DevEndpoint`에서는 순수 Python 라이브러리만 사용할 수 있습니다. pandas Python 데이터 분석 라이브러리 등 C 확장을 활용하는 라이브러리는 현재 지원되지 않습니다.

- `ExtraJarsS3Path` – UTF-8 문자열입니다.

  `DevEndpoint`에서 로드되어야 할 S3 버킷에 있는 하나 이상의 Java `.jar` 파일에 대한 경로입니다.

  > **Note**
  >
  > `DevEndpoint`에서는 순수 Java/Scala 라이브러리만 사용할 수 있습니다.

운영

- CreateDevEndpoint 작업(Python: create_dev_endpoint)

- UpdateDevEndpoint 작업(Python: update_dev_endpoint)
- DeleteDevEndpoint 작업(Python: delete_dev_endpoint)
- GetDevEndpoint 작업(Python: get_dev_endpoint)
- GetDevEndpoints 작업(Python: get_dev_endpoints)
- BatchGetDevEndpoints 작업(Python: batch_get_dev_endpoints)
- ListDevEndpoints 작업(Python: list_dev_endpoints)

CreateDevEndpoint 작업(Python: create_dev_endpoint)

새 개발 엔드포인트를 생성합니다.

요청

- `EndpointName` – 필수(Required): UTF-8 문자열입니다.

  새 `DevEndpoint`에 지정된 이름입니다.
- `RoleArn` – 필수(Required): AWS IAM ARN string pattern과(와) 일치하는 UTF-8 문자열입니다.

  `DevEndpoint`의 IAM 역할입니다.
- `SecurityGroupIds` – UTF-8 문자열의 배열입니다.

  새로운 `DevEndpoint`가 사용할 보안 그룹의 보안 그룹 ID입니다.
- `SubnetId` – UTF-8 문자열입니다.

  사용할 새 `DevEndpoint`의 서브넷 ID입니다.
- `PublicKey` – UTF-8 문자열입니다.

  인증용으로 이 `DevEndpoint`에서 사용될 퍼블릭 키입니다. 사용할 권장 속성이 퍼블릭 키이므로 이전 버전과의 호환성을 위해 이 속성이 제공됩니다.
- `PublicKeys` – 5개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  인증용으로 개발 엔드포인트에서 사용될 퍼블릭 키 목록입니다. 퍼블릭 키를 사용하면 클라이언트마다 다른 프라이빗 키를 지정할 수 있으므로 이 속성의 사용이 단일 퍼블릭 키보다 우선됩니다.

> **Note**
>
> 이전에 퍼블릭 키로 엔드포인트를 생성한 경우, 퍼블릭 키 목록을 설정할 수 있으려면 해당 키를 제거해야 합니다. `deletePublicKeys` 속성의 퍼블릭 키 콘텐츠와 `addPublicKeys` 속성의 새 키 목록을 사용하여 `UpdateDevEndpoint` API를 호출하십시오.

- `NumberOfNodes` - 숫자(정수)입니다.

  이 `DevEndpoint`에 할당할 AWS Glue 데이터 처리 장치(DPU) 수입니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  개발 엔드포인트로 할당되는 미리 정의된 작업자 유형입니다. Standard, G.1X 또는 G.2X 값을 허용합니다.

  - `Standard` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 50GB 디스크와, 작업자당 실행기 2개를 제공합니다.
  - `G.1X` 작업자 유형의 경우, 각 작업자가 1DPU(4vCPU, 16GB 메모리, 64GB 디스크)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 메모리 집약적인 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다.
  - `G.2X` 작업자 유형의 경우, 각 작업자가 2DPU(8vCPU, 32GB 메모리, 128GB 디스크)에 매핑되고, 작업자당 실행기 1개를 제공합니다. 메모리 집약적인 작업의 경우 이 작업자 유형을 사용하는 것이 좋습니다.

  알려진 문제: 개발 엔드포인트가 `G.2X` `WorkerType` 구성으로 만들어질 경우 개발 엔드포인트의 Spark 드라이버가 4 vCPU, 16GB 메모리 및 64GB 디스크에서 실행됩니다.

- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  Glue 버전에 따라AWS Glue에서 지원하는 Apache Spark와 Python의 버전이 정해집니다. Python 버전은 개발 엔드포인트에서 ETL 스크립트를 실행하기 위해 지원되는 버전을 나타냅니다.

  이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전에 대한 자세한 내용은 개발자 안내서의 Glue 버전을 참조하세요.

  Glue 버전 지정 없이 생성된 개발 엔드포인트는 Glue 0.9로 기본 지정됩니다.

CreateDevEndpoint 또는 UpdateDevEndpoint API의 Arguments 파라미터를 사용하여 개발 엔드포인트에 지원되는 Python 버전을 지정할 수 있습니다. 인수가 제공되지 않으면 버전은 기본적으로 Python 2입니다.

- NumberOfWorkers - 숫자(정수)입니다.

  개발 엔드포인트로 할당되는 정의된 workerType의 작업자 수입니다.

  정의할 수 있는 최대 작업자 수는 G.1X의 경우 299개, G.2X의 경우 149개입니다.

- ExtraPythonLibsS3Path – UTF-8 문자열입니다.

  DevEndpoint에서 로드되어야 할 Amazon S3 버킷에 있는 하나 이상의 Python 라이브러리에 대한 경로입니다. 여러 값은 쉼표(,)로 구분된 완전한 경로여야 합니다.

  > **Note**
  >
  > DevEndpoint에서는 순수 Python 라이브러리만 사용할 수 있습니다. pandas Python 데이터 분석 라이브러리 등 C 확장을 활용하는 라이브러리는 아직 지원되지 않습니다.

- ExtraJarsS3Path – UTF-8 문자열입니다.

  DevEndpoint에서 로드되어야 할 S3 버킷에 있는 하나 이상의 Java .jar 파일에 대한 경로입니다.

- SecurityConfiguration – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 DevEndpoint에 사용할 SecurityConfiguration 구조의 이름입니다.

- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이 DevEndpoint에서 사용할 태그입니다. 태그를 사용하여 DevEndpoint에 대한 액세스를 제한할 수 있습니다. AWS Glue의 태그에 대한 자세한 내용은 개발자 안내서의 AWS Glue의 AWS 태그를 참조하세요.

- Arguments – 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

각 값은 UTF-8 문자열입니다.

DevEndpoint 구성에 사용되는 인수의 맵입니다.

응답

- EndpointName – UTF-8 문자열입니다.

  새로운 DevEndpoint에 할당된 이름입니다.
- Status – UTF-8 문자열입니다.

  새 DevEndpoint의 현재 상태입니다.
- SecurityGroupIds – UTF-8 문자열의 배열입니다.

  새로운 DevEndpoint에 지정된 보안 그룹입니다.
- SubnetId – UTF-8 문자열입니다.

  새로운 DevEndpoint에 지정된 서브넷 ID입니다.
- RoleArn – AWS IAM ARN string pattern과(와) 일치하는 UTF-8 문자열입니다.

  새로운 DevEndpoint에 할당된 역할의 Amazon 리소스 이름(ARN)입니다.
- YarnEndpointAddress – UTF-8 문자열입니다.

  이 DevEndpoint가 사용하는 YARN 엔드포인트 주소입니다.
- ZeppelinRemoteSparkInterpreterPort - 숫자(정수)입니다.

  원격 Apache Spark 인터프리터용 Apache Zeppelin 포트입니다.
- NumberOfNodes - 숫자(정수)입니다.

  이 DevEndpoint에 할당된 AWS Glue 데이터 처리 장치(DPU) 수입니다.
- WorkerType – UTF-8 문자열입니다(유효한 값: Standard="" | G.1X="" | G.2X="" | G.025X="" | G.4X="" | G.8X="" | Z.2X="").

  개발 엔드포인트로 할당되는 미리 정의된 작업자 유형입니다. Standard, G.1X 또는 G.2X 값일 수 있습니다.
- GlueVersion – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

Glue 버전에 따라AWS Glue에서 지원하는 Apache Spark와 Python의 버전이 정해집니다. Python 버전은 개발 엔드포인트에서 ETL 스크립트를 실행하기 위해 지원되는 버전을 나타냅니다.

이용 가능한 AWS Glue 버전과 그에 상응하는 Spark 및 Python 버전에 대한 자세한 내용은 개발자 안내서의 Glue 버전을 참조하세요.

- `NumberOfWorkers` - 숫자(정수)입니다.

  개발 엔드포인트로 할당되는 정의된 `workerType`의 작업자 수입니다.

- `AvailabilityZone` – UTF-8 문자열입니다.

  이 `DevEndpoint`가 위치한 AWS 가용 영역입니다.

- `VpcId` – UTF-8 문자열입니다.

  이 `DevEndpoint`가 사용한 Virtual Private Cloud(VPC)의 ID입니다.

- `ExtraPythonLibsS3Path` – UTF-8 문자열입니다.

  `DevEndpoint`에서 로드되는 S3 버킷에 있는 하나 이상의 Python 라이브러리에 대한 경로입니다.

- `ExtraJarsS3Path` – UTF-8 문자열입니다.

  `DevEndpoint`에서 로드되는 S3 버킷에 있는 하나 이상의 Java `.jar`에 대한 경로입니다.

- `FailureReason` – UTF-8 문자열입니다.

  이 `DevEndpoint`의 현재 실패 이유입니다.

- `SecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 `DevEndpoint`에 사용 중인 `SecurityConfiguration` 구조의 이름입니다.

- `CreatedTimestamp` – 타임스탬프입니다.

  이 `DevEndpoint`가 생성된 시점입니다.

- `Arguments` – 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  이 `DevEndpoint`를 구성하는 데 사용되는 인수의 맵입니다.

유효한 인수는 다음과 같습니다.

- `"--enable-glue-datacatalog": ""`

`CreateDevEndpoint` 또는 `UpdateDevEndpoint` API의 `Arguments` 파라미터를 사용하여 개발 엔드포인트에 지원되는 Python 버전을 지정할 수 있습니다. 인수가 제공되지 않으면 버전은 기본적으로 Python 2입니다.

오류

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`

UpdateDevEndpoint 작업(Python: update_dev_endpoint)

지정한 개발 엔드포인트를 업데이트합니다.

요청

- `EndpointName` – 필수(Required): UTF-8 문자열입니다.

  업데이트할 `DevEndpoint`의 이름입니다.
- `PublicKey` – UTF-8 문자열입니다.

  사용할 `DevEndpoint`의 퍼블릭 키입니다.
- `AddPublicKeys` – 5개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  사용할 `DevEndpoint`의 퍼블릭 키 목록입니다.
- `DeletePublicKeys` – 5개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  `DevEndpoint`에서 삭제할 퍼블릭 키 목록입니다.

- `CustomLibraries` – DevEndpointCustomLibraries 객체입니다.

  이 `DevEndpoint`에 로드될 사용자 지정 Python 또는 Java 라이브러리입니다.

- `UpdateEtlLibraries` – 부울입니다.

  개발 엔드포인트에 로드될 사용자 지정 라이브러리 목록을 업데이트해야 하면 `True`이고 그렇지 않으면 `False`입니다.

- `DeleteArguments` – UTF-8 문자열의 배열입니다.

  `DevEndpoint` 구성에 사용되는 인수의 맵에서 삭제할 인수 키의 목록입니다.

- `AddArguments` – 100개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  `DevEndpoint` 구성에 사용되는 인수의 맵을 추가하기 위한 인수의 맵입니다.

  유효한 인수는 다음과 같습니다.

  - `"--enable-glue-datacatalog": ""`

  `CreateDevEndpoint` 또는 `UpdateDevEndpoint` API의 `Arguments` 파라미터를 사용하여 개발 엔드포인트에 지원되는 Python 버전을 지정할 수 있습니다. 인수가 제공되지 않으면 버전은 기본적으로 Python 2입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`

DeleteDevEndpoint 작업(Python: delete_dev_endpoint)

지정한 개발 엔드포인트를 삭제합니다.

요청

- `EndpointName` – 필수(Required): UTF-8 문자열입니다.

  `DevEndpoint`의 이름입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

GetDevEndpoint 작업(Python: get_dev_endpoint)

지정된 개발 엔드포인트에 대한 정보를 가져옵니다.

> **Note**
>
> Virtual Private Cloud(VPC)에서 개발 엔드포인트를 생성하면, AWS Glue은(는) 프라이빗 IP 주소만 반환하며 퍼블릭 IP 주소 필드는 입력되지 않습니다. 비 VPC 개발 엔드포인트를 생성할 때 AWS Glue은(는) 퍼블릭 IP 주소만 반환합니다.

요청

- `EndpointName` – 필수(Required): UTF-8 문자열입니다.

  정보를 검색할 `DevEndpoint`의 이름입니다.

응답

- `DevEndpoint` – DevEndpoint 객체입니다.

  `DevEndpoint` 정의입니다.

오류

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

GetDevEndpoints 작업(Python: get_dev_endpoints)

이 AWS 계정에서 모든 개발 엔드포인트를 검색합니다.

> **Note**
>
> Virtual Private Cloud(VPC)에서 개발 엔드포인트를 생성하면, AWS Glue는 프라이빗 IP 주소만 반환하며 퍼블릭 IP 주소 필드는 입력되지 않습니다. 비 VPC 개발 엔드포인트를 생성할 때 AWS Glue은(는) 퍼블릭 IP 주소만 반환합니다.

요청

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 정보의 최대 크기.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `DevEndpoints` – DevEndpoint 객체의 배열입니다.

  `DevEndpoint` 정의 목록입니다.

- NextToken – UTF-8 문자열입니다.

  모든 DevEndpoint 정의가 아직 반환되지 않은 경우의 지속 토큰입니다.

오류

- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- InvalidInputException

BatchGetDevEndpoints 작업(Python: batch_get_dev_endpoints)

주어진 개발 엔드포인트 이름 목록에 대한 리소스 메타데이터 목록을 반환합니다. ListDevEndpoints 작업을 호출한 후에는 권한이 부여된 데이터에 액세스하기 위해 이 작업을 호출할 수 있습니다. 이 작업은 태그를 사용하는 권한 조건을 포함해 모든 IAM 권한을 지원합니다.

요청

- customerAccountId – UTF-8 문자열입니다.

  AWS 계정 ID입니다.
- DevEndpointNames – 필수(Required): 1~25개 문자열의 UTF-8 문자열의 배열입니다.

  DevEndpoint 이름(ListDevEndpoint 작업에서 반환된 이름일 수 있음)의 목록입니다.

응답

- DevEndpoints – DevEndpoint 객체의 배열입니다.

  DevEndpoint 정의 목록입니다.
- DevEndpointsNotFound – 1~25개 문자열로 구성된 UTF-8 문자열의 배열입니다.

  찾을 수 없는 DevEndpoints 목록입니다.

오류

- AccessDeniedException

- InternalServiceException
- OperationTimeoutException
- InvalidInputException

ListDevEndpoints 작업(Python: list_dev_endpoints)

이 AWS 계정의 모든 DevEndpoint 리소스 또는 지정된 태그를 가진 리소스의 이름을 검색합니다. 이 작업을 통해 계정에서 사용 가능한 리소스와 그 이름을 확인할 수 있습니다.

이 작업을 수행하면 응답에서 필터로 사용할 수 있는 선택 사항인 Tags 필드가 검색되기 때문에 태그가 지정된 리소스를 하나의 그룹으로 검색할 수 있습니다. 태그 필터링을 사용하기로 선택하면 태그가 포함된 리소스만 검색됩니다.

요청

- NextToken – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.
- MaxResults – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 최대 크기.
- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이렇게 태그가 지정된 리소스만 반환하도록 지정합니다.

응답

- DevEndpointNames – UTF-8 문자열의 배열입니다.

  계정의 모든 DevEndpoint 또는 지정된 태그를 가진 DevEndpoint의 이름입니다.
- NextToken – UTF-8 문자열입니다.

  반환된 목록이 사용가능한 마지막 지표를 포함하지 경우의 연속 토큰입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Schema Registry

스키마 레지스트리 API는 AWS Glue에서의 스키마 작업과 관련된 API 및 데이터 유형에 대해 설명합니다.

데이터 타입

- RegistryId 구조
- RegistryListItem 구조
- MetadataInfo 구조
- OtherMetadataValueListItem 구조
- SchemaListItem 구조
- SchemaVersionListItem 구조
- MetadataKeyValuePair 구조
- SchemaVersionErrorItem 구조
- ErrorDetails 구조
- SchemaVersionNumber 구조
- SchemaId 구조

RegistryId 구조

레지스트리 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.

필드

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

레지스트리의 이름입니다. 조회에만 사용됩니다. RegistryArn 또는 RegistryName 중 하나가 제공되어야 합니다.

- RegistryArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  업데이트할 레지스트리의 ARN입니다. RegistryArn 또는 RegistryName 중 하나가 제공되어야 합니다.

RegistryListItem 구조

레지스트리에 대한 세부 정보를 포함하는 구조입니다.

필드

- RegistryName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 이름입니다.

- RegistryArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 Amazon 리소스 이름(ARN)입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  레지스트리에 대한 설명입니다.

- Status – UTF-8 문자열입니다(유효한 값: AVAILABLE | DELETING).

  레지스트리의 상태입니다.

- CreatedTime – UTF-8 문자열입니다.

  레지스트리가 생성된 데이터입니다.

- UpdatedTime – UTF-8 문자열입니다.

  레지스트리가 업데이트된 날짜입니다.

MetadataInfo 구조

스키마 버전에 대한 메타데이터 정보를 포함하는 구조입니다.

필드

- `MetadataValue` – Custom string pattern #14과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  메타데이터 키의 해당하는 값입니다.
- `CreatedTime` – UTF-8 문자열입니다.

  항목이 생성된 시간입니다.
- `OtherMetadataValueList` – OtherMetadataValueListItem 객체의 배열입니다.

  동일한 메타데이터 키에 속하는 다른 메타데이터입니다.

OtherMetadataValueListItem 구조

동일한 메타데이터 키에 속하는 스키마 버전에 대한 다른 메타데이터를 포함하는 구조입니다.

필드

- `MetadataValue` – Custom string pattern #14과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  동일한 메타데이터 키에 속하는 스키마 버전에 대한 다른 메타데이터를 포함하는 구조입니다.
- `CreatedTime` – UTF-8 문자열입니다.

  항목이 생성된 시간입니다.

SchemaListItem 구조

스키마에 대한 최소한의 세부 정보를 포함하는 객체입니다.

필드

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

스키마가 있는 레지스트리의 이름입니다.

- SchemaName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  스키마의 이름입니다.

- SchemaArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN)입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  스키마에 대한 설명입니다.

- SchemaStatus – UTF-8 문자열입니다(유효한 값: AVAILABLE | PENDING | DELETING).

  스키마의 상태입니다.

- CreatedTime – UTF-8 문자열입니다.

  스키마가 생성된 날짜 및 시간입니다.

- UpdatedTime – UTF-8 문자열입니다.

  스키마가 업데이트된 날짜 및 시간입니다.

SchemaVersionListItem 구조

스키마 버전에 대한 세부 정보를 포함하는 객체입니다.

필드

- SchemaArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN).

- SchemaVersionId – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 고유한 식별자입니다.

- `VersionNumber` - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  스키마 버전의 상태입니다.

- `CreatedTime` – UTF-8 문자열입니다.

  스키마 버전이 생성된 날짜 및 시간입니다.

MetadataKeyValuePair 구조

메타데이터에 대한 키 값 페어를 포함하는 구조입니다.

필드

- `MetadataKey` – Custom string pattern #14과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  메타데이터 키입니다.

- `MetadataValue` – Custom string pattern #14과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  메타데이터 키의 해당하는 값입니다.

SchemaVersionErrorItem 구조

스키마 버전에 대한 작업에 대한 오류 세부 정보가 포함된 객체입니다.

필드

- `VersionNumber` - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.

- `ErrorDetails` – ErrorDetails 객체입니다.

  스키마 버전에 대한 오류 세부 정보입니다.

ErrorDetails 구조

오류 세부 정보가 포함된 객체입니다.

필드

- `ErrorCode` – UTF-8 문자열입니다.

  오류에 대한 오류 코드입니다.
- `ErrorMessage` – UTF-8 문자열입니다.

  오류에 대한 오류 메시지입니다.

SchemaVersionNumber 구조

스키마 버전 정보를 포함하는 구조입니다.

필드

- `LatestVersion` – 부울입니다.

  스키마에 사용할 수 있는 최신 버전입니다.
- `VersionNumber` - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.

SchemaId 구조

AWS Glue 스키마 레지스트리에 있는 스키마의 고유 ID입니다.

필드

- `SchemaArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN). `SchemaArn` 또는 `SchemaName` 중 하나가 제공되어야 합니다.
- `SchemaName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

스키마의 이름입니다. SchemaArn 또는 SchemaName 중 하나가 제공되어야 합니다.

- RegistryName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  해당 스키마가 포함된 스키마 레지스트리의 이름입니다.

운영

- CreateRegistry 작업(Python: create_registry)
- CreateSchema 작업(Python: create_schema)
- GetSchema 작업(Python: get_schema)
- ListSchemaVersions 작업(Python: list_schema_versions)
- GetSchemaVersion 작업(Python: get_schema_version)
- GetSchemaVersionsDiff 작업(Python: get_schema_versions_diff)
- ListRegistries 작업(Python: list_registries)
- ListSchemas 작업(Python: list_schemas)
- RegisterSchemaVersion 작업(Python: register_schema_version)
- UpdateSchema 작업(Python: update_schema)
- CheckSchemaVersionValidity 작업(Python: check_schema_version_validity)
- UpdateRegistry 작업(Python: update_registry)
- GetSchemaByDefinition 작업(Python: get_schema_by_definition)
- GetRegistry 작업(Python: get_registry)
- PutSchemaVersionMetadata 작업(Python: put_schema_version_metadata)
- QuerySchemaVersionMetadata 작업(Python: query_schema_version_metadata)
- RemoveSchemaVersionMetadata 작업(Python: remove_schema_version_metadata)
- DeleteRegistry 작업(Python: delete_registry)
- DeleteSchema 작업(Python: delete_schema)
- DeleteSchemaVersions 작업(Python: delete_schema_versions)

CreateRegistry 작업(Python: create_registry)

스키마 컬렉션을 보유하는 데 사용할 수 있는 새 레지스트리를 생성합니다.

요청

- `RegistryName` – 필수(Required): Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리 이름은 최대 255자이며 문자, 숫자, 하이픈, 밑줄, 달러 기호 또는 해시 표시만 포함할 수 있습니다. 공백은 없습니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  레지스트리에 대한 설명입니다. 설명이 제공되지 않으면 이에 대한 기본값이 없습니다.

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  키 값 페어가 포함된 AWS 태그이며 콘솔, 명령줄 또는 API로 검색할 수 있습니다.

응답

- `RegistryArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  새로 생성된 레지스트리의 Amazon 리소스 이름(ARN)입니다.

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 이름입니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  레지스트리에 대한 설명입니다.

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

레지스트리에 대한 태그입니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

CreateSchema 작업(Python: create_schema)

새 스키마 집합을 생성하고 스키마 정의를 등록합니다. 실제로 버전을 등록하지 않고 스키마 집합이 이미 존재하는 경우 오류를 반환합니다.

스키마 집합이 생성되면 버전 체크포인트가 첫 번째 버전으로 설정됩니다. 호환성 모드 "DISABLED"는 첫 번째 스키마 버전 이후에 추가되는 모든 추가 스키마 버전을 제한합니다. 다른 모든 호환성 모드의 경우 호환성 설정의 검증은 `RegisterSchemaVersion` API를 사용할 때 두 번째 버전부터만 적용됩니다.

이 API가 `RegistryId` 없이 호출되면 레지스트리 데이터베이스 테이블에 "default-registry" 항목이 생성됩니다(아직 없는 경우).

요청

- `RegistryId` – RegistryId 객체입니다.

  이것은 레지스트리 자격 증명 필드를 포함하는 래퍼 셰이프입니다. 제공되지 않으면 기본 레지스트리가 사용됩니다. 동일한 ARN 포맷은 `arn:aws:glue:us-east-2:<customer id>:registry/default-registry:random-5-letter-id`입니다.
- `SchemaName` – 필수(Required): Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  스키마 이름은 최대 255자이며 문자, 숫자, 하이픈, 밑줄, 달러 기호 또는 해시 표시만 포함할 수 있습니다. 공백은 없습니다.

- `DataFormat` – 필수: UTF-8 문자열입니다(유효한 값: `AVRO` | `JSON` | `PROTOBUF`).

  스키마 정의의 데이터 형식입니다. 현재 `AVRO`, `JSON` 및 `PROTOBUF`가 지원됩니다.

- `Compatibility` – UTF-8 문자열입니다(유효한 값: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  스키마의 호환성 모드입니다. 가능한 값은 다음과 같습니다.

  - NONE: 호환 모드가 적용되지 않습니다. 개발 시나리오에서 또는 스키마에 적용할 호환성 모드를 모르는 경우 이 선택 사항을 사용할 수 있습니다. 추가된 모든 새 버전은 호환성 검사를 거치지 않고 수락됩니다.
  - DISABLED: 이 호환성 선택 항목은 특정 스키마에 대한 버전 관리를 방지합니다. 이 선택 사항을 사용하여 스키마의 향후 버전 관리를 방지할 수 있습니다.
  - BACKWARD: 이 호환성 선택 항목은 데이터 수신자가 현재 및 하나의 이전 스키마 버전을 모두 읽을 수 있도록 하므로 권장됩니다. 즉, 예를 들어 새 스키마 버전은 데이터 필드를 삭제하거나 이러한 필드의 유형을 변경할 수 없으므로 이전 버전을 사용하는 리더가 읽을 수 없습니다.
  - BACKWARD_ALL: 이 호환성 선택을 통해 데이터 수신자는 현재 및 모든 이전 스키마 버전을 모두 읽을 수 있습니다. 필드를 삭제하거나 선택적 필드를 추가하고 모든 이전 스키마 버전과의 호환성을 확인해야 할 때 이 선택 사항을 사용할 수 있습니다.
  - FORWARD: 이 호환성 선택을 통해 데이터 수신기는 현재 및 하나의 다음 스키마 버전을 모두 읽을 수 있지만 반드시 이후 버전은 아닙니다. 필드를 추가하거나 선택적 필드를 삭제해야 할 때 이 선택 사항을 사용할 수 있지만 마지막 스키마 버전과의 호환성만 확인합니다.
  - FORWARD_ALL: 이 호환성 선택을 통해 데이터 수신기는 새로 등록된 스키마의 생산자가 작성한 내용을 읽을 수 있습니다. 필드를 추가하거나 선택적 필드를 삭제하고 모든 이전 스키마 버전과의 호환성을 확인해야 할 때 이 선택 사항을 사용할 수 있습니다.
  - FULL: 이 호환성 선택을 통해 데이터 수신자는 이전 또는 다음 버전의 스키마를 사용하여 생산자가 작성한 데이터를 읽을 수 있지만 반드시 이전 또는 이후 버전은 아닙니다. 선택적 필드를 추가하거나 제거해야 할 때 이 선택 사항을 사용할 수 있지만 마지막 스키마 버전과의 호환성만 확인합니다.
  - FULL_ALL: 이 호환성 선택을 통해 데이터 수신자는 모든 이전 스키마 버전을 사용하여 생산자가 작성한 데이터를 읽을 수 있습니다. 선택적 필드를 추가 또는 제거하고 모든 이전 스키마 버전과의 호환성을 확인해야 할 때 이 선택 사항을 사용할 수 있습니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  스키마에 대한 설명입니다(선택 사항). 설명이 제공되지 않으면 이에 대한 자동 기본값이 없습니다.

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  키 값 페어가 포함된 AWS 태그이며 콘솔, 명령줄 또는 API로 검색할 수 있습니다. 지정된 경우AWS tags-on-create 패턴을 따릅니다.

- `SchemaDefinition` – Custom string pattern #13과(와) 일치하는 1~170,000바이트 길이의 UTF-8 문자열입니다.

  `SchemaName`에 대해 `DataFormat` 설정을 사용하는 스키마 정의입니다.

응답

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 이름입니다.

- `RegistryArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 Amazon 리소스 이름(ARN)입니다.

- `SchemaName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  스키마의 이름입니다.

- `SchemaArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN).

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  스키마에 대한 설명입니다(생성 시 지정한 경우).

- `DataFormat` – UTF-8 문자열입니다(유효한 값: `AVRO` | `JSON` | `PROTOBUF`).

  스키마 정의의 데이터 형식입니다. 현재 `AVRO`, `JSON` 및 `PROTOBUF`가 지원됩니다.

- `Compatibility` – UTF-8 문자열입니다(유효한 값: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  스키마 호환성 모드입니다.
- `SchemaCheckpoint` - 1~100,000의 숫자(정수)입니다.

  체크포인트의 버전 번호(호환성 모드가 마지막으로 변경된 시간)입니다.
- `LatestSchemaVersion` - 1~100,000의 숫자(정수)입니다.

  반환된 스키마 정의와 연결된 스키마의 최신 버전입니다.
- `NextSchemaVersion` - 1~100,000의 숫자(정수)입니다.

  반환된 스키마 정의와 연결된 스키마의 다음 버전입니다.
- `SchemaStatus` – UTF-8 문자열입니다(유효한 값: `AVAILABLE` | `PENDING` | `DELETING`).

  스키마의 상태입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  스키마의 태그입니다.
- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  첫 번째 스키마 버전의 고유 식별자입니다.
- `SchemaVersionStatus` – UTF-8 문자열입니다(유효한 값: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  생성된 첫 번째 스키마 버전의 상태입니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `AlreadyExistsException`

- ResourceNumberLimitExceededException
- ConcurrentModificationException
- InternalServiceException

GetSchema 작업(Python: get_schema)

지정된 스키마를 자세히 설명합니다.

요청

- SchemaId – 필수(Required): SchemaId 객체입니다.

  스키마 자격 증명 필드를 포함하는 래퍼 구조입니다. 구조에는 다음이 포함됩니다.

  - SchemaId$SchemaArn: 스키마의 Amazon 리소스 이름(ARN)입니다. SchemaArn 또는 SchemaName 및 RegistryName이 제공되어야 합니다.
  - SchemaId$SchemaName: 스키마의 이름입니다. SchemaArn 또는 SchemaName 및 RegistryName이 제공되어야 합니다.

응답

- RegistryName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 이름입니다.

- RegistryArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 Amazon 리소스 이름(ARN)입니다.

- SchemaName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  스키마의 이름입니다.

- SchemaArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN).

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  스키마에 대한 설명입니다(생성 시 지정한 경우).

- DataFormat – UTF-8 문자열입니다(유효한 값: AVRO | JSON | PROTOBUF).

  스키마 정의의 데이터 형식입니다. 현재 AVRO, JSON 및 PROTOBUF가 지원됩니다.

- Compatibility – UTF-8 문자열입니다(유효한 값: NONE | DISABLED | BACKWARD | BACKWARD_ALL | FORWARD | FORWARD_ALL | FULL | FULL_ALL).

  스키마의 호환성 모드입니다.

- SchemaCheckpoint - 1~100,000의 숫자(정수)입니다.

  체크포인트의 버전 번호(호환성 모드가 마지막으로 변경된 시간)입니다.

- LatestSchemaVersion - 1~100,000의 숫자(정수)입니다.

  반환된 스키마 정의와 연결된 스키마의 최신 버전입니다.

- NextSchemaVersion - 1~100,000의 숫자(정수)입니다.

  반환된 스키마 정의와 연결된 스키마의 다음 버전입니다.

- SchemaStatus – UTF-8 문자열입니다(유효한 값: AVAILABLE | PENDING | DELETING).

  스키마의 상태입니다.

- CreatedTime – UTF-8 문자열입니다.

  스키마가 생성된 날짜 및 시간입니다.

- UpdatedTime – UTF-8 문자열입니다.

  스키마가 업데이트된 날짜 및 시간입니다.

오류

- InvalidInputException
- AccessDeniedException
- EntityNotFoundException
- InternalServiceException

ListSchemaVersions 작업(Python: list_schema_versions)

최소한의 정보로 생성한 스키마 버전 목록을 반환합니다. [삭제됨(Deleted)] 상태의 스키마 버전은 결과에 포함되지 않습니다. 사용 가능한 스키마 버전이 없으면 빈 결과가 반환됩니다.

요청

- `SchemaId` – 필수(Required): SchemaId 객체입니다.

  스키마 자격 증명 필드를 포함하는 래퍼 구조입니다. 구조에는 다음이 포함됩니다.

  - SchemaId$SchemaArn: 스키마의 Amazon 리소스 이름(ARN)입니다. `SchemaArn` 또는 `SchemaName` 및 `RegistryName`이 제공되어야 합니다.
  - SchemaId$SchemaName: 스키마의 이름입니다. `SchemaArn` 또는 `SchemaName` 및 `RegistryName`이 제공되어야 합니다.
- `MaxResults` – 1~100의 숫자(정수)입니다.

  페이지당 필요한 최대 결과 수입니다. 값이 제공되지 않으면 페이지당 기본값이 25로 설정됩니다.
- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `Schemas` – SchemaVersionListItem 객체의 배열입니다.

  각 스키마 버전의 세부 정보를 포함하는 `SchemaVersionList` 객체의 배열입니다.
- `NextToken` – UTF-8 문자열입니다.

  목록의 현재 세그먼트가 마지막이 아니면 반환된 토큰 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersion 작업(Python: get_schema_version)

스키마 버전을 생성하거나 등록할 때 할당된 고유 ID로 지정된 스키마를 가져옵니다. [삭제됨(Deleted)] 상태의 스키마 버전은 결과에 포함되지 않습니다.

요청

- `SchemaId` – SchemaId 객체입니다.

  스키마 자격 증명 필드를 포함하는 래퍼 구조입니다. 구조에는 다음이 포함됩니다.

  - SchemaId$SchemaArn: 스키마의 Amazon 리소스 이름(ARN)입니다. `SchemaArn` 또는 `SchemaName` 및 `RegistryName`이 제공되어야 합니다.
  - SchemaId$SchemaName: 스키마의 이름입니다. `SchemaArn` 또는 `SchemaName` 및 `RegistryName`이 제공되어야 합니다.
- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 `SchemaVersionId`입니다. 이 필드는 스키마 ID로 가져오기 위해 필요합니다. 이것 또는 `SchemaId` 래퍼를 제공해야 합니다.
- `SchemaVersionNumber` – SchemaVersionNumber 객체입니다.

  스키마의 버전 번호입니다.

응답

- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 `SchemaVersionId`입니다.
- `SchemaDefinition` – Custom string pattern #13과(와) 일치하는 1~170,000바이트 길이의 UTF-8 문자열입니다.

  스키마 ID에 대한 스키마 정의입니다.
- `DataFormat` – UTF-8 문자열입니다(유효한 값: `AVRO` | `JSON` | `PROTOBUF`).

  스키마 정의의 데이터 형식입니다. 현재 `AVRO`, `JSON` 및 `PROTOBUF`가 지원됩니다.
- `SchemaArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

스키마의 Amazon 리소스 이름(ARN).

- `VersionNumber` - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  스키마 버전의 상태입니다.

- `CreatedTime` – UTF-8 문자열입니다.

  스키마 버전이 생성된 날짜 및 시간입니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersionsDiff 작업(Python: get_schema_versions_diff)

Schema Registry에 저장된 두 스키마 버전 간에 지정된 차이 유형의 스키마 버전 차이를 가져옵니다.

이 API를 사용하면 동일한 스키마 아래의 두 스키마 정의 간에 두 스키마 버전을 비교할 수 있습니다.

요청

- `SchemaId` – 필수(Required): SchemaId 객체입니다.

  스키마 자격 증명 필드를 포함하는 래퍼 구조입니다. 구조에는 다음이 포함됩니다.

  - SchemaId$SchemaArn: 스키마의 Amazon 리소스 이름(ARN)입니다. `SchemaArn` 또는 `SchemaName` 중 하나가 제공되어야 합니다.
  - SchemaId$SchemaName: 스키마의 이름입니다. `SchemaArn` 또는 `SchemaName` 중 하나가 제공되어야 합니다.

- `FirstSchemaVersionNumber` – 필수(Required): SchemaVersionNumber 객체입니다.

  비교할 두 스키마 버전 중 첫 번째 버전입니다.

- SecondSchemaVersionNumber – 필수(Required): SchemaVersionNumber 객체입니다.

  비교할 두 스키마 버전 중 두 번째 버전입니다.

- SchemaDiffType – 필수: UTF-8 문자열입니다(유효한 값: SYNTAX_DIFF).

  현재 지원되는 diff 유형인 SYNTAX_DIFF를 나타냅니다.

응답

- Diff – Custom string pattern #13과(와) 일치하는 1~340,000바이트 길이의 UTF-8 문자열입니다.

  JsonPatch 포맷의 문자열로 스키마 간의 차이점입니다.

오류

- InvalidInputException
- EntityNotFoundException
- AccessDeniedException
- InternalServiceException

ListRegistries 작업(Python: list_registries)

최소한의 레지스트리 정보로 생성한 레지스트리 목록을 반환합니다. Deleting 상태의 레지스트리는 결과에 포함되지 않습니다. 사용 가능한 레지스트리가 없으면 빈 결과가 반환됩니다.

요청

- MaxResults – 1~100의 숫자(정수)입니다.

  페이지당 필요한 최대 결과 수입니다. 값이 제공되지 않으면 페이지당 기본값이 25로 설정됩니다.

- NextToken – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- Registries – RegistryListItem 객체의 배열입니다.

각 레지스트리에 대한 최소한의 세부 정보를 포함하는 RegistryDetailedListItem 객체의 배열입니다.

- NextToken – UTF-8 문자열입니다.

  목록의 현재 세그먼트가 마지막이 아니면 반환된 토큰 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

오류

- InvalidInputException
- AccessDeniedException
- InternalServiceException

ListSchemas 작업(Python: list_schemas)

최소한의 세부 정보가 포함된 스키마 목록을 반환합니다. [삭제 중(Deleting)] 상태의 스키마는 결과에 포함되지 않습니다. 사용 가능한 스키마가 없으면 빈 결과가 반환됩니다.

RegistryId가 제공되지 않으면 레지스트리 전체의 모든 스키마가 API 응답의 일부가 됩니다.

요청

- RegistryId – RegistryId 객체입니다.

  레지스트리 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.

- MaxResults – 1~100의 숫자(정수)입니다.

  페이지당 필요한 최대 결과 수입니다. 값이 제공되지 않으면 페이지당 기본값이 25로 설정됩니다.

- NextToken – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- Schemas – SchemaListItem 객체의 배열입니다.

  각 스키마의 세부 정보를 포함하는 SchemaListItem 객체의 배열입니다.

- NextToken – UTF-8 문자열입니다.

  목록의 현재 세그먼트가 마지막이 아니면 반환된 토큰 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

오류

- InvalidInputException
- AccessDeniedException
- EntityNotFoundException
- InternalServiceException

RegisterSchemaVersion 작업(Python: register_schema_version)

기존 스키마에 새 버전을 추가합니다. 새 버전의 스키마가 스키마 집합의 호환성 요구 사항을 충족하지 않는 경우 오류를 반환합니다. 이 API는 새 스키마 집합을 생성하지 않으며 스키마 집합이 Schema Registry에 아직 없는 경우 404 오류를 반환합니다.

이것이 Schema Registry에 등록되는 첫 번째 스키마 정의인 경우 이 API는 스키마 버전을 저장하고 즉시 반환합니다. 그렇지 않으면 이 호출은 호환성 모드로 인해 다른 작업보다 더 오래 실행될 가능성이 있습니다. SchemaVersionId로 GetSchemaVersion API를 호출하여 호환 모드를 확인할 수 있습니다.

동일한 스키마 정의가 이미 Schema Registry에 버전으로 저장되어 있는 경우 기존 스키마의 스키마 ID가 호출자에게 반환됩니다.

요청

- SchemaId – 필수(Required): SchemaId 객체입니다.

  스키마 자격 증명 필드를 포함하는 래퍼 구조입니다. 구조에는 다음이 포함됩니다.

  - SchemaId$SchemaArn: 스키마의 Amazon 리소스 이름(ARN)입니다. SchemaArn 또는 SchemaName 및 RegistryName이 제공되어야 합니다.
  - SchemaId$SchemaName: 스키마의 이름입니다. SchemaArn 또는 SchemaName 및 RegistryName이 제공되어야 합니다.
- SchemaDefinition – 필수(Required): Custom string pattern #13과(와) 일치하는 1~170,000바이트 길이의 UTF-8 문자열입니다.

SchemaName에 대해 DataFormat 설정을 사용하는 스키마 정의입니다.

응답

- SchemaVersionId – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  이 스키마의 버전을 나타내는 고유 ID입니다.
- VersionNumber - 1~100,000의 숫자(정수)입니다.

  이 스키마의 버전입니다(동기화 흐름 전용, 이것이 첫 번째 버전인 경우).
- Status – UTF-8 문자열입니다(유효한 값: AVAILABLE | PENDING | FAILURE | DELETING).

  스키마 버전의 상태입니다.

오류

- InvalidInputException
- AccessDeniedException
- EntityNotFoundException
- ResourceNumberLimitExceededException
- ConcurrentModificationException
- InternalServiceException

UpdateSchema 작업(Python: update_schema)

스키마 집합에 대한 설명, 호환성 설정 또는 버전 검사점을 업데이트합니다.

호환성 설정을 업데이트하기 위해 호출은 새 호환성 설정을 사용하여 전체 스키마 버전 집합에 대한 호환성을 확인하지 않습니다. Compatibility 값이 제공되면 VersionNumber(체크포인트)도 필요합니다. API는 일관성을 위해 체크포인트 버전 번호를 검증합니다.

VersionNumber(체크포인트) 값이 제공되면 Compatibility는 선택 사항이며 스키마에 대한 검사점을 설정/재설정하는 데 사용할 수 있습니다.

이 업데이트는 스키마가 AVAILABLE 상태인 경우에만 발생합니다.

요청

- SchemaId – 필수(Required): SchemaId 객체입니다.

  스키마 자격 증명 필드를 포함하는 래퍼 구조입니다. 구조에는 다음이 포함됩니다.

  - SchemaId$SchemaArn: 스키마의 Amazon 리소스 이름(ARN)입니다. SchemaArn 또는 SchemaName 중 하나가 제공되어야 합니다.
  - SchemaId$SchemaName: 스키마의 이름입니다. SchemaArn 또는 SchemaName 중 하나가 제공되어야 합니다.
- SchemaVersionNumber – SchemaVersionNumber 객체입니다.

  체크 포인팅에 필요한 버전 번호입니다. VersionNumber 또는 Compatibility 중 하나가 제공되어야 합니다.
- Compatibility – UTF-8 문자열입니다(유효한 값: NONE | DISABLED | BACKWARD | BACKWARD_ALL | FORWARD | FORWARD_ALL | FULL | FULL_ALL).

  스키마에 대한 새 호환성 설정입니다.
- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  스키마에 대한 새 설명입니다.

응답

- SchemaArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN).
- SchemaName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  스키마의 이름입니다.
- RegistryName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  해당 스키마가 포함된 레지스트리의 이름입니다.

오류

- InvalidInputException
- AccessDeniedException
- EntityNotFoundException
- ConcurrentModificationException
- InternalServiceException

CheckSchemaVersionValidity 작업(Python: check_schema_version_validity)

제공된 스키마를 검증합니다. 이 호출은 부작용이 없으며 단순히 DataFormat을 포맷으로 사용하여 제공된 스키마를 사용하여 검증합니다. 스키마 집합 이름을 사용하지 않으므로 호환성 검사가 수행되지 않습니다.

요청

- DataFormat – 필수: UTF-8 문자열입니다(유효한 값: AVRO | JSON | PROTOBUF).

  스키마 정의의 데이터 형식입니다. 현재 AVRO, JSON 및 PROTOBUF가 지원됩니다.
- SchemaDefinition – 필수(Required): Custom string pattern #13과(와) 일치하는 1~170,000바이트 길이의 UTF-8 문자열입니다.

  검증해야 하는 스키마의 정의입니다.

응답

- Valid – 부울입니다.

  스키마가 유효하면 true를 반환하고 그렇지 않으면 false를 반환합니다.
- Error – 1~5,000바이트 길이의 UTF-8 문자열입니다.

  검증 실패 메시지입니다.

오류

- InvalidInputException
- AccessDeniedException

- `InternalServiceException`

UpdateRegistry 작업(Python: update_registry)

스키마 컬렉션을 보유하는 데 사용되는 기존 레지스트리를 업데이트합니다. 업데이트된 속성은 레지스트리와 관련되며 레지스트리 내의 스키마를 수정하지 않습니다.

요청

- `RegistryId` – 필수(Required): RegistryId 객체입니다.

  이는 레지스트리 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.
- `Description` – 필수(Required): URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  레지스트리에 대한 설명입니다. 설명이 제공되지 않으면 이 필드는 업데이트되지 않습니다.

응답

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트된 레지스트리의 이름입니다.
- `RegistryArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  업데이트된 레지스트리의 Amazon 리소스 이름(ARN)입니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InternalServiceException`

GetSchemaByDefinition 작업(Python: get_schema_by_definition)

SchemaDefinition으로 스키마를 검색합니다. 스키마 정의는 Schema Registry로 전송되고 정규화되고 해시됩니다. 해시가 SchemaName 또는 ARN의 범위 내에서 일치하는 경우(또는 제공되지 않은 경우 기본 레지스트리) 해당 스키마의 메타데이터가 반환됩니다. 그렇지 않으면 404 또는 NotFound 오류가 반환됩니다. Deleted 상태의 스키마 버전은 결과에 포함되지 않습니다.

요청

- SchemaId – 필수(Required): SchemaId 객체입니다.

  스키마 자격 증명 필드를 포함하는 래퍼 구조입니다. 구조에는 다음이 포함됩니다.

  - SchemaId$SchemaArn: 스키마의 Amazon 리소스 이름(ARN)입니다. SchemaArn 또는 SchemaName 중 하나가 제공되어야 합니다.
  - SchemaId$SchemaName: 스키마의 이름입니다. SchemaArn 또는 SchemaName 중 하나가 제공되어야 합니다.
- SchemaDefinition – 필수(Required): Custom string pattern #13과(와) 일치하는 1~170,000바이트 길이의 UTF-8 문자열입니다.

  스키마 세부 정보가 필요한 스키마의 정의입니다.

응답

- SchemaVersionId – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 스키마 ID입니다.
- SchemaArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN).
- DataFormat – UTF-8 문자열입니다(유효한 값: AVRO | JSON | PROTOBUF).

  스키마 정의의 데이터 형식입니다. 현재 AVRO, JSON 및 PROTOBUF가 지원됩니다.
- Status – UTF-8 문자열입니다(유효한 값: AVAILABLE | PENDING | FAILURE | DELETING).

  스키마 버전의 상태입니다.
- CreatedTime – UTF-8 문자열입니다.

스키마가 생성된 날짜 및 시간입니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetRegistry 작업(Python: get_registry)

지정된 레지스트리를 자세히 설명합니다.

요청

- `RegistryId` – 필수(Required): RegistryId 객체입니다.

  이는 레지스트리 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.

응답

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 이름입니다.
- `RegistryArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 Amazon 리소스 이름(ARN)입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  레지스트리에 대한 설명입니다.
- `Status` – UTF-8 문자열입니다(유효한 값: `AVAILABLE` | `DELETING`).

  레지스트리의 상태입니다.
- `CreatedTime` – UTF-8 문자열입니다.

레지스트리가 생성된 날짜 및 시간입니다.

- `UpdatedTime` – UTF-8 문자열입니다.

  레지스트리가 업데이트된 날짜 및 시간입니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

PutSchemaVersionMetadata 작업(Python: put_schema_version_metadata)

지정된 스키마 버전 ID에 대한 메타데이터 키 값 페어를 넣습니다. 스키마 버전당 최대 10개의 키 값 페어가 허용됩니다. 하나 이상의 호출을 통해 추가할 수 있습니다.

요청

- `SchemaId` – SchemaId 객체입니다.

  스키마의 고유 ID입니다.

- `SchemaVersionNumber` – SchemaVersionNumber 객체입니다.

  스키마의 버전 번호입니다.

- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 고유한 버전 ID입니다.

- `MetadataKeyValue` – 필수(Required): MetadataKeyValuePair 객체입니다.

  메타데이터 키의 해당하는 값입니다.

응답

- `SchemaArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

스키마의 Amazon 리소스 이름(ARN)입니다.

- `SchemaName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  스키마의 이름입니다.

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 이름입니다.

- `LatestVersion` – 부울입니다.

  스키마의 최신 버전입니다.

- `VersionNumber` - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.

- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 고유한 버전 ID입니다.

- `MetadataKey` – Custom string pattern #14과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  메타데이터 키입니다.

- `MetadataValue` –Custom string pattern #14과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  메타데이터 키의 값입니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`

QuerySchemaVersionMetadata 작업(Python: query_schema_version_metadata)

스키마 버전 메타데이터 정보에 대한 쿼리입니다.

요청

- `SchemaId` – SchemaId 객체입니다.

  스키마 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.

- `SchemaVersionNumber` – SchemaVersionNumber 객체입니다.

  스키마의 버전 번호입니다.

- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 고유한 버전 ID입니다.

- `MetadataList` – MetadataKeyValuePair 객체의 배열입니다.

  메타데이터에 대한 키-값 페어를 검색합니다. 제공되지 않으면 모든 메타데이터 정보를 가져옵니다.

- `MaxResults` – 1~50의 숫자(정수)입니다.

  페이지당 필요한 최대 결과 수입니다. 값이 제공되지 않으면 페이지당 기본값이 25로 설정됩니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰입니다.

응답

- `MetadataInfoMap` – 키-값 페어의 맵 배열입니다.

  각 키는 Custom string pattern #14과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  각 값은 MetadataInfo 객체입니다.

  메타데이터 키 및 관련 값의 맵입니다.

- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

스키마 버전의 고유한 버전 ID입니다.

- `NextToken` – UTF-8 문자열입니다.

  목록의 현재 세그먼트가 마지막이 아니면 반환된 토큰 목록에 페이지를 매기는 지속적인 토큰은 반환됩니다.

오류

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`

RemoveSchemaVersionMetadata 작업(Python: remove_schema_version_metadata)

지정된 스키마 버전 ID에 대한 스키마 버전 메타데이터에서 키 값 페어를 제거합니다.

요청

- `SchemaId` – SchemaId 객체입니다.

  스키마 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.

- `SchemaVersionNumber` – SchemaVersionNumber 객체입니다.

  스키마의 버전 번호입니다.

- `SchemaVersionId` – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전의 고유한 버전 ID입니다.

- `MetadataKeyValue` – 필수(Required): MetadataKeyValuePair 객체입니다.

  메타데이터 키의 값입니다.

응답

- SchemaArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  스키마의 Amazon 리소스 이름(ARN).
- SchemaName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  스키마의 이름입니다.
- RegistryName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  레지스트리의 이름입니다.
- LatestVersion – 부울입니다.

  스키마의 최신 버전입니다.
- VersionNumber - 1~100,000의 숫자(정수)입니다.

  스키마의 버전 번호입니다.
- SchemaVersionId – Custom string pattern #44과(와) 일치하는 36바이트 이상 길이의 UTF-8 문자열입니다.

  스키마 버전에 대한 버전 ID입니다.
- MetadataKey – Custom string pattern #14과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  메타데이터 키입니다.
- MetadataValue –Custom string pattern #14과(와) 일치하는 1~256바이트 길이의 UTF-8 문자열입니다.

  메타데이터 키의 값입니다.

오류

- InvalidInputException
- AccessDeniedException
- EntityNotFoundException

DeleteRegistry 작업(Python: delete_registry)

스키마와 모든 해당 버전을 포함한 전체 레지스트리를 삭제합니다. 삭제 작업의 상태를 얻으려면 비동기 호출 후 `GetRegistry` API를 호출할 수 있습니다. 레지스트리를 삭제하면 `UpdateRegistry`, `CreateSchema`, `UpdateSchema` 및 `RegisterSchemaVersion` API와 같은 레지스트리에 대한 모든 온라인 작업이 비활성화됩니다.

요청

- `RegistryId` – 필수(Required): RegistryId 객체입니다.

  이는 레지스트리 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.

응답

- `RegistryName` – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제되는 레지스트리의 이름입니다.

- `RegistryArn` – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  삭제 중인 레지스트리의 Amazon 리소스 이름(ARN)입니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `AVAILABLE` | `DELETING`).

  레지스트리의 상태입니다. 작업이 성공하면 `Deleting` 상태가 반환됩니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

DeleteSchema 작업(Python: delete_schema)

스키마 집합 및 모든 해당 버전을 포함하여 전체 스키마 집합을 삭제합니다. 삭제 작업의 상태를 얻으려면 비동기 호출 후 GetSchema API를 호출할 수 있습니다. 레지스트리를 삭제하면 GetSchemaByDefinition 및 RegisterSchemaVersion API와 같은 스키마에 대한 모든 온라인 작업이 비활성화됩니다.

요청

- SchemaId – 필수(Required): SchemaId 객체입니다.

  이는 스키마 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.

응답

- SchemaArn – Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  삭제 중인 스키마의 Amazon 리소스 이름(ARN)입니다.
- SchemaName – Custom string pattern #45과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제 중인 스키마의 이름입니다.
- Status – UTF-8 문자열입니다(유효한 값: AVAILABLE | PENDING | DELETING).

  스키마의 상태입니다.

오류

- InvalidInputException
- EntityNotFoundException
- AccessDeniedException
- ConcurrentModificationException

DeleteSchemaVersions 작업(Python: delete_schema_versions)

지정된 스키마에서 버전을 제거합니다. 버전 번호 또는 범위가 제공될 수 있습니다. 호환 모드가 BACKWARDS_FULL과 같이 필요한 버전의 삭제를 금지하는 경우 오류가 반환됩니다. 이 호출 후 `GetSchemaVersions` API를 호출하면 삭제된 버전의 상태가 나열됩니다.

버전 번호 범위에 체크포인트 버전이 포함된 경우 API는 409 충돌을 반환하고 삭제를 진행하지 않습니다. 이 API를 사용하기 전에 먼저 `DeleteSchemaCheckpoint` API를 사용하여 체크포인트를 제거해야 합니다.

`DeleteSchemaVersions` API를 사용하여 스키마 집합의 첫 번째 스키마 버전을 삭제할 수 없습니다. 첫 번째 스키마 버전은 `DeleteSchema` API로만 삭제할 수 있습니다. 이 작업은 스키마 버전 아래에 첨부된 `SchemaVersionMetadata`도 삭제합니다. 영구 삭제는 데이터베이스에 적용됩니다.

호환 모드가 BACKWARDS_FULL과 같이 필요한 버전의 삭제를 금지하는 경우 오류가 반환됩니다.

요청

- `SchemaId` – 필수(Required): SchemaId 객체입니다.

  이는 스키마 이름과 Amazon 리소스 이름(ARN)을 포함할 수 있는 래퍼 구조입니다.
- `Versions` – 필수(Required): Custom string pattern #15과(와) 일치하는 1~100,000바이트 길이의 UTF-8 문자열입니다.

  다음 포맷의 버전 범위가 제공될 수 있습니다.
  - 단일 버전 번호, 5
  - 범위, 5~8 : 버전 5, 6, 7, 8을 삭제합니다.

응답

- `SchemaVersionErrors` – SchemaVersionErrorItem 객체의 배열입니다.

  각각 오류 및 스키마 버전을 포함하는 `SchemaVersionErrorItem` 객체 목록입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`

- `ConcurrentModificationException`

워크플로

워크플로 API는 AWS Glue에서의 워크플로 생성, 업데이트 또는 확인과 관련된 API 및 데이터 유형에 대해 설명합니다. 워크플로 및 작업 실행의 경우 작업 실행 기록을 90일 동안 액세스할 수 있습니다.

데이터 타입

- JobNodeDetails 구조
- CrawlerNodeDetails 구조
- TriggerNodeDetails 구조
- 크롤 구조
- 노드 구조
- 엣지 구조
- 워크플루 구조
- WorkflowGraph 구조
- WorkflowRun 구조
- WorkflowRunStatistics 구조
- StartingEventBatchCondition 구조
- 블루프린트 구조
- BlueprintDetails 구조
- LastActiveDefinition 구조
- BlueprintRun 구조

JobNodeDetails 구조

워크플로에 있는 작업 노드의 세부 정보입니다.

필드

- `JobRuns` – JobRun 객체의 배열입니다.

  작업 노드가 나타내는 작업 실행에 대한 정보입니다.

CrawlerNodeDetails 구조

워크플로에 있는 크롤러 노드의 세부 정보입니다.

필드

- `Crawls` – Crawl 객체의 배열입니다.

  크롤 노드가 나타내는 크롤의 목록입니다.

TriggerNodeDetails 구조

워크플로에 있는 트리거 노드의 세부 정보입니다.

필드

- `Trigger` – 트리거 객체입니다.

  트리거 노드가 나타내는 트리거의 정보입니다.

크롤 구조

워크플로에 있는 크롤의 세부 정보입니다.

필드

- `State` – UTF-8 문자열입니다(유효 값: `RUNNING` | `CANCELLING` | `CANCELLED` | `SUCCEEDED` | `FAILED` | `ERROR`).

  크롤러의 상태입니다.
- `StartedOn` – 타임스탬프입니다.

  크롤이 시작된 날짜와 시간입니다.
- `CompletedOn` – 타임스탬프입니다.

  크롤이 완료된 날짜와 시간입니다.
- `ErrorMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  크롤과 연결된 오류 메시지입니다.

- `LogGroup` – Log group string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  크롤과 연결된 로그 그룹입니다.

- `LogStream` – Log-stream string pattern과(와) 일치하는 1~512바이트 길이의 UTF-8 문자열입니다.

  크롤과 연결된 로그 스트림입니다.

노드 구조

노드는 워크플로 그래프에서 AWS Glue 구성 요소(트리거, 크롤러 또는 작업)를 나타냅니다.

필드

- `Type` – UTF-8 문자열입니다(유효한 값: `CRAWLER` | `JOB` | `TRIGGER`).

  노드가 나타내는 AWS Glue 구성 요소의 유형입니다.

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  노드가 나타내는 AWS Glue 구성 요소의 이름입니다.

- `UniqueId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  워크플로 내의 노드에 할당된 고유 ID입니다.

- `TriggerDetails` – TriggerNodeDetails 객체입니다.

  노드가 트리거를 나타낼 때 트리거의 세부 정보입니다.

- `JobDetails` – JobNodeDetails 객체입니다.

  노드가 작업을 나타낼 때 작업의 세부 정보입니다.

- `CrawlerDetails` – CrawlerNodeDetails 객체입니다.

  노드가 크롤러를 나타낼 때 크롤러의 세부 정보입니다.

엣지 구조

엣지는 엣지가 속한 워크플로의 일부인 두 개의 AWS Glue 구성 요소 간에 방향이 있는 연결을 나타냅니다.

필드

- `SourceId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  엣지가 시작되는 워크플로 내 노드의 고유 ID입니다.
- `DestinationId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  엣지가 종료되는 워크플로 내 노드의 고유 ID입니다.

워크플루 구조

워크플로는 복잡한 ETL 태스크를 완료하기 위해 실행되는 여러 종속 AWS Glue 작업 및 크롤러의 모음입니다. 워크플로는 모든 작업 및 크롤러의 실행 및 모니터링을 관리합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  워크플로의 이름입니다.
- `Description` – UTF-8 문자열입니다.

  워크플로에 대한 설명입니다.
- `DefaultRunProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  워크플로의 각 실행의 일부로 사용할 속성 모음입니다. 실행 속성은 워크플로의 각 작업에서 사용할 수 있습니다. 작업은 흐름의 다음 작업에 대한 속성을 수정할 수 있습니다.
- `CreatedOn` – 타임스탬프입니다.

  워크플로가 생성된 날짜와 시간입니다.
- `LastModifiedOn` – 타임스탬프입니다.

  워크플로가 마지막으로 수정된 날짜와 시간입니다.
- `LastRun` – WorkflowRun 객체입니다.

워크플로의 마지막 실행에 대한 정보입니다.

- `Graph` – WorkflowGraph 객체입니다.

  워크플로에 속하는 모든 AWS Glue 구성 요소를 이들 간의 방향 있는 연결과 노드를 엣지로 나타내는 그래프입니다.

- `CreationStatus` – UTF-8 문자열입니다(유효 값: `CREATING` | `CREATED` | `CREATION_FAILED`).

  워크플로의 생성 상태입니다.

- `MaxConcurrentRuns` - 숫자(정수)입니다.

  이 파라미터를 사용하여 데이터에 대한 원치 않는 다중 업데이트를 방지하거나 비용을 제어하거나 경우에 따라 구성 요소 작업의 최대 동시 실행 수를 초과하는 것을 방지할 수 있습니다. 이 파라미터를 공백으로 두면 동시 워크플로 실행 수에 제한이 없습니다.

- `BlueprintDetails` – BlueprintDetails 객체입니다.

  이 구조는 이 특정 워크플로가 생성된 블루프린트의 세부 정보를 나타냅니다.

WorkflowGraph 구조

워크플로 그래프는 워크플로에 있는 모든 AWS Glue 구성 요소와 그 사이의 모든 방향 있는 연결이 포함된 완전한 워크플로를 나타냅니다.

필드

- `Nodes` – 노드 객체의 배열입니다.

  노드로 나타내는 워크플로에 속한 AWS Glue 구성 요소의 목록입니다.

- `Edges` – Edge 객체의 배열입니다.

  워크플로에 속한 노드 간의 모든 방향 있는 연결의 목록입니다.

WorkflowRun 구조

워크플로 실행은 모든 실행 시간 정보를 제공하는 워크플로의 실행입니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

실행된 워크플로의 이름입니다.

- `WorkflowRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 워크플로 실행의 ID입니다.

- `PreviousRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이전 워크플로 실행의 ID입니다.

- `WorkflowRunProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  실행 중에 설정된 워크플로 실행 속성입니다.

- `StartedOn` – 타임스탬프입니다.

  워크플로 실행이 시작된 날짜와 시간입니다.

- `CompletedOn` – 타임스탬프입니다.

  워크플로 실행이 완료된 날짜와 시간입니다.

- `Status` – UTF-8 문자열입니다(유효 값: `RUNNING` | `COMPLETED` | `STOPPING` | `STOPPED` | `ERROR`).

  워크플로 실행의 상태입니다.

- `ErrorMessage` – UTF-8 문자열입니다.

  이 오류 메시지는 워크플로 실행을 시작할 때 발생할 수 있는 모든 오류에 대해 설명합니다. 현재 유일한 오류 메시지는 "워크플로에 대한 동시 실행 초과(Concurrent runs exceeded for workflow): `foo`"입니다.

- `Statistics` – WorkflowRunStatistics 객체입니다.

  실행의 통계입니다.

- `Graph` – WorkflowGraph 객체입니다.

  워크플로에 속하는 모든 AWS Glue 구성 요소를 이들 간의 방향 있는 연결과 노드를 엣지로 나타내는 그래프입니다.

- `StartingEventBatchCondition` – StartingEventBatchCondition 객체입니다.

  워크플로 실행을 시작한 배치 조건입니다.

WorkflowRunStatistics 구조

워크플로 실행 통계는 워크플로 실행에 대한 통계를 제공합니다.

필드

- `TotalActions` - 숫자(정수)입니다.

  워크플로 실행에 있는 총 작업 수입니다.
- `TimeoutActions` - 숫자(정수)입니다.

  시간 초과된 총 작업 수입니다.
- `FailedActions` - 숫자(정수)입니다.

  실패한 총 작업 수입니다.
- `StoppedActions` - 숫자(정수)입니다.

  중지된 총 작업 수입니다.
- `SucceededActions` - 숫자(정수)입니다.

  성공한 총 작업 수입니다.
- `RunningActions` - 숫자(정수)입니다.

  실행 상태의 총 작업 수입니다.
- `ErroredActions` - 숫자(정수)입니다.

  워크플로 실행에서 ERROR 상태의 작업 실행 수를 나타냅니다.
- `WaitingActions` - 숫자(정수)입니다.

  워크플로 실행에서 WAITING 상태의 작업 실행 수를 나타냅니다.

StartingEventBatchCondition 구조

워크플로 실행을 시작한 배치 조건입니다. BatchSize 멤버가 0이 아닌 경우 배치 크기의 이벤트 수가 도착했거나 BatchWindow 멤버가 0이 아닌 경우 배치 기간이 만료되었습니다.

필드

- `BatchSize` - 숫자(정수)입니다.

  배치의 이벤트 수입니다.
- `BatchWindow` - 숫자(정수)입니다.

  배치 기간의 기간(초)입니다.

블루프린트 구조

블루프린트의 세부 정보입니다.

필드

- `Name` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.
- `Description` – 1~512바이트 길이의 UTF-8 문자열입니다.

  블루프린트에 대한 설명입니다.
- `CreatedOn` – 타임스탬프입니다.

  블루프린트가 등록된 날짜 및 시간입니다.
- `LastModifiedOn` – 타임스탬프입니다.

  블루프린트가 마지막으로 수정된 날짜 및 시간입니다.
- `ParameterSpec` – 1~131,072바이트 길이의 UTF-8 문자열입니다.

  블루프린트에 대한 파라미터 사양 목록을 나타내는 JSON 문자열입니다.
- `BlueprintLocation` – UTF-8 문자열입니다.

  블루프린트가 게시되는 Amazon S3의 경로를 지정합니다.
- `BlueprintServiceLocation` – UTF-8 문자열입니다.

CreateBlueprint/UpdateBlueprint를 호출하여 AWS Glue에 블루프린트를 등록할 때 블루프린트가 복사되는 Amazon S3의 경로를 지정합니다.

- Status – UTF-8 문자열입니다(유효 값: CREATING | ACTIVE | UPDATING | FAILED).

  블루프린트 등록의 상태입니다.

  - [생성(Creating)] - 블루프린트 등록이 진행 중입니다.
  - [활성(Active)] - 블루프린트가 성공적으로 등록되었습니다.
  - [업데이트 중(Updating)] - 블루프린트 등록에 대한 업데이트가 진행 중입니다.
  - [실패(Failed)] - 블루프린트 등록에 실패했습니다.

- ErrorMessage – UTF-8 문자열입니다.

  오류 메시지입니다.

- LastActiveDefinition – LastActiveDefinition 객체입니다.

  블루프린트의 여러 버전이 있고 최신 버전에 일부 오류가 있는 경우 이 속성은 서비스에서 사용할 수 있는 마지막으로 성공한 블루프린트 정의를 나타냅니다.

BlueprintDetails 구조

블루프린트의 세부 정보입니다.

필드

- BlueprintName – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.

- RunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 블루프린트의 실행 ID입니다.

LastActiveDefinition 구조

블루프린트의 여러 버전이 있고 최신 버전에 일부 오류가 있는 경우 이 속성은 서비스에서 사용할 수 있는 마지막으로 성공한 블루프린트 정의를 나타냅니다.

필드

- `Description` – 1~512바이트 길이의 UTF-8 문자열입니다.

  블루프린트에 대한 설명입니다.
- `LastModifiedOn` – 타임스탬프입니다.

  블루프린트가 마지막으로 수정된 날짜 및 시간입니다.
- `ParameterSpec` – 1~131,072바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 파라미터를 지정하는 JSON 문자열입니다.
- `BlueprintLocation` – UTF-8 문자열입니다.

  AWS Glue 개발자가 블루프린트를 게시하는 Amazon S3의 경로를 지정합니다.
- `BlueprintServiceLocation` – UTF-8 문자열입니다.

  블루프린트를 생성하거나 업데이트할 때 블루프린트가 복사되는 Amazon S3의 경로를 지정합니다.

BlueprintRun 구조

블루프린트 실행의 세부 정보입니다.

필드

- `BlueprintName` – Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.
- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 블루프린트 실행의 실행 ID입니다.
- `WorkflowName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  성공적인 블루프린트 실행의 결과로 생성된 워크플로의 이름입니다. 블루프린트 실행에 오류가 있으면 워크플로가 생성되지 않습니다.
- `State` – UTF-8 문자열입니다(유효 값: `RUNNING` | `SUCCEEDED` | `FAILED` | `ROLLING_BACK`).

  블루프린트 실행 상태입니다. 가능한 값은 다음과 같습니다.

  - [실행 중(Running)] - 블루프린트 실행이 진행 중입니다.
  - [성공(Succeeded)] - 블루프린트 실행이 성공적으로 완료되었습니다.
  - [실패(Failed)] - 블루프린트 실행이 실패하고 롤백이 완료되었습니다.
  - [롤백 중(Rolling Back)] - 블루프린트 실행이 실패하여 롤백이 진행 중입니다.
- `StartedOn` – 타임스탬프입니다.

  블루프린트 실행이 시작된 날짜 및 시간입니다.
- `CompletedOn` – 타임스탬프입니다.

  블루프린트 실행이 완료된 날짜 및 시간입니다.
- `ErrorMessage` – UTF-8 문자열입니다.

  블루프린트를 실행하는 동안 표시되는 모든 오류를 나타냅니다.
- `RollbackErrorMessage` – UTF-8 문자열입니다.

  워크플로의 엔터티를 생성하는 동안 오류가 발생하면 해당 시점까지 생성된 엔터티를 롤백하고 삭제하려고 합니다. 이 속성은 생성된 엔터티를 삭제하는 동안 발생한 오류를 나타냅니다.
- `Parameters` – 1~131,072바이트 길이의 UTF-8 문자열입니다.

  블루프린트 파라미터(문자열). `Blueprint$ParameterSpec`에 정의된 파라미터 사양에서 필요한 각 키에 대한 값을 제공해야 합니다.
- `RoleArn` – Custom string pattern #30과(와) 일치하는 1~1,024바이트 길이의 UTF-8 문자열입니다.

  역할 ARN입니다. 이 역할은 AWS Glue 서비스에서 수임하며 워크플로 및 워크플로의 기타 엔터티를 생성하는 데 사용됩니다.

운영

- CreateWorkflow 작업(Python: create_workflow)
- UpdateWorkflow 작업(Python: update_workflow)
- DeleteWorkflow 작업(Python: delete_workflow)
- GetWorkflow 작업(Python: get_workflow)
- ListWorkflows 작업(Python: list_workflows)
- BatchGetWorkflows 작업(Python: batch_get_workflows)
- GetWorkflowRun 작업(Python: get_workflow_run)

- GetWorkflowRuns 작업(Python: get_workflow_runs)
- GetWorkflowRunProperties 작업(Python: get_workflow_run_properties)
- PutWorkflowRunProperties 작업(Python: put_workflow_run_properties)
- CreateBlueprint 작업(Python: create_blueprint)
- UpdateBlueprint 작업(Python: update_blueprint)
- DeleteBlueprint 작업(Python: delete_blueprint)
- ListBlueprints 작업(Python: list_blueprints)
- BatchGetBlueprints 작업(Python: batch_get_blueprints)
- StartBlueprintRun 작업(Python: start_blueprint_run)
- GetBlueprintRun 작업(Python: get_blueprint_run)
- GetBlueprintRuns 작업(Python: get_blueprint_runs)
- StartWorkflowRun 작업(Python: start_workflow_run)
- StopWorkflowRun 작업(Python: stop_workflow_run)
- ResumeWorkflowRun 작업(Python: resume_workflow_run)

CreateWorkflow 작업(Python: create_workflow)

새 워크플로를 생성합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  워크플로에 할당할 이름입니다. 이름은 계정 내에서 고유해야 합니다.
- `Description` – UTF-8 문자열(120,000바이트 이하).

  워크플로에 대한 설명입니다.
- `DefaultRunProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  워크플로의 각 실행의 일부로 사용할 속성 모음입니다.

실행 속성이 로깅될 수 있습니다. 일반 텍스트 보안 암호를 속성으로 전달하지 마세요. 워크플로우 실행 내에서 보안 암호를 사용하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이 워크플로에 사용할 태그입니다.

- `MaxConcurrentRuns` - 숫자(정수)입니다.

  이 파라미터를 사용하여 데이터에 대한 원치 않는 다중 업데이트를 방지하거나 비용을 제어하거나 경우에 따라 구성 요소 작업의 최대 동시 실행 수를 초과하는 것을 방지할 수 있습니다. 이 파라미터를 공백으로 두면 동시 워크플로 실행 수에 제한이 없습니다.

응답

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  요청의 일부로 제공된 워크플로의 이름입니다.

오류

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

UpdateWorkflow 작업(Python: update_workflow)

기존 워크플로를 업데이트합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트할 워크플로의 이름입니다.

- Description – UTF-8 문자열(120,000바이트 이하).

  워크플로에 대한 설명입니다.

- DefaultRunProperties – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  워크플로의 각 실행의 일부로 사용할 속성 모음입니다.

  실행 속성이 로깅될 수 있습니다. 일반 텍스트 보안 암호를 속성으로 전달하지 마세요. 워크플로우 실행 내에서 보안 암호를 사용하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

- MaxConcurrentRuns - 숫자(정수)입니다.

  이 파라미터를 사용하여 데이터에 대한 원치 않는 다중 업데이트를 방지하거나 비용을 제어하거나 경우에 따라 구성 요소 작업의 최대 동시 실행 수를 초과하는 것을 방지할 수 있습니다. 이 파라미터를 공백으로 두면 동시 워크플로 실행 수에 제한이 없습니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  입력에서 지정된 워크플로의 이름입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

- ConcurrentModificationException

DeleteWorkflow 작업(Python: delete_workflow)

워크플로를 삭제합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 워크플로의 이름입니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  입력에서 지정된 워크플로의 이름입니다.

오류

- InvalidInputException
- InternalServiceException
- OperationTimeoutException
- ConcurrentModificationException

GetWorkflow 작업(Python: get_workflow)

워크플로에 대한 리소스 메타데이터를 검색합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색할 워크플로의 이름입니다.

- IncludeGraph – 부울입니다.

워크플로 리소스 메타데이터를 반환할 때 그래프를 포함할지 여부를 지정합니다.

응답

- `Workflow` – 워크플로 객체입니다.

  워크플로에 대한 리소스 메타데이터입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

ListWorkflows 작업(Python: list_workflows)

계정에서 생성된 워크플로의 이름을 나열합니다.

요청

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.
- `MaxResults` - 1 이상 25 이하의 숫자(정수)입니다.

  반환할 목록의 최대 크기.

응답

- `Workflows` – 1~25개 문자열로 구성된 UTF-8 문자열의 배열입니다.

  계정에 있는 워크플로의 이름 목록입니다.
- `NextToken` – UTF-8 문자열입니다.

  일부 워크플로 이름이 반환되지 않은 경우의 지속 토큰입니다.

오류

- InvalidInputException
- InternalServiceException
- OperationTimeoutException

BatchGetWorkflows 작업(Python: batch_get_workflows)

제공된 워크플로 이름 목록에 대한 리소스 메타데이터의 목록을 반환합니다. ListWorkflows 작업을 호출한 후에는 권한이 부여된 데이터에 액세스하기 위해 이 작업을 호출할 수 있습니다. 이 작업은 태그를 사용하는 권한 조건을 포함해 모든 IAM 권한을 지원합니다.

요청

- Names – 필수(Required): 1~25개 문자열의 UTF-8 문자열의 배열입니다.

  ListWorkflows 작업에서 반환된 이름일 수 있는 워크플로 이름의 목록입니다.
- IncludeGraph – 부울입니다.

  워크플로 리소스 메타데이터를 반환할 때 그래프를 포함할지 여부를 지정합니다.

응답

- Workflows – 워크플로 객체의 배열이며 구조는 1~25개입니다.

  워크플로 리소스 메타데이터의 목록입니다.
- MissingWorkflows – 1~25개 문자열로 구성된 UTF-8 문자열의 배열입니다.

  찾을 수 없는 워크플로의 이름 목록입니다.

오류

- InternalServiceException
- OperationTimeoutException
- InvalidInputException

GetWorkflowRun 작업(Python: get_workflow_run)

제공된 워크플로 실행에 대한 메타데이터를 검색합니다. 워크플로 및 작업 실행의 경우 작업 실행 기록을 90일 동안 액세스할 수 있습니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행되고 있는 워크플로의 이름입니다.

- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  워크플로 실행의 ID입니다.

- IncludeGraph – 부울입니다.

  워크플로 그래프를 응답에 포함할지 여부를 지정합니다.

응답

- Run – WorkflowRun 객체입니다.

  요청된 워크플로 실행 메타데이터입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

GetWorkflowRuns 작업(Python: get_workflow_runs)

제공된 워크플로의 모든 실행에 대한 메타데이터를 검색합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행 메타데이터를 반환해야 하는 워크플로의 이름입니다.

- IncludeGraph – 부울입니다.

  워크플로 그래프를 응답에 포함할지 여부를 지정합니다.

- NextToken – UTF-8 문자열입니다.

  응답의 최대 크기입니다.

- MaxResults – 1~1,000의 숫자(정수)입니다.

  응답에 포함할 최대 워크플로 실행 수입니다.

응답

- Runs – WorkflowRun 객체의 배열이며 구조는 1~1,000개입니다.

  워크플로 실행 메타데이터 객체의 목록입니다.

- NextToken – UTF-8 문자열입니다.

  일부 요청된 워크플로 실행이 반환되지 않은 경우 지속 토큰입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

GetWorkflowRunProperties 작업(Python: get_workflow_run_properties)

실행 중에 설정된 워크플로 실행 속성을 검색합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행된 워크플로의 이름입니다.
- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행 속성을 반환해야 하는 워크플로 실행의 ID입니다.

응답

- RunProperties – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  지정된 실행 중에 설정된 워크플로 실행 속성입니다.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

PutWorkflowRunProperties 작업(Python: put_workflow_run_properties)

제공된 워크플로 실행에 대해 지정된 워크플로 실행 속성을 입력합니다. 지정된 실행에 대한 속성이 이미 있으면 이 속성이 값을 재정의하고 그렇지 않으면 속성을 기존 속성에 추가합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행된 워크플로의 이름입니다.

- `RunId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행 속성을 업데이트해야 하는 워크플로 실행의 ID입니다.

- `RunProperties` – 필수(Required): 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  지정된 실행에 대해 입력할 속성입니다.

  실행 속성이 로깅될 수 있습니다. 일반 텍스트 보안 암호를 속성으로 전달하지 마세요. 워크플로우 실행 내에서 보안 암호를 사용하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

응답

- 무응답 파라미터.

오류

- `AlreadyExistsException`
- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

CreateBlueprint 작업(Python: create_blueprint)

AWS Glue에 블루프린트를 등록합니다.

요청

- Name – 필수(Required): Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.

- Description – 1~512바이트 길이의 UTF-8 문자열입니다.

  블루프린트에 대한 설명입니다.

- BlueprintLocation – 필수(Required): Custom string pattern #32과(와) 일치하는 1~8,192바이트 길이의 UTF-8 문자열입니다.

  블루프린트가 게시되는 Amazon S3의 경로를 지정합니다.

- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이 블루프린트에 적용할 태그입니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  등록된 블루프린트의 이름을 반환합니다.

오류

- AlreadyExistsException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException
- ResourceNumberLimitExceededException

UpdateBlueprint 작업(Python: update_blueprint)

등록된 블루프린트를 업데이트합니다.

요청

- Name – 필수(Required): Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.

- Description – 1~512바이트 길이의 UTF-8 문자열입니다.

  블루프린트에 대한 설명입니다.

- BlueprintLocation – 필수(Required): Custom string pattern #32과(와) 일치하는 1~8,192바이트 길이의 UTF-8 문자열입니다.

  블루프린트가 게시되는 Amazon S3의 경로를 지정합니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트된 블루프린트의 이름을 반환합니다.

오류

- EntityNotFoundException
- ConcurrentModificationException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException
- IllegalBlueprintStateException

DeleteBlueprint 작업(Python: delete_blueprint)

기존 블루프린트를 삭제합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 블루프린트의 이름입니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제된 블루프린트의 이름을 반환합니다.

오류

- InvalidInputException
- OperationTimeoutException
- InternalServiceException

ListBlueprints 작업(Python: list_blueprints)

계정의 모든 블루프린트 이름을 나열합니다.

요청

- NextToken – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.
- MaxResults - 1 이상 25 이하의 숫자(정수)입니다.

  반환할 목록의 최대 크기.
- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  AWS 리소스 태그로 목록을 필터링합니다.

응답

- Blueprints – UTF-8 문자열의 배열입니다.

  계정에 있는 블루프린트의 이름 목록입니다.
- NextToken – UTF-8 문자열입니다.

  모든 블루프린트 이름이 반환하지 않은 경우의 지속 토큰입니다.

오류

- InvalidInputException
- InternalServiceException
- OperationTimeoutException

BatchGetBlueprints 작업(Python: batch_get_blueprints)

블루프린트 목록에 대한 정보를 검색합니다.

요청

- Names – 필수(Required): 1~25개 문자열의 UTF-8 문자열의 배열입니다.

  블루프린트 이름의 목록입니다.
- IncludeBlueprint – 부울입니다.

  응답에 워크플로를 포함할지 여부를 지정합니다.
- IncludeParameterSpec – 부울입니다.

  응답에 블루프린트에 대한 파라미터를 JSON 문자열로 포함할지 여부를 지정합니다.

응답

- Blueprints – 블루프린트 객체의 배열입니다.

  블루프린트 목록을 Blueprints 객체로 반환합니다.
- MissingBlueprints – UTF-8 문자열의 배열입니다.

찾을 수 없는 BlueprintNames 목록을 반환합니다.

오류

- InternalServiceException
- OperationTimeoutException
- InvalidInputException

StartBlueprintRun 작업(Python: start_blueprint_run)

지정된 블루프린트의 새 실행을 시작합니다.

요청

- BlueprintName – 필수(Required): Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.

- Parameters – 1~131,072바이트 길이의 UTF-8 문자열입니다.

  파라미터를 BlueprintParameters 객체로 지정합니다.

- RoleArn – 필수(Required): Custom string pattern #30과(와) 일치하는 1~1,024바이트 길이의 UTF-8 문자열입니다.

  워크플로 생성에 사용되는 IAM 역할을 지정합니다.

응답

- RunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 블루프린트 실행의 실행 ID입니다.

오류

- InvalidInputException
- OperationTimeoutException

- InternalServiceException
- ResourceNumberLimitExceededException
- EntityNotFoundException
- IllegalBlueprintStateException

GetBlueprintRun 작업(Python: get_blueprint_run)

블루프린트 실행의 세부 정보를 검색합니다.

요청

- BlueprintName – 필수(Required): Custom string pattern #31과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.

- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색하려는 블루프린트 실행의 실행 ID입니다.

응답

- BlueprintRun – BlueprintRun 객체입니다.

  BlueprintRun 객체를 반환합니다.

오류

- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException

GetBlueprintRuns 작업(Python: get_blueprint_runs)

지정된 블루프린트에 대한 블루프린트 실행의 세부 정보를 검색합니다.

요청

- `BlueprintName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  블루프린트의 이름입니다.

- `NextToken` – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 최대 크기.

응답

- `BlueprintRuns` – BlueprintRun 객체의 배열입니다.

  `BlueprintRun` 객체 목록을 반환합니다.

- `NextToken` – UTF-8 문자열입니다.

  모든 블루프린트 실행이 반환하지 않은 경우의 지속 토큰입니다.

오류

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

StartWorkflowRun 작업(Python: start_workflow_run)

지정된 워크플로의 새 실행을 시작합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

시작할 워크플로의 이름입니다.

- `RunProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 UTF-8 문자열입니다.

  새 워크플로 실행에 대한 워크플로 실행 속성입니다.

  실행 속성이 로깅될 수 있습니다. 일반 텍스트 보안 암호를 속성으로 전달하지 마세요. 워크플로우 실행 내에서 보안 암호를 사용하려는 경우 AWS Glue 연결, AWS Secrets Manager 또는 다른 보안 암호 관리 메커니즘에서 검색합니다.

응답

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  새 실행의 ID입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

StopWorkflowRun 작업(Python: stop_workflow_run)

지정된 워크플로 실행의 실행을 중지합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

중지할 워크플로의 이름입니다.

- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  중지할 워크플로 실행의 ID입니다.

응답

- 무응답 파라미터.

오류

- InvalidInputException
- EntityNotFoundException
- InternalServiceException
- OperationTimeoutException
- IllegalWorkflowStateException

ResumeWorkflowRun 작업(Python: resume_workflow_run)

이전에 부분적으로 완료된 워크플로 실행의 선택한 노드를 다시 시작하고 워크플로 실행을 재개합니다. 선택한 노드와 선택한 노드의 다운스트림에 있는 모든 노드가 실행됩니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  재개할 워크플로의 이름입니다.

- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  재개할 워크플로 실행의 ID입니다.

- NodeIds – 필수(Required): UTF-8 문자열의 배열입니다.

다시 시작하려는 노드의 노드 ID 목록입니다. 다시 시작될 노드에는 원래 실행에서 실행 시도가 있어야 합니다.

응답

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  재개된 워크플로 실행에 할당된 새 ID입니다. 워크플로 실행의 각 재개에는 새 실행 ID가 있습니다.
- `NodeIds` – UTF-8 문자열의 배열입니다.

  실제로 다시 시작된 노드의 노드 ID 목록입니다.

오류

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentRunsExceededException`
- `IllegalWorkflowStateException`

사용 프로필

사용 프로필 API는 AWS Glue에서의 사용 프로필 생성, 업데이트 또는 확인과 관련된 API 및 데이터 유형에 대해 설명합니다.

데이터 타입

- ProfileConfiguration 구조
- ConfigurationObject 구조
- UsageProfileDefinition 구조

ProfileConfiguration 구조

관리자가 AWS Glue 사용 프로필에서 구성하는 작업 및 세션 값을 지정합니다.

필드

- `SessionConfiguration` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 ConfigurationObject 객체입니다.

  AWS Glue 세션에 대한 구성 파라미터의 키 값 맵.
- `JobConfiguration` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 ConfigurationObject 객체입니다.

  AWS Glue 작업에 대한 구성 파라미터의 키 값 맵.

ConfigurationObject 구조

관리자가 AWS Glue 사용 프로필에 구성된 각 작업 또는 세션 파라미터에 대해 설정하는 값을 지정합니다.

필드

- `DefaultValue` – Custom string pattern #35과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  파라미터의 기본값.
- `AllowedValues` – UTF-8 문자열의 배열입니다.

  파라미터에 허용되는 값 목록.
- `MinValue` – Custom string pattern #35과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  파라미터에 허용되는 최솟값.
- `MaxValue` – Custom string pattern #35과(와) 일치하는 1~128바이트 길이의 UTF-8 문자열입니다.

  파라미터에 허용되는 최댓값.

UsageProfileDefinition 구조

AWS Glue 사용 프로필을 설명합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용 프로필의 이름.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  사용 프로필에 대한 설명.
- `CreatedOn` – 타임스탬프입니다.

  사용 프로필을 생성한 날짜와 시간.
- `LastModifiedOn` – 타임스탬프입니다.

  사용 프로필을 마지막으로 수정한 날짜 및 시간.

운영

- CreateUsageProfile 작업(Python: create_usage_profile)
- GetUsageProfile 작업(Python: get_usage_profile)
- UpdateUsageProfile 작업(Python: update_usage_profile)
- DeleteUsageProfile 작업(Python: delete_usage_profile)
- ListUsageProfiles 작업(Python: list_usage_profiles)

CreateUsageProfile 작업(Python: create_usage_profile)

AWS Glue 사용 프로필을 생성합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용 프로필의 이름.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  사용 프로필에 대한 설명.
- `Configuration` – 필수(Required): ProfileConfiguration 객체입니다.

  프로필의 작업 및 세션 값을 지정하는 `ProfileConfiguration` 객체.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  사용 프로필에 적용된 태그 목록.

응답

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  생성한 사용 프로필의 이름.

오류

- `InvalidInputException`
- `InternalServiceException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `OperationNotSupportedException`

GetUsageProfile 작업(Python: get_usage_profile)

지정된 AWS Glue 사용 프로필에 대한 정보를 검색합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

검색할 사용 프로필의 이름.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용 프로필의 이름.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  사용 프로필에 대한 설명.

- Configuration – ProfileConfiguration 객체입니다.

  프로필의 작업 및 세션 값을 지정하는 ProfileConfiguration 객체.

- CreatedOn – 타임스탬프입니다.

  사용 프로필을 생성한 날짜와 시간.

- LastModifiedOn – 타임스탬프입니다.

  사용 프로필을 마지막으로 수정한 날짜 및 시간.

오류

- InvalidInputException
- InternalServiceException
- EntityNotFoundException
- OperationTimeoutException
- OperationNotSupportedException

UpdateUsageProfile 작업(Python: update_usage_profile)

AWS Glue 사용 프로필을 업데이트합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용 프로필의 이름.
- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  사용 프로필에 대한 설명.
- Configuration – 필수(Required): ProfileConfiguration 객체입니다.

  프로필의 작업 및 세션 값을 지정하는 ProfileConfiguration 객체.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트한 사용 프로필의 이름.

오류

- InvalidInputException
- InternalServiceException
- EntityNotFoundException
- OperationTimeoutException
- OperationNotSupportedException
- ConcurrentModificationException

DeleteUsageProfile 작업(Python: delete_usage_profile)

지정된 AWS Glue 사용 프로필을 삭제합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

삭제할 사용 프로필의 이름.

응답

- 무응답 파라미터.

오류

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `OperationNotSupportedException`

ListUsageProfiles 작업(Python: list_usage_profiles)

모든 AWS Glue 사용 프로필을 나열합니다.

요청

- `NextToken` – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  이것이 지속적으로 호출되면 지속적인 토큰을 포함합니다.
- `MaxResults` - 1 이상 200 이하의 숫자(정수)입니다.

  한 번의 응답으로 반환할 최대 사용 프로필 수.

응답

- `Profiles` – UsageProfileDefinition 객체의 배열입니다.

  사용 프로필(`UsageProfileDefinition`) 객체 목록.
- `NextToken` – 400,000바이트 이하 길이의 UTF-8 문자열입니다.

  현재 목록 부분이 유지가 되지 않으면 연속 토큰이 존재합니다.

오류

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `OperationNotSupportedException`

기계 학습 API

기계 학습 API는 기계 학습 데이터 형식을 설명하며 변환을 생성, 삭제 또는 업데이트하거나 기계 학습 작업 실행을 시작하기 위한 API를 포함합니다.

데이터 타입

- TransformParameters 구조
- EvaluationMetrics 구조
- MLTransform 구조
- FindMatchesParameters 구조
- FindMatchesMetrics 구조
- ConfusionMatrix 구조
- GlueTable 구조
- TaskRun 구조
- TransformFilterCriteria 구조
- TransformSortCriteria 구조
- TaskRunFilterCriteria 구조
- TaskRunSortCriteria 구조
- TaskRunProperties 구조
- FindMatchesTaskRunProperties 구조
- ImportLabelsTaskRunProperties 구조
- ExportLabelsTaskRunProperties 구조
- LabelingSetGenerationTaskRunProperties 구조
- SchemaColumn 구조
- TransformEncryption 구조

- MLUserDataEncryption 구조
- ColumnImportance 구조

TransformParameters 구조

기계 학습 변환과 연결된 알고리즘별 파라미터입니다.

필드

- `TransformType` – 필수: UTF-8 문자열입니다(유효한 값: `FIND_MATCHES`).

  기계 학습 변환의 유형입니다.

  기계 학습 변환 유형에 대한 자세한 내용은 기계 학습 변환 생성을 참조하십시오.
- `FindMatchesParameters` – FindMatchesParameters 객체입니다.

  일치 항목 찾기 알고리즘에 대한 파라미터입니다.

EvaluationMetrics 구조

평가 지표는 기계 학습 변환의 예상 품질을 제공합니다.

필드

- `TransformType` – 필수: UTF-8 문자열입니다(유효한 값: `FIND_MATCHES`).

  기계 학습 변환의 유형입니다.
- `FindMatchesMetrics` – FindMatchesMetrics 객체입니다.

  일치 항목 찾기 알고리즘에 대한 평가 지표입니다.

MLTransform 구조

기계 학습 변환의 구조입니다.

필드

- `TransformId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

기계 학습 변환에 대해 생성된 고유 변환 ID입니다. ID는 고유한 것으로 보장되며 변경되지 않습니다.

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 사용자 정의 이름입니다. 이름은 고유한 것으로 보장되지 않으며 언제든지 변경할 수 있습니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  기계 학습 변환에 대한 사용자 정의된 긴 형식의 설명 텍스트입니다. 설명은 고유한 것으로 보장되지 않으며 언제든지 변경할 수 있습니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `NOT_READY` | `READY` | `DELETING`).

  기계 학습 변환의 현재 상태입니다.

- `CreatedOn` – 타임스탬프입니다.

  타임스탬프입니다. 이 기계 학습 변환이 생성된 시간과 날짜입니다.

- `LastModifiedOn` – 타임스탬프입니다.

  타임스탬프입니다. 이 기계 학습 변환이 수정된 마지막 시점입니다.

- `InputRecordTables` – GlueTable 객체의 배열이며 구조는 10개 이하입니다.

  변환에 사용된 AWS Glue 테이블 정의의 목록입니다.

- `Parameters` – TransformParameters 객체입니다.

  `TransformParameters` 객체입니다. 파라미터를 통해 기계 학습 변환이 학습하는 데이터 및 다양한 트레이드오프(예: 재현율 대비 귀중함 또는 비용 대비 정확도)에 대한 기본 설정을 지정하여 기계 학습 변환의 동작을 튜닝(사용자 지정)할 수 있습니다.

- `EvaluationMetrics` – EvaluationMetrics 객체입니다.

  `EvaluationMetrics` 객체입니다. 평가 지표는 기계 학습 변환의 예상 품질을 제공합니다.

- `LabelCount` - 숫자(정수)입니다.

  이 변환에 대해 AWS Glue에서 생성된 레이블 지정 파일에 대한 카운트 식별자입니다. 더 좋은 변환을 만들면 레이블 지정 파일을 반복적으로 다운로드하고 레이블 지정하고 업로드할 수 있습니다.

- `Schema` – SchemaColumn 객체의 배열이며 구조는 100개 이하입니다.

이 변환이 실행할 수 있는 열과 데이터 형식을 나타내는 키-값 페어의 맵입니다. 100열의 상한이 있습니다.

- `Role` – UTF-8 문자열입니다.

  필수 권한이 있는 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다. 필요한 권한에는 AWS Glue 리소스에 대한 AWS Glue 서비스 역할 권한과 변환에서 요구하는 Amazon S3 권한 모두가 포함됩니다.

  - 이 역할에는 AWS Glue의 리소스에 대한 액세스 허용을 위한 AWS Glue 서비스 역할 권한이 필요합니다. AWS Glue에 액세스하는 IAM 사용자에게 정책 연결을 참조하세요.
  - 이 역할에는 작업 실행에서 이 변환에 사용되는 Amazon Simple Storage Service(Amazon S3) 소스, 대상, 임시 디렉터리, 스크립트 및 모든 라이브러리에 대한 권한이 필요합니다.

- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 값은 이러한 기계 학습 변환과 호환되는 AWS Glue의 버전을 결정합니다. Glue 1.0은 대부분의 고객에게 권장됩니다. 값이 설정되지 않은 경우 Glue 호환성은 Glue 0.9로 기본 설정됩니다. 자세한 내용은 개발자 안내서의 AWS Glue 버전을 참조하세요.

- `MaxCapacity` - 숫자(double)입니다.

  이 변환의 태스크 실행에 할당된 AWS Glue 데이터 처리 장치(DPU)의 수입니다. 2~100DPU를 할당할 수 있으며, 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  `MaxCapacity`는 `NumberOfWorkers` 및 `WorkerType`과 함께 사용할 수 없는 옵션입니다.

  - `NumberOfWorkers`나 `WorkerType` 중 하나가 설정되면 `MaxCapacity`를 설정할 수 없습니다.
  - `MaxCapacity`가 설정되면 `NumberOfWorkers`와 `WorkerType` 모두 설정할 수 없습니다.
  - `WorkerType`이 설정되면 `NumberOfWorkers`가 필요합니다(반대의 경우도 마찬가지).
  - `MaxCapacity`와 `NumberOfWorkers`는 1 이상이어야 합니다.

  `WorkerType` 필드를 `Standard` 이외의 다른 값으로 설정하면 `MaxCapacity` 필드가 자동으로 설정되고 읽기 전용이 됩니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  이 변환의 작업이 실행될 때 할당되는 미리 정의된 작업자의 유형입니다. Standard, G.1X 또는 G.2X 값을 허용합니다.

  - `Standard` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 50GB 디스크와, 작업자당 실행기 2개를 제공합니다.
  - `G.1X` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 64GB 디스크와, 작업자당 실행기 1개를 제공합니다.
  - `G.2X` 작업자 유형의 경우, 각 작업자가 8vCPU, 32GB 메모리 및 128GB 디스크와, 작업자당 실행기 1개를 제공합니다.

  `MaxCapacity`는 `NumberOfWorkers` 및 `WorkerType`과 함께 사용할 수 없는 옵션입니다.

  - `NumberOfWorkers`나 `WorkerType` 중 하나가 설정되면 `MaxCapacity`를 설정할 수 없습니다.
  - `MaxCapacity`가 설정되면 `NumberOfWorkers`와 `WorkerType` 모두 설정할 수 없습니다.
  - `WorkerType`이 설정되면 `NumberOfWorkers`가 필요합니다(반대의 경우도 마찬가지).
  - `MaxCapacity`와 `NumberOfWorkers`는 1 이상이어야 합니다.
- `NumberOfWorkers` - 숫자(정수)입니다.

  변환의 작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.

  `WorkerType`이 설정되면 `NumberOfWorkers`가 필요합니다(반대의 경우도 마찬가지).
- `Timeout` – 1 이상의 숫자(정수)입니다.

  기계 학습 변환의 시간 초과(분)입니다.
- `MaxRetries` - 숫자(정수)입니다.

  기계 학습 변환의 `MLTaskRun`이 실패한 후 최대 재시도 횟수입니다.
- `TransformEncryption` – TransformEncryption 객체입니다.

  사용자 데이터 액세스에 적용되는 변환의 유휴 시 암호화 설정입니다. 기계 학습 변환에서는 KMS를 사용하여 Amazon S3의 암호화된 사용자 데이터에 액세스할 수 있습니다.

FindMatchesParameters 구조

일치 항목 찾기 변환을 구성하기 위한 파라미터입니다.

필드

- `PrimaryKeyColumnName` – Single-line string pattern과(와) 일치하는 1~1,024바이트 길이의 UTF-8 문자열입니다.

소스 테이블에서 행을 고유하게 식별하는 열의 이름입니다. 일치하는 레코드를 식별하기 위해 사용됩니다.

- `PrecisionRecallTradeoff` – 1.0 이하의 숫자(실수)입니다.

  정밀도와 재현율 간의 균형을 위해 변환을 튜닝할 때 선택하는 값입니다. 값 0.5는 기본 설정 없음, 값 1.0은 순전히 정밀도에 대한 바이어스, 값 0.0은 재현율에 대한 바이어스를 의미합니다. 이 값은 트레이드오프이기 때문에 1.0에 가까운 값을 선택하면 매우 낮은 재현율을 의미하고 0.0에 가까운 값을 선택하면 매우 낮은 정밀도를 나타냅니다.

  정밀도 지표는 모델이 일치를 정확하게 예측하는 빈도를 나타냅니다.

  재현율 지표는 실제 일치에 대해 모델이 일치를 예측하는 빈도를 나타냅니다.

- `AccuracyCostTradeoff` – 1.0 이하의 숫자(실수)입니다.

  정확도와 비용 간의 균형을 위해 변환을 튜닝할 때 선택하는 값입니다. 값 0.5는 시스템이 정확도 및 비용 문제의 균형을 유지하고 있음을 의미합니다. 값 1.0은 순전히 정확도에 대한 바이어스를 의미하며 일반적으로 더 높은 비용, 때로는 상당히 더 높은 비용을 나타냅니다. 값 0.0은 순전히 비용에 대한 바이어스를 의미하며 비교적 정확하지 않은 `FindMatches` 변환, 때로는 용인할 수 없는 수준의 정확도를 나타냅니다.

  정확도는 변환이 참 긍정과 참 부정을 얼마나 잘 찾는지 측정합니다. 정확도를 증가시키려면 더 많은 기계 리소스와 비용이 필요합니다. 하지만 이렇게 하면 재현율도 증가합니다.

  비용은 변환을 실행하는 데 얼마나 많은 컴퓨팅 리소스(따라서 비용)가 사용되는지를 측정합니다.

- `EnforceProvidedLabels` – 부울입니다.

  사용자가 제공한 레이블과 일치하는 강제 출력을 켜거나 끄는 값입니다. 값이 `True`이면 `find matches` 변환은 제공된 레이블과 일치하도록 출력을 강제합니다. 결과는 일반 융합 결과를 재정의합니다. 값이 `False`이면 `find matches` 변환은 제공된 모든 레이블이 존중될 것을 보장하지 않으며 결과는 교육된 모델에 따라 다릅니다.

  이 값을 true로 설정하면 융합 실행 시간이 증가합니다.

FindMatchesMetrics 구조

일치 항목 찾기 알고리즘에 대한 평가 지표입니다. 기계 학습 변환의 품질은 변환을 가져와서 몇 가지 일치를 예측하고 동일한 데이터세트의 알려진 일치 항목과 결과를 비교하여 측정됩니다. 품질 지표는 데이터의 하위 세트를 기반으로 하므로 정밀하지 않습니다.

필드

- `AreaUnderPRCurve` – 1.0 이하의 숫자(실수)입니다.

  정밀도/재현율 곡선(AUPRC) 아래 면적은 변환의 전체 품질을 측정하는 단일 숫자이며, 이 숫자는 재현율 대비 정밀도에 대해 수행한 선택과 무관합니다. 값이 높을수록 더 매력적인 재현율 대비 정밀도 트레이드오프가 있음을 나타냅니다.

  자세한 내용은 Wikipedia의 정밀도 및 재현율을 참조하십시오.

- `Precision` – 1.0 이하의 숫자(실수)입니다.

  정밀도 지표는 변환이 일치를 정확하게 예측하는 빈도를 나타냅니다. 특히 정밀도는 변환이 총 참 긍정에서 참 긍정을 얼마나 잘 찾는지를 측정합니다.

  자세한 내용은 Wikipedia의 정밀도 및 재현율을 참조하십시오.

- `Recall` – 1.0 이하의 숫자(실수)입니다.

  재현율 지표는 실제 일치에 대해 변환이 일치를 예측하는 빈도를 나타냅니다. 특히 재현율은 변환이 소스 데이터의 총 레코드에서 참 긍정을 얼마나 잘 찾는지를 측정합니다.

  자세한 내용은 Wikipedia의 정밀도 및 재현율을 참조하십시오.

- `F1` – 1.0 이하의 숫자(실수)입니다.

  최대 F1 지표는 0~1 범위에서 변환의 정확도를 나타냅니다. 여기서 1은 최상의 정확도입니다.

  자세한 내용은 Wikipedia의 F1 점수를 참조하십시오.

- `ConfusionMatrix` – ConfusionMatrix 객체입니다.

  혼동 행렬은 변환이 무엇을 정확하게 예측하고 있으며 어떤 유형의 오류가 발생하고 있는지를 보여줍니다.

  자세한 내용은 Wikipedia의 혼동 행렬을 참조하십시오.

- `ColumnImportances` – ColumnImportance 객체의 배열이며 구조는 100개 이하입니다.

  중요도 내림차순으로 정렬된 열 중요도 지표를 포함하는 `ColumnImportance` 구조 목록입니다.

ConfusionMatrix 구조

혼동 행렬은 변환이 무엇을 정확하게 예측하고 있으며 어떤 유형의 오류가 발생하고 있는지를 보여 줍니다.

자세한 내용은 Wikipedia의 혼동 행렬을 참조하십시오.

필드

- `NumTruePositives` - 숫자(정수)입니다.

  변환에 대한 혼동 행렬에서 변환이 올바르게 찾은 데이터의 일치 항목 수입니다.
- `NumFalsePositives` - 숫자(정수)입니다.

  변환에 대한 혼동 행렬에서 변환이 일치 항목으로 잘못 분류한 데이터의 불일치 항목 수입니다.
- `NumTrueNegatives` - 숫자(정수)입니다.

  변환에 대한 혼동 행렬에서 변환이 올바르게 거부한 데이터의 일치 항목 수입니다.
- `NumFalseNegatives` - 숫자(정수)입니다.

  변환에 대한 혼동 행렬에서 변환이 찾지 못한 데이터의 일치 항목 수입니다.

GlueTable 구조

입력 또는 출력 데이터에 사용되는 AWS Glue Data Catalog의 데이터베이스와 테이블입니다.

필드

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue Data Catalog의 데이터베이스 이름입니다.
- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue Data Catalog의 테이블 이름입니다.
- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue Data Catalog의 고유 식별자입니다.

- `ConnectionName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue Data Catalog에 대한 연결 이름입니다.
- `AdditionalOptions` – 1~10개 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 URI address multi-line string pattern와 일치하는 설명 문자열(2,048바이트 이하)입니다.

  테이블에 대한 추가 옵션입니다. 현재 지원되는 키는 두 가지입니다.
  - `pushDownPredicate`: 데이터 세트 내 모든 파일을 나열하거나 읽지 않아도 파티션에 필터링합니다.
  - `catalogPartitionPredicate`: AWS Glue Data Catalog에서 파티션 인덱스를 사용하여 서버 측 파티션 프루닝을 사용합니다.

TaskRun 구조

기계 학습 변환과 연결된 샘플링 파라미터입니다.

필드

- `TransformId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  변환의 고유 식별자입니다.
- `TaskRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 실행의 고유 식별자입니다.
- `Status` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `FINISHED` | `FAILED` | `PENDING_EXECUTION` | `TIMED_OUT` | `CANCELING` | `CANCELED` | `RECEIVED_BY_TASKRUNNER`).

  요청된 작업 실행의 현재 상태입니다.
- `LogGroupName` – UTF-8 문자열입니다.

  이 작업 실행과 연결된 보안 로깅을 위한 로그 그룹의 이름입니다.
- `Properties` – TaskRunProperties 객체입니다.

  이 작업 실행과 연결된 구성 속성을 지정합니다.

- `ErrorString` – UTF-8 문자열입니다.

  이 작업 실행과 연결된 오류 문자열의 목록입니다.

- `StartedOn` – 타임스탬프입니다.

  이 작업 실행이 시작된 날짜와 시간입니다.

- `LastModifiedOn` – 타임스탬프입니다.

  요청된 작업 실행이 업데이트된 마지막 시점입니다.

- `CompletedOn` – 타임스탬프입니다.

  요청된 작업 실행이 완료된 마지막 시점입니다.

- `ExecutionTime` - 숫자(정수)입니다.

  이 작업 실행이 리소스를 사용한 시간(초).입니다.

TransformFilterCriteria 구조

기계 학습 변환을 필터링하는 데 사용되는 기준입니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환을 필터링하는 데 사용되는 고유의 변환 이름입니다.

- `TransformType` – UTF-8 문자열입니다(유효한 값: `FIND_MATCHES`).

  기계 학습 변환을 필터링하는 데 사용되는 기계 학습 변환의 유형입니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `NOT_READY` | `READY` | `DELETING`).

  마지막으로 알려진 변환 상태를 기준으로 기계 학습 변환 목록을 필터링합니다(변환을 사용할 수 있는지 여부를 나타냄). "NOT_READY", "READY" 또는 "DELETING" 중 하나입니다.

- `GlueVersion` – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 값은 이러한 기계 학습 변환과 호환되는 AWS Glue의 버전을 결정합니다. Glue 1.0은 대부분의 고객에게 권장됩니다. 값이 설정되지 않은 경우 Glue 호환성은 Glue 0.9로 기본 설정됩니다. 자세한 내용은 개발자 안내서의 AWS Glue 버전을 참조하세요.

- `CreatedBefore` – 타임스탬프입니다.

  그 이전에 변환이 생성된 시간과 날짜입니다.

- `CreatedAfter` – 타임스탬프입니다.

  그 이후에 변환이 생성된 시간과 날짜입니다.

- `LastModifiedBefore` – 타임스탬프입니다.

  이 날짜 이전에 마지막으로 수정된 변환에서 필터링합니다.

- `LastModifiedAfter` – 타임스탬프입니다.

  이 날짜 이후에 마지막으로 수정된 변환에서 필터링합니다.

- `Schema` – SchemaColumn 객체의 배열이며 구조는 100개 이하입니다.

  특정 스키마를 사용하여 데이터세트에서 필터링합니다. `Map<Column, Type>` 객체는 이 변환이 허용하는 스키마를 나타내는 키-값 페어의 배열입니다. 여기서 `Column`은 열의 이름이고 `Type`은 정수 또는 문자열과 같은 데이터 형식입니다. 100열의 상한이 있습니다.

TransformSortCriteria 구조

기계 학습 변환과 연결된 정렬 기준입니다.

필드

- `Column` – 필수: UTF-8 문자열입니다(유효한 값: `NAME` | `TRANSFORM_TYPE` | `STATUS` | `CREATED` | `LAST_MODIFIED`).

  기계 학습 변환과 연결된 정렬 기준에 사용할 열입니다.

- `SortDirection` – 필수: UTF-8 문자열입니다(유효한 값: `DESCENDING` | `ASCENDING`).

  기계 학습 변환과 연결된 정렬 기준에 사용할 정렬 방향입니다.

TaskRunFilterCriteria 구조

기계 학습 변환에 대한 작업 실행을 필터링하는 데 사용되는 기준입니다.

필드

- TaskRunType – UTF-8 문자열입니다(유효한 값: EVALUATION | LABELING_SET_GENERATION | IMPORT_LABELS | EXPORT_LABELS | FIND_MATCHES).

  작업 실행의 유형입니다.

- Status – UTF-8 문자열입니다(유효한 값: RUNNING | FINISHED | FAILED | PENDING_EXECUTION | TIMED_OUT | CANCELING | CANCELED | RECEIVED_BY_TASKRUNNER).

  작업 실행의 현재 상태입니다.

- StartedBefore – 타임스탬프입니다.

  이 날짜 이전에 시작된 작업 실행에서 필터링합니다.

- StartedAfter – 타임스탬프입니다.

  이 날짜 이후에 시작된 작업 실행에서 필터링합니다.

TaskRunSortCriteria 구조

기계 학습 변환에 대한 작업 실행 목록을 정렬하는 데 사용되는 정렬 기준입니다.

필드

- Column – 필수: UTF-8 문자열입니다(유효한 값: TASK_RUN_TYPE | STATUS | STARTED).

  기계 학습 변환에 대한 작업 실행 목록을 정렬하는 데 사용할 열입니다.

- SortDirection – 필수: UTF-8 문자열입니다(유효한 값: DESCENDING | ASCENDING).

  기계 학습 변환에 대한 작업 실행 목록을 정렬하는 데 사용할 정렬 방향입니다.

TaskRunProperties 구조

작업 실행에 대한 구성 속성입니다.

필드

- TaskType – UTF-8 문자열입니다(유효한 값: EVALUATION | LABELING_SET_GENERATION | IMPORT_LABELS | EXPORT_LABELS | FIND_MATCHES).

작업 실행의 유형입니다.

- `ImportLabelsTaskRunProperties` – ImportLabelsTaskRunProperties 객체입니다.

  가져오기 레이블 작업 실행에 대한 구성 속성입니다.
- `ExportLabelsTaskRunProperties` – ExportLabelsTaskRunProperties 객체입니다.

  내보내기 레이블 작업 실행에 대한 구성 속성입니다.
- `LabelingSetGenerationTaskRunProperties` – LabelingSetGenerationTaskRunProperties 객체입니다.

  레이블 지정 세트 생성 작업 실행에 대한 구성 속성입니다.
- `FindMatchesTaskRunProperties` – FindMatchesTaskRunProperties 객체입니다.

  일치 항목 찾기 작업 실행에 대한 구성 속성입니다.

FindMatchesTaskRunProperties 구조

일치 항목 찾기 작업 실행에 대한 구성 속성을 지정합니다.

필드

- `JobId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  일치 항목 찾기 작업 실행의 작업 ID입니다.
- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  일치 항목 찾기 작업 실행에 대한 작업에 할당된 이름입니다
- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  일치 항목 찾기 작업 실행의 작업 실행 ID입니다.

ImportLabelsTaskRunProperties 구조

가져오기 레이블 작업 실행에 대한 구성 속성을 지정합니다.

필드

- `InputS3Path` – UTF-8 문자열입니다.

레이블을 가져올 Amazon Simple Storage Service(Amazon S3) 경로입니다.

- `Replace` – 부울입니다.

  기존 레이블을 덮어쓸지 여부를 표시합니다.

ExportLabelsTaskRunProperties 구조

내보내기 레이블 작업 실행에 대한 구성 속성을 지정합니다.

필드

- `OutputS3Path` – UTF-8 문자열입니다.

  레이블을 내보낼 Amazon Simple Storage Service(Amazon S3) 경로입니다.

LabelingSetGenerationTaskRunProperties 구조

레이블 지정 세트 생성 작업 실행에 대한 구성 속성을 지정합니다.

필드

- `OutputS3Path` – UTF-8 문자열입니다.

  레이블 지정 세트를 생성할 Amazon Simple Storage Service(Amazon S3) 경로입니다.

SchemaColumn 구조

이 변환이 실행할 수 있는 열과 데이터 형식을 나타내는 키-값 페어입니다. `MLTransform`의 `Schema` 파라미터에는 이러한 구조가 최대 100개까지 포함될 수 있습니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~1,024바이트 길이의 UTF-8 문자열입니다.

  열의 이름입니다.

- `DataType` – Single-line string pattern과(와) 일치하는 131,072바이트 이하 길이의 UTF-8 문자열입니다.

  열에 있는 데이터의 형식입니다.

TransformEncryption 구조

사용자 데이터 액세스에 적용되는 변환의 유휴 시 암호화 설정입니다. 기계 학습 변환에서는 KMS를 사용하여 Amazon S3의 암호화된 사용자 데이터에 액세스할 수 있습니다.

또한 가져온 레이블 및 교육된 변환은 이제 고객이 제공한 KMS 키를 사용하여 암호화할 수 있습니다.

필드

- `MlUserDataEncryption` – MLUserDataEncryption 객체입니다.

  `MLUserDataEncryption` 객체는 암호화 모드 및 고객 제공 KMS 키 ID를 포함합니다.
- `TaskRunSecurityConfigurationName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  보안 구성의 이름입니다.

MLUserDataEncryption 구조

사용자 데이터 액세스에 적용되는 변환의 유휴 시 암호화 설정입니다.

필드

- `MlUserDataEncryptionMode` – 필수: UTF-8 문자열입니다(유효한 값: `DISABLED` | `SSE-KMS="SSEKMS"`).

  사용자 데이터에 적용되는 암호화 모드입니다. 유효한 값은 다음과 같습니다.
  - DISABLED: 암호화가 비활성화됨
  - SSEKMS: Amazon S3에 저장된 사용자 데이터에 대해 AWS Key Management Service(SSE-KMS) 와 서버 측 암호화를 사용합니다.
- `KmsKeyId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  고객이 제공한 KMS 키의 ID입니다.

ColumnImportance 구조

열에 대한 열 이름 및 열 중요도 점수를 포함하는 구조입니다.

열 중요도는 레코드에서 다른 열보다 더 중요한 열을 식별하여 열이 모델에 어떻게 기여하는지 이해하는 데 도움이 됩니다.

필드

- `ColumnName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  열의 이름입니다.
- `Importance` – 1.0 이하의 숫자(실수)입니다.

  열에 대한 열 중요도 점수(10진수)입니다.

운영

- CreateMLTransform 작업(Python: create_ml_transform)
- UpdateMLTransform 작업(Python: update_ml_transform)
- DeleteMLTransform 작업(Python: delete_ml_transform)
- GetMLTransform 작업(Python: get_ml_transform)
- GetMLTransforms 작업(Python: get_ml_transforms)
- ListMLTransforms 작업(Python: list_ml_transforms)
- StartMLEvaluationTaskRun 작업(Python: start_ml_evaluation_task_run)
- StartMLLabelingSetGenerationTaskRun 작업(Python: start_ml_labeling_set_generation_task_run)
- GetMLTaskRun 작업(Python: get_ml_task_run)
- GetMLTaskRuns 작업(Python: get_ml_task_runs)
- CancelMLTaskRun 작업(Python: cancel_ml_task_run)
- StartExportLabelsTaskRun 작업(Python: start_export_labels_task_run)
- StartImportLabelsTaskRun 작업(Python: start_import_labels_task_run)

CreateMLTransform 작업(Python: create_ml_transform)

AWS Glue 기계 학습 변환을 생성합니다. 이 작업은 변환과 변환을 교육하는 데 필요한 모든 파라미터를 생성합니다.

데이터 중복 제거에 기계 학습 변환(예: FindMatches 변환)을 사용하는 프로세스의 첫 번째 단계로 이 작업을 호출합니다. 알고리즘에 사용할 파라미터 외에도 선택 사항인 Description을 제공할 수 있습니다.

데이터에서 학습하고 고품질 기계 학습 변환을 생성하는 과정의 일부로 AWS Glue가 자동으로 실행하는 태스크에 대한 특정 파라미터도 지정해야 합니다. 이러한 파라미터에는 Role과 선택 사항으로 AllocatedCapacity, Timeout 및 MaxRetries가 포함됩니다. 자세한 내용은 작업을 참조하십시오.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  변환을 생성할 때 변환에 부여한 고유 이름입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  정의하고 있는 기계 학습 변환에 대한 설명입니다. 기본값은 빈 문자열입니다.

- InputRecordTables – 필수(Required): GlueTable 객체의 배열이며 구조는 10개 이하입니다.

  변환에 사용된 AWS Glue 테이블 정의의 목록입니다.

- Parameters – 필수(Required): TransformParameters 객체입니다.

  사용된 변환 유형에 특정한 알고리즘 파라미터입니다. 조건부로 변환 유형에 따라 다릅니다.

- Role – 필수(Required): UTF-8 문자열입니다.

  필수 권한이 있는 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다. 필요한 권한에는 AWS Glue 리소스에 대한 AWS Glue 서비스 역할 권한과 변환에서 요구하는 Amazon S3 권한 모두가 포함됩니다.

  - 이 역할에는 AWS Glue의 리소스에 대한 액세스 허용을 위한 AWS Glue 서비스 역할 권한이 필요합니다. AWS Glue에 액세스하는 IAM 사용자에게 정책 연결을 참조하세요.
  - 이 역할에는 작업 실행에서 이 변환에 사용되는 Amazon Simple Storage Service(Amazon S3) 소스, 대상, 임시 디렉터리, 스크립트 및 모든 라이브러리에 대한 권한이 필요합니다.

- GlueVersion – Custom string pattern #47과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

이 값은 이러한 기계 학습 변환과 호환되는 AWS Glue의 버전을 결정합니다. Glue 1.0은 대부분의 고객에게 권장됩니다. 값이 설정되지 않은 경우 Glue 호환성은 Glue 0.9로 기본 설정됩니다. 자세한 내용은 개발자 안내서의 AWS Glue 버전을 참조하세요.

- `MaxCapacity` - 숫자(double)입니다.

  이 변환의 태스크 실행에 할당된 AWS Glue 데이터 처리 장치(DPU)의 수입니다. 2~100DPU를 할당할 수 있으며, 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  `MaxCapacity`는 `NumberOfWorkers` 및 `WorkerType`과 함께 사용할 수 없는 옵션입니다.

  - `NumberOfWorkers`나 `WorkerType` 중 하나가 설정되면 `MaxCapacity`를 설정할 수 없습니다.
  - `MaxCapacity`가 설정되면 `NumberOfWorkers`와 `WorkerType` 모두 설정할 수 없습니다.
  - `WorkerType`이 설정되면 `NumberOfWorkers`가 필요합니다(반대의 경우도 마찬가지).
  - `MaxCapacity`와 `NumberOfWorkers`는 1 이상이어야 합니다.

  `WorkerType` 필드를 `Standard` 이외의 다른 값으로 설정하면 `MaxCapacity` 필드가 자동으로 설정되고 읽기 전용이 됩니다.

  `WorkerType` 필드를 `Standard` 이외의 다른 값으로 설정하면 `MaxCapacity` 필드가 자동으로 설정되고 읽기 전용이 됩니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  이 작업이 실행될 때 할당되는 미리 정의된 작업자의 유형입니다. Standard, G.1X 또는 G.2X 값을 허용합니다.

  - `Standard` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 50GB 디스크와, 작업자당 실행기 2개를 제공합니다.
  - `G.1X` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 64GB 디스크와, 작업자당 실행기 1개를 제공합니다.
  - `G.2X` 작업자 유형의 경우, 각 작업자가 8vCPU, 32GB 메모리 및 128GB 디스크와, 작업자당 실행기 1개를 제공합니다.

  `MaxCapacity`는 `NumberOfWorkers` 및 `WorkerType`과 함께 사용할 수 없는 옵션입니다.

  - `NumberOfWorkers`나 `WorkerType` 중 하나가 설정되면 `MaxCapacity`를 설정할 수 없습니다.
  - `MaxCapacity`가 설정되면 `NumberOfWorkers`와 `WorkerType` 모두 설정할 수 없습니다.

  - `WorkerType`이 설정되면 `NumberOfWorkers`가 필요합니다(반대의 경우도 마찬가지).
  - `MaxCapacity`와 `NumberOfWorkers`는 1 이상이어야 합니다.
- `NumberOfWorkers` - 숫자(정수)입니다.

  이 작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.

  `WorkerType`이 설정되면 `NumberOfWorkers`가 필요합니다(반대의 경우도 마찬가지).
- `Timeout` – 1 이상의 숫자(정수)입니다.

  이 변환에 대한 작업 실행의 제한 시간(분)입니다. 이 값은 이 변환에 대한 작업 실행이 종료되고 `TIMEOUT` 상태로 전환되기 전에 리소스를 사용할 수 있는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.
- `MaxRetries` - 숫자(정수)입니다.

  작업 실행이 실패한 후 이 변환에 대한 작업을 재시도할 최대 횟수입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이러한 기계 학습 변환에서 사용할 태그입니다. 태그를 사용하여 기계 학습 변환에 대한 액세스를 제한할 수 있습니다. AWS Glue의 태그에 대한 자세한 내용은 개발자 안내서의 AWS Glue의 AWS 태그를 참조하세요.
- `TransformEncryption` – TransformEncryption 객체입니다.

  사용자 데이터 액세스에 적용되는 변환의 유휴 시 암호화 설정입니다. 기계 학습 변환에서는 KMS를 사용하여 Amazon S3의 암호화된 사용자 데이터에 액세스할 수 있습니다.

응답

- `TransformId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  변환에 대해 생성되는 고유 식별자입니다.

오류

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`
- `ResourceNumberLimitExceededException`
- `IdempotentParameterMismatchException`

UpdateMLTransform 작업(Python: update_ml_transform)

기존 기계 학습 변환을 업데이트합니다. 더 좋은 결과를 달성하도록 알고리즘 파라미터를 튜닝하려면 이 작업을 호출합니다.

이 작업을 호출한 후, `StartMLEvaluationTaskRun` 작업을 호출하여 새로운 파라미터가 목표를 얼마나 잘 달성했는지에 액세스할 수 있습니다(예: 기계 학습 변환의 품질 개선 또는 비용 효과 향상).

요청

- `TransformId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  변환을 생성할 때 생성된 고유 식별자입니다.
- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  생성할 때 변환에 부여한 고유 이름입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  변환에 대한 설명입니다. 기본값은 빈 문자열입니다.
- `Parameters` – TransformParameters 객체입니다.

  사용된 변환 유형(알고리즘)에 특정한 구성 파라미터입니다. 조건부로 변환 유형에 따라 다릅니다.
- `Role` – UTF-8 문자열입니다.

  필수 권한이 있는 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다.

- `GlueVersion` – Custom string pattern #47과 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 값은 이러한 기계 학습 변환과 호환되는 AWS Glue의 버전을 결정합니다. Glue 1.0은 대부분의 고객에게 권장됩니다. 값이 설정되지 않은 경우 Glue 호환성은 Glue 0.9로 기본 설정됩니다. 자세한 내용은 개발자 안내서의 AWS Glue 버전을 참조하세요.

- `MaxCapacity` - 숫자(double)입니다.

  이 변환의 태스크 실행에 할당된 AWS Glue 데이터 처리 장치(DPU)의 수입니다. 2~100DPU를 할당할 수 있으며, 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  `WorkerType` 필드를 `Standard` 이외의 다른 값으로 설정하면 `MaxCapacity` 필드가 자동으로 설정되고 읽기 전용이 됩니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  이 작업이 실행될 때 할당되는 미리 정의된 작업자의 유형입니다. Standard, G.1X 또는 G.2X 값을 허용합니다.

  - `Standard` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 50GB 디스크와, 작업자당 실행기 2개를 제공합니다.
  - `G.1X` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 64GB 디스크와, 작업자당 실행기 1개를 제공합니다.
  - `G.2X` 작업자 유형의 경우, 각 작업자가 8vCPU, 32GB 메모리 및 128GB 디스크와, 작업자당 실행기 1개를 제공합니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  이 작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  이 변환에 대한 작업 실행의 제한 시간(분)입니다. 이 값은 이 변환에 대한 작업 실행이 종료되고 `TIMEOUT` 상태로 전환되기 전에 리소스를 사용할 수 있는 최대 시간입니다. 기본값은 2,880 분(48시간)입니다.

- `MaxRetries` - 숫자(정수)입니다.

  작업 실행이 실패한 후 이 변환에 대한 작업을 재시도할 최대 횟수입니다.

응답

- `TransformId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  업데이트된 변환의 고유 식별자입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`

DeleteMLTransform 작업(Python: delete_ml_transform)

AWS Glue 기계 학습 변환을 삭제합니다. 기계 학습 변환은 기계 학습을 통해 사람이 제공한 사례에서 학습하여 수행할 변환에 대한 세부 정보를 학습하는 특수한 유형의 변환입니다. 그런 다음 이러한 변환은 AWS Glue에 의해 저장됩니다. 변환이 더 이상 필요하지 않으면 `DeleteMLTransforms`를 호출하여 변환을 삭제할 수 있습니다. 하지만 삭제된 변환을 여전히 참조하는 AWS Glue 작업은 더 이상 성공하지 못합니다.

요청

- `TransformId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 변환의 고유 식별자입니다.

응답

- `TransformId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제된 변환의 고유 식별자입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetMLTransform 작업(Python: get_ml_transform)

AWS Glue 기계 학습 변환 아티팩트와 모든 해당 메타데이터를 가져옵니다. 기계 학습 변환은 기계 학습을 통해 사람이 제공한 사례에서 학습하여 수행할 변환에 대한 세부 정보를 학습하는 특수한 유형의 변환입니다. 그런 다음 이러한 변환은 AWS Glue에 의해 저장됩니다. `GetMLTransform`을 호출하여 해당 메타데이터를 검색할 수 있습니다.

요청

- `TransformId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  변환을 생성할 때 생성된 변환의 고유 식별자입니다.

응답

- `TransformId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  변환을 생성할 때 생성된 변환의 고유 식별자입니다.
- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  변환을 생성할 때 변환에 지정된 고유 이름입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  변환에 대한 설명입니다.
- `Status` – UTF-8 문자열입니다(유효한 값: `NOT_READY` | `READY` | `DELETING`).

  변환의 마지막 알려진 상태입니다(변환을 사용할 수 있는지 여부를 나타냄). "NOT_READY", "READY" 또는 "DELETING" 중 하나입니다.

- `CreatedOn` – 타임스탬프입니다.

  변환이 생성된 날짜와 시간입니다.

- `LastModifiedOn` – 타임스탬프입니다.

  변환이 수정된 날짜와 시간입니다.

- `InputRecordTables` – GlueTable 객체의 배열이며 구조는 10개 이하입니다.

  변환에 사용된 AWS Glue 테이블 정의의 목록입니다.

- `Parameters` – TransformParameters 객체입니다.

  사용된 알고리즘에 특정한 구성 파라미터입니다.

- `EvaluationMetrics` – EvaluationMetrics 객체입니다.

  최신 평가 지표입니다.

- `LabelCount` - 숫자(정수)입니다.

  이 변환에 사용 가능한 레이블 수입니다.

- `Schema` – SchemaColumn 객체의 배열이며 구조는 100개 이하입니다.

  이 변환이 허용하는 스키마를 나타내는 `Map<Column, Type>` 객체입니다. 100열의 상한이 있습니다.

- `Role` – UTF-8 문자열입니다.

  필수 권한이 있는 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다.

- `GlueVersion` – Custom string pattern #47과 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 값은 이러한 기계 학습 변환과 호환되는 AWS Glue의 버전을 결정합니다. Glue 1.0은 대부분의 고객에게 권장됩니다. 값이 설정되지 않은 경우 Glue 호환성은 Glue 0.9로 기본 설정됩니다. 자세한 내용은 개발자 안내서의 AWS Glue 버전을 참조하세요.

- `MaxCapacity` - 숫자(double)입니다.

  이 변환의 태스크 실행에 할당된 AWS Glue 데이터 처리 장치(DPU)의 수입니다. 2~100DPU를 할당할 수 있으며, 기본값은 10입니다. DPU는 4 vCPU의 컴퓨팅 파워와 16GB 메모리로 구성된 프로세싱 파워의 상대적 측정값입니다. 자세한 내용은 AWS Glue 요금 페이지를 참조하세요.

  `WorkerType` 필드를 `Standard` 이외의 다른 값으로 설정하면 `MaxCapacity` 필드가 자동으로 설정되고 읽기 전용이 됩니다.

- `WorkerType` – UTF-8 문자열입니다(유효한 값: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  이 작업이 실행될 때 할당되는 미리 정의된 작업자의 유형입니다. Standard, G.1X 또는 G.2X 값을 허용합니다.

  - `Standard` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 50GB 디스크와, 작업자당 실행기 2개를 제공합니다.
  - `G.1X` 작업자 유형의 경우, 각 작업자가 4vCPU, 16GB 메모리 및 64GB 디스크와, 작업자당 실행기 1개를 제공합니다.
  - `G.2X` 작업자 유형의 경우, 각 작업자가 8vCPU, 32GB 메모리 및 128GB 디스크와, 작업자당 실행기 1개를 제공합니다.
- `NumberOfWorkers` - 숫자(정수)입니다.

  이 작업이 실행될 때 할당되는 정의된 `workerType`의 작업자 수입니다.
- `Timeout` – 1 이상의 숫자(정수)입니다.

  이 변환에 대한 작업 실행의 제한 시간(분)입니다. 이 값은 이 변환에 대한 작업 실행이 종료되고 `TIMEOUT` 상태로 전환되기 전에 리소스를 사용할 수 있는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.
- `MaxRetries` - 숫자(정수)입니다.

  작업 실행이 실패한 후 이 변환에 대한 작업을 재시도할 최대 횟수입니다.
- `TransformEncryption` – TransformEncryption 객체입니다.

  사용자 데이터 액세스에 적용되는 변환의 유휴 시 암호화 설정입니다. 기계 학습 변환에서는 KMS를 사용하여 Amazon S3의 암호화된 사용자 데이터에 액세스할 수 있습니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetMLTransforms 작업(Python: get_ml_transforms)

기존 AWS Glue 기계 학습 변환의 정렬 가능하고 필터링 가능한 목록을 가져옵니다. 기계 학습 변환은 기계 학습을 통해 사람이 제공한 사례에서 학습하여 수행할 변환에 대한 세부 정보를 학습하는 특수한 유형의 변환입니다. 그런 다음 이러한 변환은 AWS Glue에 의해 저장되며, `GetMLTransforms`를 호출하여 해당 메타데이터를 검색할 수 있습니다.

요청

- `NextToken` – UTF-8 문자열입니다.

  결과를 오프셋하기 위한 페이지 매김 토큰입니다.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.

- `Filter` – TransformFilterCriteria 객체입니다.

  변환 필터링 기준입니다.

- `Sort` – TransformSortCriteria 객체입니다.

  정렬 기준입니다.

응답

- `Transforms` – 필수(Required): MLTransform 객체의 배열입니다.

  기계 학습 변환의 목록입니다.

- `NextToken` – UTF-8 문자열입니다.

  추가 결과를 사용할 수 있는 경우 페이지 매김 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListMLTransforms 작업(Python: list_ml_transforms)

이 AWS 계정에 있는 기존의 AWS Glue 기계 학습 변환이나 지정된 태그가 있는 리소스에 대한 정렬 및 필터링 가능한 목록을 검색합니다. 이 작업을 수행하면 응답의 필터로 사용할 수 있는 Tags 필드 옵션이 검색되기 때문에 태그가 지정된 리소스를 하나의 그룹으로 검색할 수 있습니다. 태그 필터링을 사용하기로 선택하면 태그가 포함된 리소스만 검색됩니다.

요청

- NextToken – UTF-8 문자열입니다.

  이것이 지속적인 요청이라면 지속적인 토큰입니다.

- MaxResults – 1~1,000의 숫자(정수)입니다.

  반환할 목록의 최대 크기.

- Filter – TransformFilterCriteria 객체입니다.

  기계 학습 변환을 필터링하는 데 사용되는 TransformFilterCriteria입니다.

- Sort – TransformSortCriteria 객체입니다.

  기계 학습 변환을 정렬하는 데 사용되는 TransformSortCriteria입니다.

- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  이렇게 태그가 지정된 리소스만 반환하도록 지정합니다.

응답

- TransformIds – 필수(Required): UTF-8 문자열의 배열입니다.

  계정의 모든 기계 학습 변환이나 태그가 지정된 기계 학습의 식별자입니다.

- NextToken – UTF-8 문자열입니다.

  반환된 목록이 사용가능한 마지막 지표를 포함하지 경우의 연속 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

StartMLEvaluationTaskRun 작업(Python: start_ml_evaluation_task_run)

변환 품질을 예상하기 위한 작업을 시작합니다.

레이블 집합을 truth의 사례로 제공하면 AWS Glue 기계 학습은 이러한 사례 중 일부를 사용하여 해당 사례에서 학습합니다. 나머지 레이블은 품질을 예상하기 위한 테스트로 사용됩니다.

실행의 고유 식별자를 반환합니다. `GetMLTaskRun`을 호출하여 `EvaluationTaskRun` 통계에 대한 자세한 정보를 가져올 수 있습니다.

요청

- `TransformId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.

응답

- `TaskRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유 식별자입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConcurrentRunsExceededException`

- MLTransformNotReadyException

StartMLLabelingSetGenerationTaskRun 작업(Python: start_ml_labeling_set_generation_task_run)

레이블 세트를 생성하고 레이블을 지정하여 변환의 품질을 개선하기 위해 기계 학습 변환에 대한 활성 학습 워크플로우를 시작합니다.

StartMLLabelingSetGenerationTaskRun이 완료되면 AWS Glue는 "레이블 지정 집합" 또는 사람이 답변할 질문 집합을 생성합니다.

FindMatches 변환의 경우 이러한 질문은 "일치하는 레코드로 완전히 구성된 그룹으로 이러한 행을 함께 그룹화하는 올바른 방법은 무엇입니까?"와 같은 형식입니다.

레이블 지정 프로세스가 완료된 후에는 StartImportLabelsTaskRun을 호출하여 레이블을 업로드할 수 있습니다. StartImportLabelsTaskRun이 완료된 후에는 기계 학습 변환의 모든 향후 실행이 새롭고 개선된 레이블을 사용하며 더 높은 품질의 변환을 수행합니다.

참고: 생성된 레이블 지정 세트를 OutputS3Path에 쓰는 데 사용되는 역할은 CreateMLTransform API에 지정된 기계 학습 변환과 연결된 역할입니다.

요청

- TransformId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.
- OutputS3Path – 필수(Required): UTF-8 문자열입니다.

  레이블 지정 세트를 생성하는 Amazon Simple Storage Service(Amazon S3) 경로입니다.

응답

- TaskRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 작업 실행과 연결된 고유의 실행 식별자입니다.

오류

- EntityNotFoundException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException
- ConcurrentRunsExceededException

GetMLTaskRun 작업(Python: get_ml_task_run)

기계 학습 변환의 특정 작업 실행에 대한 세부 사항을 가져옵니다. 기계 학습 태스크 실행은 다양한 기계 학습 워크플로의 일부로 AWS Glue가 자동으로 실행하는 비동기 태스크입니다. TaskRunID 및 해당 상위 변환의 TransformID와 함께 GetMLTaskRun을 호출하여 모든 작업 실행의 통계를 확인할 수 있습니다.

요청

- TransformId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.
- TaskRunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행의 고유 식별자입니다.

응답

- TransformId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행의 고유 식별자입니다.
- TaskRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.
- Status – UTF-8 문자열입니다(유효한 값: RUNNING | FINISHED | FAILED | PENDING_EXECUTION | TIMED_OUT | CANCELING | CANCELED | RECEIVED_BY_TASKRUNNER).

이 작업 실행의 상태입니다.

- LogGroupName – UTF-8 문자열입니다.

  작업 실행과 연결된 로그 그룹의 이름입니다.

- Properties – TaskRunProperties 객체입니다.

  작업 실행과 연결된 속성의 목록입니다.

- ErrorString – UTF-8 문자열입니다.

  작업 실행과 연결된 오류 문자열입니다.

- StartedOn – 타임스탬프입니다.

  이 작업 실행이 시작된 날짜와 시간입니다.

- LastModifiedOn – 타임스탬프입니다.

  이 작업 실행이 마지막으로 수정된 날짜와 시간입니다.

- CompletedOn – 타임스탬프입니다.

  이 작업 실행이 완료된 날짜와 시간입니다.

- ExecutionTime - 숫자(정수)입니다.

  이 작업 실행이 리소스를 사용한 시간(초).입니다.

오류

- EntityNotFoundException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException

GetMLTaskRuns 작업(Python: get_ml_task_runs)

기계 학습 변환에 대한 실행 목록을 가져옵니다. 기계 학습 태스크 실행은 다양한 기계 학습 워크플로의 일부로 AWS Glue가 자동으로 실행하는 비동기 태스크입니다. 이 단원에 기록된 해당 상위 변환의 TransformID 및 기타 선택적 파라미터와 함께 GetMLTaskRuns를 호출하여 정렬 가능하고 필터링 가능한 기계 학습 작업 실행 목록을 가져올 수 있습니다.

이 작업은 기록 실행 목록을 반환하며 페이지를 매겨야 합니다.

요청

- `TransformId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.

- `NextToken` – UTF-8 문자열입니다.

  결과의 페이지 매김에 대한 토큰입니다. 기본값은 비어 있음입니다.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.

- `Filter` – TaskRunFilterCriteria 객체입니다.

  작업 실행에 대한 `TaskRunFilterCriteria` 구조의 필터 기준입니다.

- `Sort` – TaskRunSortCriteria 객체입니다.

  작업 실행에 대한 `TaskRunSortCriteria` 구조의 정렬 기준입니다.

응답

- `TaskRuns` – TaskRun 객체의 배열입니다.

  변환과 연결된 작업 실행의 목록입니다.

- `NextToken` – UTF-8 문자열입니다.

  추가 결과를 사용할 수 있는 경우 페이지 매김 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

CancelMLTaskRun 작업(Python: cancel_ml_task_run)

작업 실행을 취소(중지)합니다. 기계 학습 태스크 실행은 다양한 기계 학습 워크플로의 일부로 AWS Glue가 자동으로 실행하는 비동기 태스크입니다. 작업 실행의 상위 변환의 `TransformID` 및 작업 실행의 `TaskRunId`와 함께 `CancelMLTaskRun`을 호출하여 언제든지 기계 학습 작업 실행을 취소할 수 있습니다.

요청

- `TransformId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.
- `TaskRunId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행의 고유 식별자입니다.

응답

- `TransformId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.
- `TaskRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행의 고유 식별자입니다.
- `Status` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `FINISHED` | `FAILED` | `PENDING_EXECUTION` | `TIMED_OUT` | `CANCELING` | `CANCELED` | `RECEIVED_BY_TASKRUNNER`).

  이 실행의 상태입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

StartExportLabelsTaskRun 작업(Python: start_export_labels_task_run)

특정 변환에 대해 레이블 지정된 모든 데이터를 내보내기 위한 비동기 작업을 시작합니다. 이 작업은 일반적인 활성 학습 워크플로우의 일부가 아닌 유일한 레이블 관련 API 호출입니다. 이전에 truth로 제출한 레이블을 제거하거나 변경하려는 경우와 같이 모든 기존 레이블을 동시에 작업하려는 경우에 일반적으로 `StartExportLabelsTaskRun`을 사용합니다. 이 API 작업은 레이블을 내보내려고 하는 `TransformId` 및 레이블을 내보낼 Amazon Simple Storage Service(Amazon S3) 경로를 허용합니다. 이 작업은 `TaskRunId`를 반환합니다. `GetMLTaskRun` API를 호출하여 작업 실행의 상태를 확인할 수 있습니다.

요청

- `TransformId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.

- `OutputS3Path` – 필수(Required): UTF-8 문자열입니다.

  레이블을 내보내는 Amazon S3 경로입니다.

응답

- `TaskRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행의 고유 식별자입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

StartImportLabelsTaskRun 작업(Python: start_import_labels_task_run)

기계 학습 변환을 교육하고 품질을 개선하는 데 사용할 추가 레이블(truth의 사례)을 제공할 수 있습니다. 일반적으로 이 API 작업은 StartMLLabelingSetGenerationTaskRun 호출로 시작하고 결국 기계 학습 변환의 품질을 개선하는 활성 학습 워크플로우의 일부로 사용됩니다.

StartMLLabelingSetGenerationTaskRun이 완료되면 AWS Glue 기계 학습은 사람이 답변할 일련의 질문을 생성합니다. (기계 학습 워크플로우에서는 이러한 질문에 답변하는 것을 흔히 '레이블 지정'이라고 합니다). FindMatches 변환의 경우 이러한 질문은 "일치하는 레코드로 완전히 구성된 그룹으로 이러한 행을 함께 그룹화하는 올바른 방법은 무엇입니까?"와 같은 형식입니다. 레이블 지정 프로세스가 완료된 후 사용자는 StartImportLabelsTaskRun을 호출하여 답변/레이블을 업로드할 수 있습니다. StartImportLabelsTaskRun이 완료된 후에는 기계 학습 변환의 모든 향후 실행이 새롭고 개선된 레이블을 사용하며 더 높은 품질의 변환을 수행합니다.

기본적으로 Replace를 true로 설정하지 않는 한 StartMLLabelingSetGenerationTaskRun은 업로드하는 모든 레이블에서 지속적으로 학습하고 업로드하는 모든 레이블을 결합합니다. Replace를 true로 설정하면 StartImportLabelsTaskRun은 이전에 업로드한 모든 레이블을 삭제하고 잊어버리며 업로드하는 정확한 세트에서만 학습합니다. 레이블 바꾸기는 잘못된 레이블을 이전에 업로드한 것을 인식하고 해당 레이블이 변환 품질에 부정적인 영향을 미치고 있다고 확신하는 경우에 유용할 수 있습니다.

GetMLTaskRun 작업을 호출하여 작업 실행의 상태를 확인할 수 있습니다.

요청

- TransformId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  기계 학습 변환의 고유 식별자입니다.
- InputS3Path – 필수(Required): UTF-8 문자열입니다.

  레이블을 가져올 Amazon Simple Storage Service(Amazon S3) 경로입니다.
- ReplaceAllLabels – 부울입니다.

  기존 레이블을 덮어쓸지 여부를 표시합니다.

응답

- TaskRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

작업 실행의 고유 식별자입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`

데이터 품질 API

데이터 품질 API는 데이터 품질 데이터 유형에 대해 설명하며 데이터 품질 규칙 세트, 실행 및 평가를 생성, 삭제 또는 업데이트하기 위한 API를 포함합니다.

데이터 타입

- DataSource 구조
- DataQualityRulesetListDetails 구조
- DataQualityTargetTable 구조
- DataQualityRulesetEvaluationRunDescription 구조
- DataQualityRulesetEvaluationRunFilter 구조
- DataQualityEvaluationRunAdditionalRunOptions 구조
- DataQualityRuleRecommendationRunDescription 구조
- DataQualityRuleRecommendationRunFilter 구조
- DataQualityResult 구조
- DataQualityAnalyzerResult 구조
- DataQualityObservation 구조
- MetricBasedObservation 구조
- DataQualityMetricValues 구조
- DataQualityRuleResult 구조

- DataQualityResultDescription 구조
- DataQualityResultFilterCriteria 구조
- DataQualityRulesetFilterCriteria 구조
- DataQualityAggregatedMetrics 구조
- StatisticAnnotation 구조
- TimestampedInclusionAnnotation 구조
- AnnotationError 구조
- DatapointInclusionAnnotation 구조
- StatisticSummaryList 목록
- StatisticSummary 구조
- RunIdentifier 구조
- StatisticModelResult 구조

DataSource 구조

데이터 품질 결과를 얻으려는 데이터 소스(AWS Glue 테이블)입니다.

필드

- `GlueTable` – 필수: GlueTable 객체입니다.

  AWS Glue 테이블

DataQualityRulesetListDetails 구조

`GetDataQualityRuleset`에서 반환되는 데이터 품질 규칙 세트를 설명합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트의 이름입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  데이터 품질 규칙 세트에 대한 설명입니다.

- `CreatedOn` – 타임스탬프입니다.

  데이터 품질 규칙 세트가 생성된 날짜와 시간입니다.

- `LastModifiedOn` – 타임스탬프입니다.

  데이터 품질 규칙 세트가 마지막으로 수정된 날짜와 시간입니다.

- `TargetTable` – DataQualityTargetTable 객체입니다.

  AWS Glue 테이블을 나타내는 객체입니다.

- `RecommendationRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  권장 실행에서 규칙 세트가 생성되면 이 실행 ID가 생성되어 두 규칙을 서로 연결합니다.

- `RuleCount` - 숫자(정수)입니다.

  규칙 세트의 규칙 수입니다.

DataQualityTargetTable 구조

AWS Glue 테이블을 나타내는 객체입니다.

필드

- `TableName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 테이블의 이름.

- `DatabaseName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 테이블이 속한 데이터베이스의 이름입니다.

- `CatalogId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  AWS Glue 테이블이 있는 카탈로그 ID입니다.

DataQualityRulesetEvaluationRunDescription 구조

데이터 품질 규칙 세트 평가 실행 결과를 설명합니다.

필드

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.
- `Status` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `FINISHED` | `FAILED` | `PENDING_EXECUTION` | `TIMED_OUT` | `CANCELING` | `CANCELED` | `RECEIVED_BY_TASKRUNNER`).

  이 실행의 상태입니다.
- `StartedOn` – 타임스탬프입니다.

  실행이 시작된 날짜와 시간입니다.
- `DataSource` – DataSource 객체입니다.

  실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.

DataQualityRulesetEvaluationRunFilter 구조

필터 기준입니다.

필드

- `DataSource` – 필수: DataSource 객체입니다.

  실행과 연결된 데이터 소스(AWS Glue 테이블)를 기반으로 필터링합니다.
- `StartedBefore` – 타임스탬프입니다.

  이 시간 이전에 시작된 실행을 기준으로 결과를 필터링합니다.
- `StartedAfter` – 타임스탬프입니다.

  이 시간 이후에 시작된 실행을 기준으로 결과를 필터링합니다.

DataQualityEvaluationRunAdditionalRunOptions 구조

평가 실행에 대해 지정할 수 있는 추가 실행 옵션입니다.

필드

- `CloudWatchMetricsEnabled` – 부울입니다.

CloudWatch 지표를 활성화할지 여부입니다.

- `ResultsS3Prefix` – UTF-8 문자열입니다.

  결과를 저장할 Amazon S3의 접두사입니다.

- `CompositeRuleEvaluationMethod` – UTF-8 문자열입니다(유효한 값: `COLUMN` | `ROW`).

  규칙 세트의 복합 규칙 평가 방법을 ROW/COLUMN으로 설정합니다.

DataQualityRuleRecommendationRunDescription 구조

데이터 품질 규칙 권장 실행 결과를 설명합니다.

필드

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `FINISHED` | `FAILED` | `PENDING_EXECUTION` | `TIMED_OUT` | `CANCELING` | `CANCELED` | `RECEIVED_BY_TASKRUNNER`).

  이 실행의 상태입니다.

- `StartedOn` – 타임스탬프입니다.

  이 실행이 시작된 날짜와 시간입니다.

- `DataSource` – DataSource 객체입니다.

  권장 실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.

DataQualityRuleRecommendationRunFilter 구조

데이터 품질 권장 실행을 나열하기 위한 필터입니다.

필드

- `DataSource` – 필수: DataSource 객체입니다.

  지정된 데이터 원본(AWS Glue 테이블)을 기반으로 필터링합니다.

- `StartedBefore` – 타임스탬프입니다.

제공된 시간 이전에 시작된 결과에 대한 시간을 기준으로 필터링합니다.

- `StartedAfter` – 타임스탬프입니다.

  제공된 시간 이후에 시작된 결과에 대한 시간을 기준으로 필터링합니다.

DataQualityResult 구조

데이터 품질 결과를 설명합니다.

필드

- `ResultId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과의 고유한 결과 ID입니다.
- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과의 프로필 ID입니다.
- `Score` – 1.0 이하의 숫자(실수)입니다.

  집계된 데이터 품질 점수입니다. 총 규칙 수에 전달된 규칙의 비율을 나타냅니다.
- `DataSource` – DataSource 객체입니다.

  데이터 품질 결과와 연결된 테이블입니다(있는 경우).
- `RulesetName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 규칙 세트의 이름입니다.
- `EvaluationContext` – UTF-8 문자열입니다.

  AWS Glue Studio의 작업 컨텍스트에서 캔버스의 각 노드에는 일반적으로 일종의 이름이 할당되며 데이터 품질 노드에는 이름이 지정됩니다. 여러 노드의 경우 `evaluationContext`에서 노드를 구분할 수 있습니다.
- `StartedOn` – 타임스탬프입니다.

  이 데이터 품질 실행이 시작된 날짜와 시간입니다.
- `CompletedOn` – 타임스탬프입니다.

  이 데이터 품질 실행이 완료된 날짜와 시간입니다.

- JobName – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 작업 이름입니다(있는 경우).

- JobRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 작업 실행 ID입니다(있는 경우).

- RulesetEvaluationRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 데이터 품질 결과에 대한 규칙 세트 평가의 고유한 실행 ID입니다.

- RuleResults – DataQualityRuleResult 객체의 배열이며 구조는 2,000개 이하입니다.

  각 규칙의 결과를 나타내는 DataQualityRuleResult 객체 목록입니다.

- AnalyzerResults – DataQualityAnalyzerResult 객체의 배열이며 구조는 2,000개 이하입니다.

  각 분석기의 결과를 나타내는 DataQualityAnalyzerResult 객체의 목록입니다.

- Observations – DataQualityObservation 객체의 배열이며 구조는 50개 이하입니다.

  규칙과 분석기를 평가한 후 생성된 관찰을 나타내는 DataQualityObservation 객체의 목록입니다.

- AggregatedMetrics – DataQualityAggregatedMetrics 객체입니다.

  행 수준 결과를 기반으로 하는 통과/실패 통계를 포함하여 처리된 행 및 규칙의 총 개수를 보여주는 DataQualityAggregatedMetrics 객체 요약입니다.

DataQualityAnalyzerResult 구조

데이터 품질 분석기의 평가 결과를 설명합니다.

필드

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 분석기의 이름입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  데이터 품질 분석기에 대한 설명입니다.

- `EvaluationMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  평가 메시지입니다.
- `EvaluatedMetrics` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 숫자(double)입니다.

  분석기 평가와 관련된 지표의 맵입니다.

DataQualityObservation 구조

규칙과 분석기를 평가한 후 생성된 관찰을 설명합니다.

필드

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  데이터 품질 관찰에 대한 설명입니다.
- `MetricBasedObservation` – MetricBasedObservation 객체입니다.

  평가된 데이터 품질 지표를 기반으로 하는 관찰을 나타내는 `MetricBasedObservation` 유형의 객체입니다.

MetricBasedObservation 구조

평가된 데이터 품질 지표를 기반으로 생성된 지표 기반 관찰을 설명합니다.

필드

- `MetricName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  관찰을 생성하는 데 사용된 데이터 품질 지표의 이름입니다.
- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.

- `MetricValues` – DataQualityMetricValues 객체입니다.

  데이터 품질 지표 값의 분석을 나타내는 유형 `DataQualityMetricValues`의 객체입니다.
- `NewRules` – UTF-8 문자열의 배열입니다.

  데이터 품질 지표 값을 기반으로 관찰의 일부로 생성된 새 데이터 품질 규칙의 목록입니다.

DataQualityMetricValues 구조

과거 데이터 분석에 따른 데이터 품질 지표 값을 설명합니다.

필드

- `ActualValue` - 숫자(double)입니다.

  데이터 품질 지표의 실제 값입니다.
- `ExpectedValue` - 숫자(double)입니다.

  과거 데이터 분석에 따른 데이터 품질 지표의 예상 값입니다.
- `LowerLimit` - 숫자(double)입니다.

  과거 데이터 분석에 따른 데이터 품질 지표 값의 하한입니다.
- `UpperLimit` - 숫자(double)입니다.

  과거 데이터 분석에 따른 데이터 품질 지표 값의 상한입니다.

DataQualityRuleResult 구조

데이터 품질 규칙 평가 결과를 설명합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙의 이름입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙에 대한 설명입니다.

- `EvaluationMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  평가 메시지입니다.

- `Result` – UTF-8 문자열입니다(유효한 값: `PASS` | `FAIL` | `ERROR`).

  규칙의 통과 또는 실패 상태입니다.

- `EvaluatedMetrics` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 숫자(double)입니다.

  규칙 평가와 관련된 지표 맵입니다.

- `EvaluatedRule` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 UTF-8 문자열입니다.

  평가된 규칙입니다.

- `RuleMetrics` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 숫자(double)입니다.

  행 수준 결과를 기반으로 규칙 평가와 연결된 지표가 포함된 맵입니다.

DataQualityResultDescription 구조

데이터 품질 결과를 설명합니다.

필드

- `ResultId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 데이터 품질 결과의 고유한 결과 ID입니다.

- `DataSource` – DataSource 객체입니다.

  데이터 품질 결과와 연결된 테이블 이름입니다.

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

데이터 품질 결과와 연결된 작업 이름입니다.

- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 작업 실행 ID입니다.

- `StartedOn` – 타임스탬프입니다.

  이 데이터 품질 결과에 대한 실행이 시작된 시간입니다.

DataQualityResultFilterCriteria 구조

데이터 품질 결과를 반환하는 데 사용되는 기준입니다.

필드

- `DataSource` – DataSource 객체입니다.

  지정된 데이터 소스별로 결과를 필터링합니다. 예를 들어 AWS Glue 테이블의 모든 결과를 검색합니다.

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  지정된 작업 이름을 기준으로 결과를 필터링합니다.

- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  지정된 작업 실행 ID를 기준으로 결과를 필터링합니다.

- `StartedAfter` – 타임스탬프입니다.

  이 시간 이후에 시작된 실행을 기준으로 결과를 필터링합니다.

- `StartedBefore` – 타임스탬프입니다.

  이 시간 이전에 시작된 실행을 기준으로 결과를 필터링합니다.

DataQualityRulesetFilterCriteria 구조

데이터 품질 규칙 세트를 필터링하는 데 사용되는 기준입니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

규칙 세트 필터 기준의 이름입니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  규칙 세트 필터 기준에 대한 설명입니다.

- `CreatedBefore` – 타임스탬프입니다.

  이 날짜 이전에 생성된 규칙 세트에서 필터링합니다.

- `CreatedAfter` – 타임스탬프입니다.

  이 날짜 이후에 생성된 규칙 세트에서 필터링합니다.

- `LastModifiedBefore` – 타임스탬프입니다.

  이 날짜 이전에 마지막으로 수정된 규칙 세트에서 필터링합니다.

- `LastModifiedAfter` – 타임스탬프입니다.

  이 날짜 이후에 마지막으로 수정된 규칙 세트에서 필터링합니다.

- `TargetTable` – DataQualityTargetTable 객체입니다.

  대상 테이블의 이름 및 데이터베이스 이름입니다.

DataQualityAggregatedMetrics 구조

행 수준 결과를 기반으로 하는 통과/실패 통계를 포함하여 처리된 행 및 규칙의 총 개수를 보여주는 지표 요약입니다.

필드

- `TotalRowsProcessed` - 숫자(double)입니다.

  데이터 품질 평가 중에 처리된 행의 총 개수입니다.

- `TotalRowsPassed` - 숫자(double)입니다.

  해당하는 모든 데이터 품질 규칙을 통과한 행의 총 개수입니다.

- `TotalRowsFailed` - 숫자(double)입니다.

  하나 이상의 데이터 품질 규칙에서 실패한 행의 총 개수입니다.

- `TotalRulesProcessed` - 숫자(double)입니다.

평가된 데이터 품질 규칙의 총 개수입니다.

- `TotalRulesPassed` - 숫자(double)입니다.

  평가 기준을 통과한 데이터 품질 규칙의 총 개수입니다.

- `TotalRulesFailed` - 숫자(double)입니다.

  평가 기준을 충족하지 못한 데이터 품질 규칙의 총 개수입니다.

StatisticAnnotation 구조

통계 주석.

필드

- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  프로필 ID.

- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.

- `StatisticRecordedOn` – 타임스탬프입니다.

  주석이 달린 통계가 기록된 시점의 타임스탬프.

- `InclusionAnnotation` – TimestampedInclusionAnnotation 객체입니다.

  통계에 적용된 포함 주석.

TimestampedInclusionAnnotation 구조

타임스탬프가 지정된 포함 주석.

필드

- `Value` – UTF-8 문자열입니다(유효한 값: `INCLUDE` | `EXCLUDE`).

  포함 주석 값.

- `LastModifiedOn` – 타임스탬프입니다.

포함 주석이 마지막으로 수정된 시점의 타임스탬프.

AnnotationError 구조

실패한 주석.

필드

- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실패한 주석의 프로필 ID.

- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실패한 주석의 통계 ID.

- `FailureReason` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  주석이 실패한 이유.

DatapointInclusionAnnotation 구조

포함 주석.

필드

- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계가 속한 데이터 품질 프로필의 ID.

- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.

- `InclusionAnnotation` – UTF-8 문자열입니다(유효한 값: `INCLUDE` | `EXCLUDE`).

  통계에 적용할 포함 주석 값.

StatisticSummaryList 목록

`StatisticSummary` 목록.

StatisticSummary 객체 어레이.

`StatisticSummary` 목록.

StatisticSummary 구조

통계에 대한 요약 정보.

필드

- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.
- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  프로필 ID.
- `RunIdentifier` – RunIdentifier 객체입니다.

  실행 식별자
- `StatisticName` – Custom string pattern #16과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 이름.
- `DoubleValue` - 숫자(double)입니다.

  통계 값.
- `EvaluationLevel` – UTF-8 문자열입니다(유효한 값: `Dataset="DATASET"` | `Column="COLUMN"` | `Multicolumn="MULTICOLUMN"`).

  통계의 평가 수준. 가능한 값: `Dataset`, `Column`, `Multicolumn`.
- `ColumnsReferenced` – UTF-8 문자열의 배열입니다.

  통계에서 참조하는 열 목록.
- `ReferencedDatasets` – UTF-8 문자열의 배열입니다.

통계에서 참조하는 데이터세트 목록.

- `StatisticProperties` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 URI address multi-line string pattern와 일치하는 설명 문자열(2,048바이트 이하)입니다.

  `NameString` 및 `DescriptionString`이 포함된 `StatisticPropertiesMap`
- `RecordedOn` – 타임스탬프입니다.

  통계가 기록된 시점의 타임스탬프.
- `InclusionAnnotation` – TimestampedInclusionAnnotation 객체입니다.

  통계의 포함 주석.

RunIdentifier 구조

실행 식별자.

필드

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행 ID.
- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  작업 실행 ID.

StatisticModelResult 구조

통계 모델 결과.

필드

- `LowerBound` - 숫자(double)입니다.

  하한.
- `UpperBound` - 숫자(double)입니다.

상한.

- `PredictedValue` - 숫자(double)입니다.

  예측된 값.

- `ActualValue` - 숫자(double)입니다.

  실제 값.

- `Date` – 타임스탬프입니다.

  날짜.

- `InclusionAnnotation` – UTF-8 문자열입니다(유효한 값: `INCLUDE` | `EXCLUDE`).

  포함 주석.

운영

- StartDataQualityRulesetEvaluationRun 작업(Python: start_data_quality_ruleset_evaluation_run)
- CancelDataQualityRulesetEvaluationRun 작업(Python: cancel_data_quality_ruleset_evaluation_run)
- GetDataQualityRulesetEvaluationRun 작업(Python: get_data_quality_ruleset_evaluation_run)
- ListDataQualityRulesetEvaluationRuns 작업(Python: list_data_quality_ruleset_evaluation_runs)
- StartDataQualityRuleRecommendationRun 작업(Python: start_data_quality_rule_recommendation_run)
- CancelDataQualityRuleRecommendationRun 작업(Python: cancel_data_quality_rule_recommendation_run)
- GetDataQualityRuleRecommendationRun 작업(Python: get_data_quality_rule_recommendation_run)
- ListDataQualityRuleRecommendationRuns 작업(Python: list_data_quality_rule_recommendation_runs)
- GetDataQualityResult 작업(Python: get_data_quality_result)
- BatchGetDataQualityResult 작업(Python: batch_get_data_quality_result)
- ListDataQualityResults 작업(Python: list_data_quality_results)
- CreateDataQualityRuleset 작업(Python: create_data_quality_ruleset)

- DeleteDataQualityRuleset 작업(Python: delete_data_quality_ruleset)
- GetDataQualityRuleset 작업(Python: get_data_quality_ruleset)
- ListDataQualityRulesets 작업(Python: list_data_quality_rulesets)
- UpdateDataQualityRuleset 작업(Python: update_data_quality_ruleset)
- ListDataQualityStatistics 작업(Python: list_data_quality_statistics)
- TimestampFilter 구조
- CreateDataQualityRulesetRequest 구조
- GetDataQualityRulesetResponse 구조
- GetDataQualityResultResponse 구조
- StartDataQualityRuleRecommendationRunRequest 구조
- GetDataQualityRuleRecommendationRunResponse 구조
- BatchPutDataQualityStatisticAnnotation 작업(Python: batch_put_data_quality_statistic_annotation)
- GetDataQualityModel 작업(Python: get_data_quality_model)
- GetDataQualityModelResult 작업(Python: get_data_quality_model_result)
- ListDataQualityStatisticAnnotations 작업(Python: list_data_quality_statistic_annotations)
- PutDataQualityProfileAnnotation 작업(Python: put_data_quality_profile_annotation)

StartDataQualityRulesetEvaluationRun 작업(Python: start_data_quality_ruleset_evaluation_run)

규칙 세트 정의(권장 또는 사용자 고유)가 있는 경우 이 작업을 호출하여 데이터 소스(AWS Glue 테이블)를 기준으로 규칙 세트를 평가합니다. 평가 시 `GetDataQualityResult` API로 검색할 수 있는 결과가 계산됩니다.

요청

- `DataSource` – 필수(Required): DataSource 객체입니다.

  이 실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.
- `Role` – 필수: UTF-8 문자열입니다.

  실행 결과를 암호화하기 위해 제공되는 IAM 역할입니다.
- `NumberOfWorkers` - 숫자(정수)입니다.

실행에 사용할 G.1X 작업자 수입니다. 기본값은 5입니다.

- Timeout – 1 이상의 숫자(정수)입니다.

  실행 제한 시간(분)입니다. 실행에서 리소스를 소비하여 중지되기 전에 TIMEOUT 상태로 들어가는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.

- ClientToken – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  멱등성에 사용되며 동일한 리소스의 여러 인스턴스를 생성하거나 시작하지 않으려면 임의의 ID(예: UUID)로 설정하는 것이 좋습니다.

- AdditionalRunOptions – DataQualityEvaluationRunAdditionalRunOptions 객체입니다.

  평가 실행에 대해 지정할 수 있는 추가 실행 옵션입니다.

- RulesetNames – 필수: 1~10개 문자열의 UTF-8 문자열 배열입니다.

  규칙 세트 이름의 목록입니다.

- AdditionalDataSources – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 DataSource 객체입니다.

  평가 실행에 대해 지정할 수 있는 추가 데이터 소스에 대한 참조 문자열의 맵입니다.

응답

- RunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

오류

- InvalidInputException
- EntityNotFoundException
- OperationTimeoutException
- InternalServiceException
- ConflictException

CancelDataQualityRulesetEvaluationRun 작업(Python: cancel_data_quality_ruleset_evaluation_run)

데이터 소스에 대해 규칙 세트를 평가 중인 실행을 취소합니다.

요청

- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException

GetDataQualityRulesetEvaluationRun 작업(Python: get_data_quality_ruleset_evaluation_run)

데이터 소스에 대해 규칙 세트를 평가하는 특정 실행을 검색합니다.

요청

- RunId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

응답

- RunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

- DataSource – DataSource 객체입니다.

  이 평가 실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.

- Role – UTF-8 문자열입니다.

  실행 결과를 암호화하기 위해 제공되는 IAM 역할입니다.

- NumberOfWorkers - 숫자(정수)입니다.

  실행에 사용할 G.1X 작업자 수입니다. 기본값은 5입니다.

- Timeout – 1 이상의 숫자(정수)입니다.

  실행 제한 시간(분)입니다. 실행에서 리소스를 소비하여 중지되기 전에 TIMEOUT 상태로 들어가는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.

- AdditionalRunOptions – DataQualityEvaluationRunAdditionalRunOptions 객체입니다.

  평가 실행에 대해 지정할 수 있는 추가 실행 옵션입니다.

- Status – UTF-8 문자열입니다(유효한 값: RUNNING | FINISHED | FAILED | PENDING_EXECUTION | TIMED_OUT | CANCELING | CANCELED | RECEIVED_BY_TASKRUNNER).

  이 실행의 상태입니다.

- ErrorString – UTF-8 문자열입니다.

  실행과 연결된 오류 문자열입니다.

- StartedOn – 타임스탬프입니다.

  이 실행이 시작된 날짜와 시간입니다.

- LastModifiedOn – 타임스탬프입니다.

  타임스탬프입니다. 이 데이터 품질 규칙 권장 실행이 수정된 마지막 시점입니다.

- CompletedOn – 타임스탬프입니다.

  이 실행이 완료된 날짜와 시간입니다.

- ExecutionTime - 숫자(정수)입니다.

이 실행이 리소스를 사용한 시간(초)입니다.

- `RulesetNames` – UTF-8 문자열의 배열입니다(1~10개 문자열).

  실행에 대한 규칙 세트 이름 목록입니다. 현재 이 파라미터는 하나의 규칙 집합 이름만 사용합니다.

- `ResultIds` – UTF-8 문자열의 배열입니다(1~10개 문자열).

  실행의 데이터 품질 결과에 대한 결과 ID 목록입니다.

- `AdditionalDataSources` – 키-값 페어의 맵 배열입니다.

  각 키는 Single-line string pattern과(와) 일치하는 1~255 바이트 길이의 UTF-8 문자열입니다.

  각 값은 DataSource 객체입니다.

  평가 실행에 대해 지정할 수 있는 추가 데이터 소스에 대한 참조 문자열의 맵입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesetEvaluationRuns 작업(Python: list_data_quality_ruleset_evaluation_runs)

규칙 세트가 데이터 소스에 대해 평가되는 필터 조건을 충족하는 모든 실행을 나열합니다.

요청

- `Filter` – DataQualityRulesetEvaluationRunFilter 객체입니다.

  필터 기준입니다.

- `NextToken` – UTF-8 문자열입니다.

  결과를 오프셋하기 위한 페이지 매김 토큰입니다.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

반환할 최대 결과 수입니다.

응답

- Runs – DataQualityRulesetEvaluationRunDescription 객체의 배열입니다.

  데이터 품질 규칙 세트 실행을 나타내는 DataQualityRulesetEvaluationRunDescription 객체 목록입니다.
- NextToken – UTF-8 문자열입니다.

  추가 결과를 사용할 수 있는 경우 페이지 매김 토큰입니다.

오류

- InvalidInputException
- OperationTimeoutException
- InternalServiceException

StartDataQualityRuleRecommendationRun 작업(Python: start_data_quality_rule_recommendation_run)

어떤 규칙을 작성해야 할지 모를 때 규칙을 생성하는 데 사용되는 권장 실행을 시작합니다. AWS Glue 데이터 품질은 데이터를 분석하고 잠재적 규칙 세트에 대한 권장 사항을 제시합니다. 그런 다음 규칙 세트를 분류하고 생성된 규칙 세트를 원하는 대로 수정할 수 있습니다.

권장 사항 실행은 90일 후에 자동으로 삭제됩니다.

요청

데이터 품질 규칙 권장 사항 요청.

- DataSource – 필수(Required): DataSource 객체입니다.

  이 실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.
- Role – 필수: UTF-8 문자열입니다.

  실행 결과를 암호화하기 위해 제공되는 IAM 역할입니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  실행에 사용할 `G.1X` 작업자 수입니다. 기본값은 5입니다.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  실행 제한 시간(분)입니다. 실행에서 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.

- `CreatedRulesetName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  규칙 세트의 이름입니다.

- `DataQualitySecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

- `ClientToken` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  멱등성에 사용되며 동일한 리소스의 여러 인스턴스를 생성하거나 시작하지 않으려면 임의의 ID(예: UUID)로 설정하는 것이 좋습니다.

응답

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

오류

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConflictException`

CancelDataQualityRuleRecommendationRun 작업(Python: cancel_data_quality_rule_recommendation_run)

규칙을 생성하는 데 사용된 지정된 권장 실행을 취소합니다.

요청

- `RunId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

응답

- 무응답 파라미터.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRuleRecommendationRun 작업(Python: get_data_quality_rule_recommendation_run)

규칙을 생성하는 데 사용된 지정된 권장 실행을 가져옵니다.

요청

- `RunId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

응답

데이터 품질 규칙 권장 사항 실행에 대한 응답.

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.
- `DataSource` – DataSource 객체입니다.

  이 실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.
- `Role` – UTF-8 문자열입니다.

  실행 결과를 암호화하기 위해 제공되는 IAM 역할입니다.
- `NumberOfWorkers` - 숫자(정수)입니다.

  실행에 사용할 `G.1X` 작업자 수입니다. 기본값은 5입니다.
- `Timeout` – 1 이상의 숫자(정수)입니다.

  실행 제한 시간(분)입니다. 실행에서 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.
- `Status` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `FINISHED` | `FAILED` | `PENDING_EXECUTION` | `TIMED_OUT` | `CANCELING` | `CANCELED` | `RECEIVED_BY_TASKRUNNER`).

  이 실행의 상태입니다.
- `ErrorString` – UTF-8 문자열입니다.

  실행과 연결된 오류 문자열입니다.
- `StartedOn` – 타임스탬프입니다.

  이 실행이 시작된 날짜와 시간입니다.
- `LastModifiedOn` – 타임스탬프입니다.

  타임스탬프입니다. 이 데이터 품질 규칙 권장 실행이 수정된 마지막 시점입니다.
- `CompletedOn` – 타임스탬프입니다.

  이 실행이 완료된 날짜와 시간입니다.
- `ExecutionTime` - 숫자(정수)입니다.

  이 실행이 리소스를 사용한 시간(초)입니다.

- `RecommendedRuleset` – UTF-8 문자열입니다(1~65536바이트 이하)

  시작 규칙 권장 실행이 완료되면 권장 규칙 세트(규칙 세트)가 생성됩니다. 이 멤버는 DQDL(데이터 품질 정의 언어) 형식 규칙을 포함합니다.

- `CreatedRulesetName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행에서 생성된 규칙 세트의 이름입니다.

- `DataQualitySecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRuleRecommendationRuns 작업(Python: list_data_quality_rule_recommendation_runs)

필터 기준에 맞는 권장 실행을 나열합니다.

요청

- `Filter` – DataQualityRuleRecommendationRunFilter 객체입니다.

  필터 기준입니다.

- `NextToken` – UTF-8 문자열입니다.

  결과를 오프셋하기 위한 페이지 매김 토큰입니다.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.

응답

- Runs – DataQualityRuleRecommendationRunDescription 객체의 배열입니다.

  DataQualityRuleRecommendationRunDescription 객체의 목록.
- NextToken – UTF-8 문자열입니다.

  추가 결과를 사용할 수 있는 경우 페이지 매김 토큰입니다.

오류

- InvalidInputException
- OperationTimeoutException
- InternalServiceException

GetDataQualityResult 작업(Python: get_data_quality_result)

데이터 품질 규칙 평가 결과를 검색합니다.

요청

- ResultId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과의 고유한 결과 ID입니다.

응답

데이터 품질 결과에 대한 응답.

- ResultId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과의 고유한 결과 ID입니다.
- ProfileId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과의 프로필 ID입니다.
- Score – 1.0 이하의 숫자(실수)입니다.

  집계된 데이터 품질 점수입니다. 총 규칙 수에 전달된 규칙의 비율을 나타냅니다.

- `DataSource` – DataSource 객체입니다.

  데이터 품질 결과와 연결된 테이블입니다(있는 경우).

- `RulesetName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 규칙 세트의 이름입니다.

- `EvaluationContext` – UTF-8 문자열입니다.

  AWS Glue Studio의 작업 컨텍스트에서 캔버스의 각 노드에는 일반적으로 일종의 이름이 할당되며 데이터 품질 노드에는 이름이 지정됩니다. 여러 노드의 경우 `evaluationContext`에서 노드를 구분할 수 있습니다.

- `StartedOn` – 타임스탬프입니다.

  이 데이터 품질 결과에 대한 실행이 시작된 날짜와 시간입니다.

- `CompletedOn` – 타임스탬프입니다.

  이 데이터 품질 결과에 대한 실행이 완료된 날짜와 시간입니다.

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 작업 이름입니다(있는 경우).

- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 작업 실행 ID입니다(있는 경우).

- `RulesetEvaluationRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  규칙 세트 평가와 연결된 고유한 실행 ID입니다.

- `RuleResults` – DataQualityRuleResult 객체의 배열이며 구조는 2,000개 이하입니다.

  각 규칙의 결과를 나타내는 `DataQualityRuleResult` 객체 목록입니다.

- `AnalyzerResults` – DataQualityAnalyzerResult 객체의 배열이며 구조는 2,000개 이하입니다.

  각 분석기의 결과를 나타내는 `DataQualityAnalyzerResult` 객체의 목록입니다.

- `Observations` – DataQualityObservation 객체의 배열이며 구조는 50개 이하입니다.

  규칙과 분석기를 평가한 후 생성된 관찰을 나타내는 `DataQualityObservation` 객체의 목록입니다.

- `AggregatedMetrics` – DataQualityAggregatedMetrics 객체입니다.

  행 수준 결과를 기반으로 하는 통과/실패 통계를 포함하여 처리된 행 및 규칙의 총 개수를 보여주는 `DataQualityAggregatedMetrics` 객체 요약입니다.

오류

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `EntityNotFoundException`

BatchGetDataQualityResult 작업(Python: batch_get_data_quality_result)

지정된 결과 ID에 대한 데이터 품질 결과 목록을 검색합니다.

요청

- `ResultIds` – 필수: 1~100개 문자열의 UTF-8 문자열 배열입니다.

  데이터 품질 결과에 대한 고유한 결과 ID 목록입니다.

응답

- `Results` – 필수(Required): DataQualityResult 객체의 배열입니다.

  데이터 품질 결과를 나타내는 `DataQualityResult` 객체 목록입니다.
- `ResultsNotFound` – UTF-8 문자열의 배열입니다(1~100개 문자열).

  결과를 찾을 수 없는 결과 ID 목록입니다.

오류

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityResults 작업(Python: list_data_quality_results)

계정의 모든 데이터 품질 실행 결과를 반환합니다.

요청

- `Filter` – DataQualityResultFilterCriteria 객체입니다.

  필터 기준입니다.
- `NextToken` – UTF-8 문자열입니다.

  결과를 오프셋하기 위한 페이지 매김 토큰입니다.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.

응답

- `Results` – 필수(Required): DataQualityResultDescription 객체의 배열입니다.

  `DataQualityResultDescription` 객체의 목록.
- `NextToken` – UTF-8 문자열입니다.

  추가 결과를 사용할 수 있는 경우 페이지 매김 토큰입니다.

오류

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

CreateDataQualityRuleset 작업(Python: create_data_quality_ruleset)

지정된 AWS Glue 테이블에 적용된 DQDL 규칙을 사용하여 데이터 품질 규칙 세트를 생성합니다.

DQDL(데이터 품질 정의 언어)을 사용하여 규칙 세트를 생성합니다. 자세한 내용은 AWS Glue 개발자 안내서를 참조하세요.

요청

데이터 품질 규칙 세트를 생성하려는 요청.

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트의 고유한 이름입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  데이터 품질 규칙 세트에 대한 설명입니다.

- Ruleset – 필수: 2~65536바이트 길이의 UTF-8 문자열입니다.

  DQDL(데이터 품질 정의 언어) 규칙 세트입니다. 자세한 내용은 AWS Glue 개발자 안내서를 참조하세요.

- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트에 적용된 태그 목록입니다.

- TargetTable – DataQualityTargetTable 객체입니다.

  데이터 품질 규칙 세트와 연결된 대상 테이블입니다.

- RecommendationRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  권장 실행의 고유한 실행 ID입니다.

- DataQualitySecurityConfiguration – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

- ClientToken – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  멱등성에 사용되며 동일한 리소스의 여러 인스턴스를 생성하거나 시작하지 않으려면 임의의 ID(예: UUID)로 설정하는 것이 좋습니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트의 고유한 이름입니다.

오류

- InvalidInputException
- AlreadyExistsException
- OperationTimeoutException
- InternalServiceException
- ResourceNumberLimitExceededException

DeleteDataQualityRuleset 작업(Python: delete_data_quality_ruleset)

데이터 품질 규칙 세트를 삭제합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트의 이름입니다.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException
- OperationTimeoutException
- InternalServiceException

GetDataQualityRuleset 작업(Python: get_data_quality_ruleset)

식별자 또는 이름을 기준으로 기존 규칙 세트를 반환합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  규칙 세트의 이름입니다.

응답

데이터 품질 규칙 세트 응답을 반환합니다.

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  규칙 세트의 이름입니다.
- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  규칙 세트에 대한 설명입니다.
- Ruleset – UTF-8 문자열입니다(1~65536바이트 이하)

  DQDL(데이터 품질 정의 언어) 규칙 세트입니다. 자세한 내용은 AWS Glue 개발자 안내서를 참조하세요.
- TargetTable – DataQualityTargetTable 객체입니다.

  대상 테이블의 이름 및 데이터베이스 이름입니다.
- CreatedOn – 타임스탬프입니다.

  타임스탬프입니다. 이 데이터 품질 규칙 세트가 생성된 날짜와 시간입니다.
- LastModifiedOn – 타임스탬프입니다.

  타임스탬프입니다. 이 데이터 품질 규칙 세트가 수정된 마지막 시점입니다.
- RecommendationRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  권장 실행에서 규칙 세트가 생성되면 이 실행 ID가 생성되어 두 규칙을 서로 연결합니다.

- `DataQualitySecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesets 작업(Python: list_data_quality_rulesets)

지정된 AWS Glue 테이블 목록에 대한 페이지 매김 규칙 세트 목록을 반환합니다.

요청

- `NextToken` – UTF-8 문자열입니다.

  결과를 오프셋하기 위한 페이지 매김 토큰입니다.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.
- `Filter` – DataQualityRulesetFilterCriteria 객체입니다.

  필터 기준입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  키-값 페어 태그의 목록입니다.

응답

- `Rulesets` – DataQualityRulesetListDetails 객체의 배열입니다.

지정된 AWS Glue 테이블 목록에 대한 페이지 매김 규칙 세트 목록입니다.

- `NextToken` – UTF-8 문자열입니다.

  추가 결과를 사용할 수 있는 경우 페이지 매김 토큰입니다.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

UpdateDataQualityRuleset 작업(Python: update_data_quality_ruleset)

지정된 데이터 품질 규칙 세트를 업데이트합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트의 이름입니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  규칙 세트에 대한 설명입니다.

- `Ruleset` – UTF-8 문자열입니다(1~65536바이트 이하).

  DQDL(데이터 품질 정의 언어) 규칙 세트입니다. 자세한 내용은 AWS Glue 개발자 안내서를 참조하세요.

응답

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트의 이름입니다.

- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  규칙 세트에 대한 설명입니다.

- `Ruleset` – UTF-8 문자열입니다(1~65536바이트 이하)

  DQDL(데이터 품질 정의 언어) 규칙 세트입니다. 자세한 내용은 AWS Glue 개발자 안내서를 참조하세요.

오류

- `EntityNotFoundException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

ListDataQualityStatistics 작업(Python: list_data_quality_statistics)

데이터 품질 통계 목록을 검색합니다.

요청

- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.

- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  프로필 ID.

- `TimestampFilter` – TimestampFilter 객체입니다.

  타임스탬프 필터.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

이 요청에서 반환할 최대 결과 수입니다.

- `NextToken` – UTF-8 문자열입니다.

  결과의 다음 페이지를 요청하기 위한 페이지 매김 토큰.

응답

- `Statistics` – StatisticSummary 객체의 배열입니다.

  `StatisticSummaryList`.
- `NextToken` – UTF-8 문자열입니다.

  결과의 다음 페이지를 요청하기 위한 페이지 매김 토큰.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`

TimestampFilter 구조

타임스탬프 필터.

필드

- `RecordedBefore` – 타임스탬프입니다.

  특정 시간 이전의 통계를 결과에 포함해야 하는 경우 해당 타임스탬프.
- `RecordedAfter` – 타임스탬프입니다.

  특정 시간 이후의 통계를 결과에 포함해야 하는 경우 해당 타임스탬프.

CreateDataQualityRulesetRequest 구조

데이터 품질 규칙 세트를 생성하려는 요청.

필드

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트의 고유한 이름입니다.

- Description – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  데이터 품질 규칙 세트에 대한 설명입니다.

- Ruleset – 필수: 2~65536바이트 길이의 UTF-8 문자열입니다.

  DQDL(데이터 품질 정의 언어) 규칙 세트입니다. 자세한 내용은 AWS Glue 개발자 안내서를 참조하세요.

- Tags – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  데이터 품질 규칙 세트에 적용된 태그 목록입니다.

- TargetTable – DataQualityTargetTable 객체입니다.

  데이터 품질 규칙 세트와 연결된 대상 테이블입니다.

- RecommendationRunId – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  권장 실행의 고유한 실행 ID입니다.

- DataQualitySecurityConfiguration – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

- ClientToken – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  멱등성에 사용되며 동일한 리소스의 여러 인스턴스를 생성하거나 시작하지 않으려면 임의의 ID(예: UUID)로 설정하는 것이 좋습니다.

GetDataQualityRulesetResponse 구조

데이터 품질 규칙 세트 응답을 반환합니다.

필드

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  규칙 세트의 이름입니다.
- `Description` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  규칙 세트에 대한 설명입니다.
- `Ruleset` – UTF-8 문자열입니다(1~65536바이트 이하)

  DQDL(데이터 품질 정의 언어) 규칙 세트입니다. 자세한 내용은 AWS Glue 개발자 안내서를 참조하세요.
- `TargetTable` – DataQualityTargetTable 객체입니다.

  대상 테이블의 이름 및 데이터베이스 이름입니다.
- `CreatedOn` – 타임스탬프입니다.

  타임스탬프입니다. 이 데이터 품질 규칙 세트가 생성된 날짜와 시간입니다.
- `LastModifiedOn` – 타임스탬프입니다.

  타임스탬프입니다. 이 데이터 품질 규칙 세트가 수정된 마지막 시점입니다.
- `RecommendationRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  권장 실행에서 규칙 세트가 생성되면 이 실행 ID가 생성되어 두 규칙을 서로 연결합니다.
- `DataQualitySecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

GetDataQualityResultResponse 구조

데이터 품질 결과에 대한 응답.

필드

- `ResultId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과의 고유한 결과 ID입니다.

- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과의 프로필 ID입니다.

- `Score` – 1.0 이하의 숫자(실수)입니다.

  집계된 데이터 품질 점수입니다. 총 규칙 수에 전달된 규칙의 비율을 나타냅니다.

- `DataSource` – DataSource 객체입니다.

  데이터 품질 결과와 연결된 테이블입니다(있는 경우).

- `RulesetName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 규칙 세트의 이름입니다.

- `EvaluationContext` – UTF-8 문자열입니다.

  AWS Glue Studio의 작업 컨텍스트에서 캔버스의 각 노드에는 일반적으로 일종의 이름이 할당되며 데이터 품질 노드에는 이름이 지정됩니다. 여러 노드의 경우 `evaluationContext`에서 노드를 구분할 수 있습니다.

- `StartedOn` – 타임스탬프입니다.

  이 데이터 품질 결과에 대한 실행이 시작된 날짜와 시간입니다.

- `CompletedOn` – 타임스탬프입니다.

  이 데이터 품질 결과에 대한 실행이 완료된 날짜와 시간입니다.

- `JobName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 작업 이름입니다(있는 경우).

- `JobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 결과와 연결된 작업 실행 ID입니다(있는 경우).

- `RulesetEvaluationRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

규칙 세트 평가와 연결된 고유한 실행 ID입니다.

- `RuleResults` – DataQualityRuleResult 객체의 배열이며 구조는 2,000개 이하입니다.

  각 규칙의 결과를 나타내는 `DataQualityRuleResult` 객체 목록입니다.

- `AnalyzerResults` – DataQualityAnalyzerResult 객체의 배열이며 구조는 2,000개 이하입니다.

  각 분석기의 결과를 나타내는 `DataQualityAnalyzerResult` 객체의 목록입니다.

- `Observations` – DataQualityObservation 객체의 배열이며 구조는 50개 이하입니다.

  규칙과 분석기를 평가한 후 생성된 관찰을 나타내는 `DataQualityObservation` 객체의 목록입니다.

- `AggregatedMetrics` – DataQualityAggregatedMetrics 객체입니다.

  행 수준 결과를 기반으로 하는 통과/실패 통계를 포함하여 처리된 행 및 규칙의 총 개수를 보여주는 `DataQualityAggregatedMetrics` 객체 요약입니다.

StartDataQualityRuleRecommendationRunRequest 구조

데이터 품질 규칙 권장 사항 요청.

필드

- `DataSource` – 필수: DataSource 객체입니다.

  이 실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.

- `Role` – 필수: UTF-8 문자열입니다.

  실행 결과를 암호화하기 위해 제공되는 IAM 역할입니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  실행에 사용할 `G.1X` 작업자 수입니다. 기본값은 5입니다.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  실행 제한 시간(분)입니다. 실행에서 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.

- `CreatedRulesetName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

규칙 세트의 이름입니다.

- `DataQualitySecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

- `ClientToken` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  멱등성에 사용되며 동일한 리소스의 여러 인스턴스를 생성하거나 시작하지 않으려면 임의의 ID(예: UUID)로 설정하는 것이 좋습니다.

GetDataQualityRuleRecommendationRunResponse 구조

데이터 품질 규칙 권장 사항 실행에 대한 응답.

필드

- `RunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 실행과 연결된 고유의 실행 식별자입니다.

- `DataSource` – DataSource 객체입니다.

  이 실행과 연결된 데이터 소스(AWS Glue 테이블)입니다.

- `Role` – UTF-8 문자열입니다.

  실행 결과를 암호화하기 위해 제공되는 IAM 역할입니다.

- `NumberOfWorkers` - 숫자(정수)입니다.

  실행에 사용할 `G.1X` 작업자 수입니다. 기본값은 5입니다.

- `Timeout` – 1 이상의 숫자(정수)입니다.

  실행 제한 시간(분)입니다. 실행에서 리소스를 소비하여 중지되기 전에 `TIMEOUT` 상태로 들어가는 최대 시간입니다. 기본값은 2,880 분(48 시간)입니다.

- `Status` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `FINISHED` | `FAILED` | `PENDING_EXECUTION` | `TIMED_OUT` | `CANCELING` | `CANCELED` | `RECEIVED_BY_TASKRUNNER`).

  이 실행의 상태입니다.

- `ErrorString` – UTF-8 문자열입니다.

  실행과 연결된 오류 문자열입니다.

- `StartedOn` – 타임스탬프입니다.

  이 실행이 시작된 날짜와 시간입니다.

- `LastModifiedOn` – 타임스탬프입니다.

  타임스탬프입니다. 이 데이터 품질 규칙 권장 실행이 수정된 마지막 시점입니다.

- `CompletedOn` – 타임스탬프입니다.

  이 실행이 완료된 날짜와 시간입니다.

- `ExecutionTime` - 숫자(정수)입니다.

  이 실행이 리소스를 사용한 시간(초)입니다.

- `RecommendedRuleset` – UTF-8 문자열입니다(1~65536바이트 이하)

  시작 규칙 권장 실행이 완료되면 권장 규칙 세트(규칙 세트)가 생성됩니다. 이 멤버는 DQDL(데이터 품질 정의 언어) 형식 규칙을 포함합니다.

- `CreatedRulesetName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실행에서 생성된 규칙 세트의 이름입니다.

- `DataQualitySecurityConfiguration` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  데이터 품질 암호화 옵션을 사용하여 생성된 보안 구성의 이름입니다.

BatchPutDataQualityStatisticAnnotation 작업(Python: batch_put_data_quality_statistic_annotation)

특정 데이터 품질 통계에 대해 시간 경과에 따른 데이터 포인트에 주석을 지정합니다.

요청

- `InclusionAnnotations` – 필수(Required): DatapointInclusionAnnotation 객체의 배열입니다.

  `DatapointInclusionAnnotation`의 목록.

- `ClientToken` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  클라이언트 토큰.

응답

- `FailedInclusionAnnotations` – AnnotationError 객체의 배열입니다.

  `AnnotationError`의 목록.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

GetDataQualityModel 작업(Python: get_data_quality_model)

자세한 정보(CompletedOn, StartedOn, FailureReason)와 함께 모델의 훈련 상태를 검색합니다.

요청

- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.
- `ProfileId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  프로필 ID.

응답

- `Status` – UTF-8 문자열입니다(유효한 값: `RUNNING` | `SUCCEEDED` | `FAILED`).

  데이터 품질 모델의 훈련 상태.

- `StartedOn` – 타임스탬프입니다.

  데이터 품질 모델 훈련이 시작된 시점의 타임스탬프.

- `CompletedOn` – 타임스탬프입니다.

  데이터 품질 모델 훈련이 완료된 시점의 타임스탬프.

- `FailureReason` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  훈련 실패 이유.

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityModelResult 작업(Python: get_data_quality_model_result)

지정된 프로필 ID에 대한 통계의 예측을 검색합니다.

요청

- `StatisticId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.

- `ProfileId` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  프로필 ID.

응답

- `CompletedOn` – 타임스탬프입니다.

  데이터 품질 모델 훈련이 완료된 시점의 타임스탬프.

- `Model` – StatisticModelResult 객체의 배열입니다.

  `StatisticModelResult` 목록

오류

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityStatisticAnnotations 작업(Python: list_data_quality_statistic_annotations)

데이터 품질 통계에 대한 주석을 검색합니다.

요청

- `StatisticId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계 ID입니다.

- `ProfileId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  프로필 ID.

- `TimestampFilter` – TimestampFilter 객체입니다.

  타임스탬프 필터.

- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  이 요청에서 반환할 최대 결과 수입니다.

- `NextToken` – UTF-8 문자열입니다.

  다음 결과 세트를 검색하기 위한 페이지 매김 토큰.

응답

- Annotations – StatisticAnnotation 객체의 배열입니다.

  통계에 적용된 StatisticAnnotation의 목록

- NextToken – UTF-8 문자열입니다.

  다음 결과 세트를 검색하기 위한 페이지 매김 토큰.

오류

- InvalidInputException
- InternalServiceException

PutDataQualityProfileAnnotation 작업(Python: put_data_quality_profile_annotation)

프로필의 모든 데이터 포인트에 주석을 작성합니다.

요청

- ProfileId – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  주석을 달 데이터 품질 모니터링 프로필의 ID.

- InclusionAnnotation – 필수: UTF-8 문자열입니다(유효한 값: INCLUDE | EXCLUDE).

  프로필에 적용할 포함 주석 값.

응답

- 무응답 파라미터.

오류

- EntityNotFoundException
- InvalidInputException

- `InternalServiceException`

민감한 데이터 감지 API

민감한 데이터 탐지 API는 정형 데이터의 열과 행에서 민감한 데이터를 탐지하는 데 사용되는 API를 설명합니다.

데이터 타입

- CustomEntityType 구조

CustomEntityType 구조

정형 데이터의 열과 행에서 민감한 데이터를 감지하기 위한 사용자 정의 패턴을 나타내는 객체입니다.

필드

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  나중에 검색하거나 삭제할 수 있는 사용자 정의 패턴의 이름입니다. 이 이름은 AWS 계정별로 고유해야 합니다.
- `RegexString` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용자 정의 패턴에서 민감한 데이터를 감지하는 데 사용되는 정규식 문자열입니다.
- `ContextWords` – UTF-8 문자열의 배열입니다(1~20개 문자열).

  컨텍스트 단어 목록입니다. 이러한 컨텍스트 단어가 정규식 주변에서 발견되지 않으면 데이터는 민감한 데이터로 감지되지 않습니다.

  컨텍스트 단어가 전달되지 않으면 정규식만 검사됩니다.

운영

- CreateCustomEntityType 작업(Python: create_custom_entity_type)
- DeleteCustomEntityType 작업(Python: delete_custom_entity_type)

- GetCustomEntityType 작업(Python: get_custom_entity_type)
- BatchGetCustomEntityTypes 작업(Python: batch_get_custom_entity_types)
- ListCustomEntityTypes 작업(Python: list_custom_entity_types)

CreateCustomEntityType 작업(Python: create_custom_entity_type)

정형 데이터의 열과 행에서 민감한 데이터를 감지하는 데 사용되는 사용자 정의 패턴을 생성합니다.

생성하는 각 사용자 정의 패턴은 정규식과 컨텍스트 단어의 선택적 목록을 지정합니다. 컨텍스트 단어가 전달되지 않으면 정규식만 검사됩니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  나중에 검색하거나 삭제할 수 있는 사용자 정의 패턴의 이름입니다. 이 이름은 AWS 계정별로 고유해야 합니다.
- `RegexString` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용자 정의 패턴에서 민감한 데이터를 감지하는 데 사용되는 정규식 문자열입니다.
- `ContextWords` – UTF-8 문자열의 배열입니다(1~20개 문자열).

  컨텍스트 단어 목록입니다. 이러한 컨텍스트 단어가 정규식 주변에서 발견되지 않으면 데이터는 민감한 데이터로 감지되지 않습니다.

  컨텍스트 단어가 전달되지 않으면 정규식만 검사됩니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  사용자 지정 엔터티 유형에 적용되는 태그 목록입니다.

응답

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

생성한 사용자 정의 패턴의 이름입니다.

Errors

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

DeleteCustomEntityType 작업(Python: delete_custom_entity_type)

이름을 지정하여 사용자 정의 패턴을 삭제합니다.

요청

- Name – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제할 사용자 정의 패턴의 이름입니다.

응답

- Name – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  삭제한 사용자 정의 패턴의 이름입니다.

Errors

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`

- `OperationTimeoutException`

GetCustomEntityType 작업(Python: get_custom_entity_type)

이름을 지정하여 사용자 패턴의 세부 정보를 검색합니다.

요청

- `Name` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색할 사용자 정의 패턴의 이름입니다.

응답

- `Name` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  검색한 사용자 정의 패턴의 이름입니다.
- `RegexString` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  사용자 정의 패턴에서 민감한 데이터를 감지하는 데 사용되는 정규식 문자열입니다.
- `ContextWords` – UTF-8 문자열의 배열입니다(1~20개 문자열).

  사용자 정의 패턴을 생성할 때 지정된 경우 컨텍스트 단어 목록입니다. 이러한 컨텍스트 단어가 정규식 주변에서 발견되지 않으면 데이터는 민감한 데이터로 감지되지 않습니다.

Errors

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

BatchGetCustomEntityTypes 작업(Python: batch_get_custom_entity_types)

이름 목록으로 지정된 사용자 정의 패턴에 대한 세부 정보를 검색합니다.

요청

- `Names` – 필수(Required): 1~50개 문자열의 UTF-8 문자열의 배열입니다.

  검색할 사용자 정의 패턴의 이름 목록입니다.

응답

- `CustomEntityTypes` – CustomEntityType 객체의 배열입니다.

  생성된 사용자 정의 패턴을 나타내는 `CustomEntityType` 객체의 목록입니다.
- `CustomEntityTypesNotFound` – UTF-8 문자열의 배열입니다(1~50개 문자열).

  찾을 수 없는 사용자 정의 패턴의 이름 목록입니다.

Errors

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

ListCustomEntityTypes 작업(Python: list_custom_entity_types)

생성된 모든 사용자 정의 패턴을 나열합니다.

요청

- `NextToken` – UTF-8 문자열입니다.

  결과를 오프셋하기 위한 페이지 매김 토큰입니다.
- `MaxResults` – 1~1,000의 숫자(정수)입니다.

  반환할 최대 결과 수입니다.
- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

키-값 페어 태그의 목록입니다.

응답

- `CustomEntityTypes` – CustomEntityType 객체의 배열입니다.

  사용자 정의 패턴을 나타내는 `CustomEntityType` 객체의 목록입니다.
- `NextToken` – UTF-8 문자열입니다.

  추가 결과를 사용할 수 있는 경우 페이지 매김 토큰입니다.

Errors

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

AWS Glue에서 API 태그 지정

데이터 타입

- 태그 구조

태그 구조

`Tag` 객체는 AWS 리소스에 할당할 수 있는 레이블을 나타냅니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다.

AWS Glue의 태그 및 리소스 액세스 제어에 대한 자세한 내용은 개발자 가이드에서 AWS Glue의 AWS 태그 및 AWS Glue 리소스 ARN 지정을 참조하세요.

필드

- `key` – 1~128바이트 길이의 UTF-8 문자열입니다.

  태그 키 객체에서 태그를 생성할 때 이 키는 필수입니다. 이 키는 대/소문자를 구분하며 접두사 aws를 포함해서는 안 됩니다.

- `value` – 256바이트 이하 길이의 UTF-8 문자열입니다.

  태그 값 이 값은 객체에서 태그를 생성할 때 선택 사항입니다. 이 값은 대/소문자를 구분하며 접두사 aws를 포함해서는 안 됩니다.

운영

- TagResource 작업(Python: tag_resource)
- UntagResource 작업(Python: untag_resource)
- GetTags 작업(Python: get_tags)

TagResource 작업(Python: tag_resource)

태그를 리소스에 추가합니다. 태그는 AWS 리소스에 할당할 수 있는 레이블입니다. AWS Glue에서는 특정 리소스만 태그 지정을 할 수 있습니다. 태그 지정이 가능한 리소스에 대한 자세한 내용은 AWS Glue의 AWS 태그를 참조하세요.

태그 관련 API를 직접 호출하기 위한 태그 지정 권한 외에도 연결에서 태그 지정 API를 직접 호출할 수 있는 `glue:GetConnection` 권한과 데이터베이스에서 태그 지정 API를 직접 호출할 수 있는 `glue:GetDatabase` 권한도 필요합니다.

요청

- `ResourceArn` – 필수(Required): Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  태그를 추가할 AWS Glue 리소스의 ARN입니다. AWS Glue 리소스 ARN에 대한 자세한 내용은 AWS Glue ARN 문자열 패턴을 참조하세요.

- `TagsToAdd` – 필수(Required): 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

이 리소스에 추가할 태그입니다.

응답

- 무응답 파라미터.

오류

- `ResourceNotFoundException`

UntagResource 작업(Python: untag_resource)

통합 리소스에서 지정된 태그를 제거합니다.

요청

- `ResourceArn` – 필수(Required): Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  통합 리소스의 Amazon 리소스 이름(ARN)입니다.
- `TagsToRemove` – 필수(Required): 50개 이하의 문자열로 구성된 UTF-8 문자열의 배열입니다.

  리소스에서 제거할 메타데이터 태그의 목록입니다.

응답

- 무응답 파라미터.

오류

- `ResourceNotFoundException`

GetTags 작업(Python: get_tags)

리소스와 연결된 태그의 목록을 검색합니다.

요청

- `ResourceArn` – 필수(Required): Custom string pattern #49과(와) 일치하는 1~10,240바이트 길이의 UTF-8 문자열입니다.

  태그를 검색할 리소스의 Amazon 리소스 이름(ARN)입니다.

응답

- `Tags` – 50개 이하의 페어로 구성된 키-값 페어의 맵 배열입니다.

  각 키는 길이가 1~128바이트인 UTF-8 문자열입니다.

  각 값은 256 바이트 이하 길이의 UTF-8 문자열입니다.

  요청된 태그입니다.

오류

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

공통 데이터 형식

일반적인 데이터 유형은 AWS Glue의 기타 일반적인 데이터 유형에 대해 설명합니다.

태그 구조

`Tag` 객체는 AWS 리소스에 할당할 수 있는 레이블을 나타냅니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다.

AWS Glue의 태그 및 리소스 액세스 제어에 대한 자세한 내용은 개발자 가이드에서 AWS Glue의 AWS 태그 및 AWS Glue 리소스 ARN 지정을 참조하세요.

필드

- `key` – 1~128바이트 길이의 UTF-8 문자열입니다.

태그 키 객체에서 태그를 생성할 때 이 키는 필수입니다. 이 키는 대/소문자를 구분하며 접두사 aws를 포함해서는 안 됩니다.

- `value` – 256바이트 이하 길이의 UTF-8 문자열입니다.

  태그 값 이 값은 객체에서 태그를 생성할 때 선택 사항입니다. 이 값은 대/소문자를 구분하며 접두사 aws를 포함해서는 안 됩니다.

DecimalNumber 구조

십진수 형식의 숫자 값을 포함합니다.

필드

- `UnscaledValue` – 필수(Required): Blob입니다.

  범위가 정해지지 않은 숫자 값.

- `Scale` – 필수(Required): 숫자(정수)입니다.

  범위가 정해지지 않은 값 어디에 십진수가 오는지 결정하는 범위입니다.

ErrorDetail 구조

오류의 세부 정보를 포함합니다.

필드

- `ErrorCode` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  이 오류와 연결된 코드입니다.

- `ErrorMessage` – URI address multi-line string pattern과(와) 일치하는 2,048바이트 이하 길이의 설명 문자열입니다.

  메시지에서 오류를 설명합니다.

PropertyPredicate 구조

속성 조건자를 정의합니다.

필드

- `Key` – 값 문자열입니다(1~1,024바이트).

  속성 키입니다.
- `Value` – 값 문자열입니다(1~1,024바이트).

  속성 값입니다.
- `Comparator` – UTF-8 문자열입니다(유효 값: `EQUALS` | `GREATER_THAN` | `LESS_THAN` | `GREATER_THAN_EQUALS` | `LESS_THAN_EQUALS`).

  비교자는 이 속성을 다른 속성과 비교합니다.

ResourceUri 구조

함수 리소스의 URI입니다.

필드

- `ResourceType` – UTF-8 문자열입니다(유효한 값: `JAR` | `FILE` | `ARCHIVE`).

  리소스의 유형.
- `Uri` – URI address multi-line string pattern과(와) 일치하는 1~1,024바이트 길이의 URI(Uniform Resource Identifier)입니다.

  리소스에 액세스하는 URI입니다.

ColumnStatistics 구조

테이블 또는 파티션에 대해 생성된 열 수준 통계를 나타냅니다.

필드

- `ColumnName` – 필수(Required): Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  통계가 속한 열의 이름입니다.
- `ColumnType` – 필수(Required): Single-line string pattern과(와) 일치하는 길이 20,000바이트 이하의 유형 이름입니다.

열의 데이터 형식.

- `AnalyzedTime` – 필수(Required): 타임스탬프입니다.

  열 통계가 생성된 시간의 타임스탬프입니다.

- `StatisticsData` – 필수(Required): ColumnStatisticsData 객체입니다.

  통계 데이터 값을 포함하는 `ColumnStatisticData` 객체입니다.

ColumnStatisticsError 구조

실패한 `ColumnStatistics` 객체와 실패 이유를 캡슐화합니다.

필드

- `ColumnStatistics` – ColumnStatistics 객체입니다.

  열의 `ColumnStatistics`입니다.

- `Error` – ErrorDetail 객체입니다.

  작업 실패 이유가 포함된 오류 메시지입니다.

ColumnError 구조

실패한 열 이름과 실패 이유를 캡슐화합니다.

필드

- `ColumnName` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  실패한 열의 이름입니다.

- `Error` – ErrorDetail 객체입니다.

  작업 실패 이유가 포함된 오류 메시지입니다.

ColumnStatisticsData 구조

열 통계 데이터의 개별 유형을 포함합니다. 하나의 데이터 객체만 설정하고 `Type` 속성으로 표시해야 합니다.

필드

- `Type` – 필수: UTF-8 문자열입니다(유효한 값: `BOOLEAN` | `DATE` | `DECIMAL` | `DOUBLE` | `LONG` | `STRING` | `BINARY`).

  열 통계 데이터의 유형입니다.
- `BooleanColumnStatisticsData` – BooleanColumnStatisticsData 객체입니다.

  부울 열 통계 데이터입니다.
- `DateColumnStatisticsData` – DateColumnStatisticsData 객체입니다.

  날짜 열 통계 데이터입니다.
- `DecimalColumnStatisticsData` – DecimalColumnStatisticsData 객체입니다.

  10진수 열 통계 데이터입니다. 내부의 UnscaledValues는 빅 엔디안을 저장하는 Base64로 인코딩된 바이너리 객체로, 이 두 가지는 십진수의 스케일링되지 않은 값을 보완하는 표현입니다.
- `DoubleColumnStatisticsData` – DoubleColumnStatisticsData 객체입니다.

  실수(Double) 열 통계 데이터입니다.
- `LongColumnStatisticsData` – LongColumnStatisticsData 객체입니다.

  정수(Long) 열 통계 데이터입니다.
- `StringColumnStatisticsData` – StringColumnStatisticsData 객체입니다.

  문자열 열 통계 데이터입니다.
- `BinaryColumnStatisticsData` – BinaryColumnStatisticsData 객체입니다.

  이진수 열 통계 데이터입니다.

BooleanColumnStatisticsData 구조

부울 데이터 열에 대해 지원되는 열 통계를 정의합니다.

필드

- `NumberOfTrues` – 필수: None 이하의 숫자(정수)입니다.

  열의 true 값 수입니다.
- `NumberOfFalses` – 필수(Required): None 이하의 숫자(정수)입니다.

열의 false 값 수입니다.

- `NumberOfNulls` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 null 값 수입니다.

DateColumnStatisticsData 구조

타임스탬프 데이터 열에 대해 지원되는 열 통계를 정의합니다.

필드

- `MinimumValue` – 타임스탬프입니다.

  열에서 최저 값입니다.

- `MaximumValue` – 타임스탬프입니다.

  열에서 최고 값입니다.

- `NumberOfNulls` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 null 값 수입니다.

- `NumberOfDistinctValues` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 고유 값 수입니다.

DecimalColumnStatisticsData 구조

고정 소수점 데이터 열에 대해 지원되는 열 통계를 정의합니다.

필드

- `MinimumValue` – DecimalNumber 객체입니다.

  열에서 최저 값입니다.

- `MaximumValue` – DecimalNumber 객체입니다.

  열에서 최고 값입니다.

- `NumberOfNulls` – 필수(Required): None 이하의 숫자(정수)입니다.

열의 null 값 수입니다.

- `NumberOfDistinctValues` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 고유 값 수입니다.

DoubleColumnStatisticsData 구조

부동 소수점 데이터 열에 대해 지원되는 열 통계를 정의합니다.

필드

- `MinimumValue` - 숫자(double)입니다.

  열에서 최저 값입니다.

- `MaximumValue` - 숫자(double)입니다.

  열에서 최고 값입니다.

- `NumberOfNulls` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 null 값 수입니다.

- `NumberOfDistinctValues` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 고유 값 수입니다.

LongColumnStatisticsData 구조

정수 데이터 열에 대해 지원되는 열 통계를 정의합니다.

필드

- `MinimumValue` - 숫자(정수)입니다.

  열에서 최저 값입니다.

- `MaximumValue` - 숫자(정수)입니다.

  열에서 최고 값입니다.

- `NumberOfNulls` – 필수(Required): None 이하의 숫자(정수)입니다.

열의 null 값 수입니다.

- `NumberOfDistinctValues` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 고유 값 수입니다.

StringColumnStatisticsData 구조

문자 시퀀스 데이터 값에 대해 지원되는 열 통계를 정의합니다.

필드

- `MaximumLength` – 필수: None 이하의 숫자(정수)입니다.

  열에서 가장 긴 문자열의 크기입니다.

- `AverageLength` – 필수(Required): None 이하의 숫자(double)입니다.

  열의 평균 문자열 길이입니다.

- `NumberOfNulls` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 null 값 수입니다.

- `NumberOfDistinctValues` – 필수(Required): None 이하의 숫자(정수)입니다.

  열의 고유 값 수입니다.

BinaryColumnStatisticsData 구조

비트 시퀀스 데이터 값에 대해 지원되는 열 통계를 정의합니다.

필드

- `MaximumLength` – 필수: None 이하의 숫자(정수)입니다.

  열에서 가장 긴 비트 시퀀스의 크기입니다.

- `AverageLength` – 필수(Required): None 이하의 숫자(double)입니다.

  열의 평균 비트 시퀀스 길이입니다.

- `NumberOfNulls` – 필수(Required): None 이하의 숫자(정수)입니다.

열의 null 값 수입니다.

문자열 패턴

API는 다음 정규식을 사용하여 다양한 문자열 파라미터 및 멤버의 유효한 값이 무엇인지 정의합니다.

- 한 줄 문자열 패턴 – `"[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*"`
- URI 주소 여러 줄 문자열 패턴 – `"[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\n\t]*"`
- Logstash Grok 문자열 패턴 – `"[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\t]*"`
- 식별자 문자열 패턴 – `"[A-Za-z_][A-Za-z0-9_]*"`
- AWS IAM ARN 문자열 패턴 – `"arn:aws:iam::\d{12}:role/.*"`
- 버전 문자열 패턴 – `"^[a-zA-Z0-9-_]+$"`
- 로그 그룹 문자열 패턴 – `"[\.\-_/#A-Za-z0-9]+"`
- 로그 스트림 문자열 패턴 – `"[^:*]*"`
- 사용자 정의 문자열 패턴 #10 – `"[a-zA-Z0-9-_]+"`
- 사용자 정의 문자열 패턴 #11 – `"[-a-zA-Z0-9+=/:_]*"`
- 사용자 정의 문자열 패턴 #12 – `"[\S\s]*"`
- 사용자 정의 문자열 패턴 #13 – `".*\S.*"`
- 사용자 정의 문자열 패턴 #14 – `"[a-zA-Z0-9-=._/@]+"`
- 사용자 정의 문자열 패턴 #15 – `"[1-9][0-9]*|[1-9][0-9]*-[1-9][0-9]*"`
- 사용자 정의 문자열 패턴 #16 – `"[A-Z][A-Za-z\.]+"`
- 사용자 정의 문자열 패턴 #17 – `"[\S]*"`
- 사용자 정의 문자열 패턴 #18 – `"[\w]*"`
- 사용자 정의 문자열 패턴 #19 – `"arn:aws[a-z\-]*:iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+"`
- 사용자 정의 문자열 패턴 #20 – `"subnet-[a-z0-9]+"`
- 사용자 정의 문자열 패턴 #21 – `"\d{12}"`
- 사용자 정의 문자열 패턴 #22 – `"([a-z]+)-([a-z]+-)?([a-z]+)-[0-9]+[a-z]+"`

- 사용자 지정 문자열 패턴 #23 – '[a-zA-Z0-9.-]*'
- 사용자 정의 문자열 패턴 #24 – "arn:aws[a-z0-9\-]*:lambda:[a-z0-9\-]+:\d{12}:function:([\w\-]{1,64})"
- 사용자 정의 문자열 패턴 #25 – "^(?!(.*[.\/\\]|aws:)).*$"
- 사용자 정의 문자열 패턴 #26 – "[^\r\n]"
- 사용자 정의 문자열 패턴 #27 – "^\w+\.\w+\.\w+$"
- 사용자 정의 문자열 패턴 #28 – "^\w+\.\w+$"
- 사용자 정의 문자열 패턴 #29 – "^$|arn:aws[a-z0-9-]*:kms:.*"
- 사용자 정의 문자열 패턴 #30 – "arn:aws[^:]*:iam::[0-9]*:role/.+"
- 사용자 정의 문자열 패턴 #31 – "[\.\-_A-Za-z0-9]+"
- 사용자 정의 문자열 패턴 #32 – "^s3://([^/]+)/([^/]+/)*([^/]+)$"
- 사용자 정의 문자열 패턴 #33 – ".*"
- 사용자 지정 문자열 패턴 #34 – "^(Sun|Mon|Tue|Wed|Thu|Fri|Sat):([01]?[0-9]|2[0-3])$"
- 사용자 지정 문자열 패턴 #35 – '[a-zA-Z0-9_.-]+'
- 사용자 지정 문자열 패턴 #36 - "^arn:aws(-(cn|us-gov|iso(-[bef])?))?:secretsmanager:.*$"
- 사용자 지정 문자열 패턴 #37 - "\S+"
- 사용자 지정 문자열 패턴 #38 – "^[\x20-\x7E]*$"
- 사용자 지정 문자열 패턴 #39 – "^([a-zA-Z0-9_=]+)\.([a-zA-Z0-9_=]+)\.([a-zA-Z0-9_\-\+\/=]*)"
- 사용자 지정 문자열 패턴 #40 – "^(https?)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|]"
- 사용자 지정 문자열 패턴 #41 – "^(https?):\/\/[^\s/$.?#].[^\s]*$"
- 사용자 지정 문자열 패턴 #42 – "^subnet-[a-z0-9]+$"
- 사용자 지정 문자열 패턴 #43 – '[\p{L}\p{N}\p{P}]*'
- 사용자 지정 문자열 패턴 #44 – '[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}'
- 사용자 지정 문자열 패턴 #45 – '[a-zA-Z0-9-_$#.]+'
- 사용자 지정 문자열 패턴 #46 – '^\d{12}$'

- 사용자 지정 문자열 패턴 #47 – `'^(\w+\.)+\w+$'`
- 사용자 지정 문자열 패턴 #48 – `'^([2-3]|3[.]9)$'`
- 사용자 지정 문자열 패턴 #49 – `'arn:aws(-(cn|us-gov|iso(-[bef])?))?:glue:.*'`
- 사용자 지정 문자열 패턴 #50 – `'(^arn:aws(-(cn|us-gov|iso(-[bef])?))?:iam::\w{12}:root)'`
- 사용자 지정 문자열 패턴 #51 – `'^arn:aws(-(cn|us-gov|iso(-[bef])?))?:iam::[0-9]{12}:role/.+'`
- 사용자 지정 문자열 패턴 #52 – `'[\s\S]*'`
- 사용자 지정 문자열 패턴 #53 – `'([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n"'= ;])*'`
- 사용자 지정 문자열 패턴 #54 – `'^[A-Z\_]+$'`
- 사용자 지정 문자열 패턴 #55 – `'^[A-Za-z0-9]+$'`
- 사용자 지정 문자열 패턴 #56 – `'[*A-Za-z0-9_-]*'`
- 사용자 지정 문자열 패턴 #57 – `'([\u0020-\u007E\r\s\n])*'`
- 사용자 지정 문자열 패턴 #58 – `'[A-Za-z0-9_-]*'`
- 사용자 지정 문자열 패턴 #59 – `'([\u0009\u000B\u000C\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF])*'`
- 사용자 지정 문자열 패턴 #60 – `'([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\s])*'`
- 사용자 지정 문자열 패턴 #61 – `'([^\r\n])*'`

예외

이 섹션에서는 문제의 원인을 찾고 해결하는 데 사용할 수 있는 AWS Glue 예외사항에 대해 설명합니다. 기계 학습과 관련된 예외에 관한 HTTP 오류 코드 및 문자열에 대한 자세한 내용은 the section called “AWS Glue 기계 학습 예외 사항” 단원을 참조하십시오.

AccessDeniedException 구조

리소스 액세스가 거부됩니다.

필드

- `Message` – UTF-8 문자열입니다.

문제를 설명하는 메시지

AlreadyExistsException 구조

생성되는 또는 추가되는 리소스가 이미 있습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ConcurrentModificationException 구조

두 절차로 동시에 리소스를 수정하고자 합니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ConcurrentRunsExceededException 구조

다양한 작업이 동시에 실행됩니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

CrawlerNotRunningException 구조

지정한 크롤러가 실행되고 있지 않습니다.

필드

- `Message` – UTF-8 문자열입니다.

문제를 설명하는 메시지

CrawlerRunningException 구조

크롤러가 이미 수행되고 있기 때문에 작업을 수행할 수 없습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

CrawlerStoppingException 구조

지정한 크롤러가 중지되고 있습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

EntityNotFoundException 구조

지정한 개체가 존재하지 않습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지
- `FromFederationSource` – 부울입니다.

  예외가 페더레이션된 소스와 관련이 있는지 여부를 나타냅니다.

FederationSourceException 구조

페더레이션 소스에 실패했습니다.

필드

- FederationSourceErrorCode – UTF-8 문자열입니다(유효한 값: AccessDeniedException | EntityNotFoundException | InvalidCredentialsException | InvalidInputException | InvalidResponseException | OperationTimeoutException | OperationNotSupportedException | InternalServiceException | PartialFailureException | ThrottlingException).

  문제의 오류 코드입니다.
- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지입니다.

FederationSourceRetryableException 구조

페더레이션 소스에 실패했지만 작업이 재시도될 수 있습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

GlueEncryptionException 구조

암호화 연산에 실패했습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지입니다.

IdempotentParameterMismatchException 구조

두 가지 다른 기록과 연결된 동일한 고유 식별자입니다.

필드

- Message – UTF-8 문자열입니다.

문제를 설명하는 메시지

IllegalWorkflowStateException 구조

워크플로가 요청된 작업을 수행할 수 있는 잘못된 상태입니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

InternalServiceException 구조

내부 서비스 오류가 발생했습니다.

필드

- Message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

InvalidExecutionEngineException 구조

알려지지 않거나 무효한 엔진이 지정되었습니다.

필드

- message – UTF-8 문자열입니다.

  문제를 설명하는 메시지

InvalidInputException 구조

제공한 입력이 유효하지 않았습니다.

필드

- Message – UTF-8 문자열입니다.

문제를 설명하는 메시지

- `FromFederationSource` – 부울입니다.

  예외가 페더레이션된 소스와 관련이 있는지 여부를 나타냅니다.

InvalidStateException 구조

데이터 상태가 잘못되었음을 나타내는 오류입니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

InvalidTaskStatusTransitionException 구조

하나의 작업이 다음 실패한 작업으로의 적절한 전환.

필드

- `message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

JobDefinitionErrorException 구조

작업 정의가 유효하지 않습니다.

필드

- `message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

JobRunInTerminalStateException 구조

작업 실행의 터미널 상태는 실패를 암시합니다.

필드

- `message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

JobRunInvalidStateTransitionException 구조

작업 실행으로 원본 상태에서 대상 상태로의 무효한 전환이 발생했습니다.

필드

- `jobRunId` – Single-line string pattern과(와) 일치하는 1~255바이트 길이의 UTF-8 문자열입니다.

  문제의 작업 실행 ID입니다.

- `message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

- `sourceState` – UTF-8 문자열입니다(유효한 값: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  소스 데이터

- `targetState` – UTF-8 문자열입니다(유효한 값: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  대상 상태

JobRunNotInTerminalStateException 구조

터미널 상태에서 작업이 실행되지 않습니다.

필드

- `message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

LateRunnerException 구조

작업 실행이 늦습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

NoScheduleException 구조

적용 가능한 일정이 없습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

OperationTimeoutException 구조

작업 시간이 초과되었습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ResourceNotReadyException 구조

리소스가 트랜잭션에 대해 준비되지 않았습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ResourceNumberLimitExceededException 구조

리소스의 숫자 제한을 초과했습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

SchedulerNotRunningException 구조

지정한 스케줄러가 실행되고 있지 않습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

SchedulerRunningException 구조

지정한 스케줄러가 이미 실행되고 있습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

SchedulerTransitioningException 구조

지정한 스케줄러가 변환되고 있습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

UnrecognizedRunnerException 구조

작업 실행이 인식되지 않습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

ValidationException 구조

값이 유효하지 않을 수 있습니다.

필드

- `Message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

VersionMismatchException 구조

버전 충돌이 있었습니다.

필드

- `message` – UTF-8 문자열입니다.

  문제를 설명하는 메시지

AWS SDK를 사용한 AWS Glue API 코드 예제

다음 코드 예제는 AWS Glue를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작

AWS Glue 시작

다음 코드 예제에서는 AWS Glue를 사용하여 시작하는 방법을 보여 줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
```

```
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListJobs를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_glue.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }

            Aws::Glue::Model::ListJobsOutcome listRunsOutcome =
 glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListJobs를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListJobs를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
  return JobNames;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListJobs를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import boto3
from botocore.exceptions import ClientError
```

```
def hello_glue():
    """
    Lists the job definitions in your AWS Glue account, using the AWS SDK for
 Python (Boto3).
    """
    try:
        # Create the Glue client
        glue = boto3.client("glue")

        # List the jobs, limiting the results to 10 per page
        paginator = glue.get_paginator("get_jobs")
        response_iterator = paginator.paginate(
            PaginationConfig={"MaxItems": 10, "PageSize": 10}
        )

        # Print the job names
        print("Here are the jobs in your account:")
        for page in response_iterator:
            for job in page["Jobs"]:
                print(f"\t{job['Name']}")

    except ClientError as e:
        print(f"Error: {e}")


if __name__ == "__main__":
    hello_glue()
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListJobs를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
    @logger.info('Here are the Glue jobs in your account:')

    paginator = @client.get_jobs(max_results: 10)
    jobs = []

    paginator.each_page do |page|
      jobs.concat(page.jobs)
    end

    if jobs.empty?
      @logger.info("You don't have any Glue jobs.")
    else
      jobs.each do |job|
        @logger.info("- #{job.name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListJobs를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListJobs을 참조하십시오.

AWS Glue API 코드 예제

- AWS SDK를 사용한 AWS Glue 코드 예제
  - AWS Glue 시작
  - AWS SDK가 있는 AWS Glue의 기본 사항 알아보기
  - AWS SDK를 사용한 AWS Glue 작업
    - AWS SDK와 함께 CreateCrawler사용
    - AWS SDK 또는 CLI와 함께 CreateJob 사용
    - AWS SDK와 함께 DeleteCrawler사용
    - AWS SDK와 함께 DeleteDatabase사용
    - AWS SDK 또는 CLI와 함께 DeleteJob 사용
    - AWS SDK와 함께 DeleteTable사용
    - AWS SDK와 함께 GetCrawler사용

- AWS SDK와 함께 GetDatabase사용
- AWS SDK 또는 CLI와 함께 GetDatabases 사용
- AWS SDK 또는 CLI와 함께 GetJob 사용
- AWS SDK 또는 CLI와 함께 GetJobRun 사용
- AWS SDK 또는 CLI와 함께 GetJobRuns 사용
- AWS SDK 또는 CLI와 함께 GetTables 사용
- AWS SDK와 함께 ListJobs사용
- AWS SDK 또는 CLI와 함께 StartCrawler 사용
- AWS SDK 또는 CLI와 함께 StartJobRun 사용

AWS SDK를 사용한 AWS Glue 코드 예제

다음 코드 예제에서는 AWS SDK에서 AWS Glue의 기본 사항을 사용하는 방법을 보여줍니다.

예시

- AWS Glue 시작
- AWS SDK가 있는 AWS Glue의 기본 사항 알아보기
- AWS SDK를 사용한 AWS Glue 작업
  - AWS SDK와 함께 CreateCrawler사용
  - AWS SDK 또는 CLI와 함께 CreateJob 사용
  - AWS SDK와 함께 DeleteCrawler사용
  - AWS SDK와 함께 DeleteDatabase사용
  - AWS SDK 또는 CLI와 함께 DeleteJob 사용
  - AWS SDK와 함께 DeleteTable사용
  - AWS SDK와 함께 GetCrawler사용
  - AWS SDK와 함께 GetDatabase사용
  - AWS SDK 또는 CLI와 함께 GetDatabases 사용
  - AWS SDK 또는 CLI와 함께 GetJob 사용
  - AWS SDK 또는 CLI와 함께 GetJobRun 사용
  - AWS SDK 또는 CLI와 함께 GetJobRuns 사용
  - AWS SDK 또는 CLI와 함께 GetTables 사용

- AWS SDK와 함께 ListJobs사용
- AWS SDK 또는 CLI와 함께 StartCrawler 사용
- AWS SDK 또는 CLI와 함께 StartJobRun 사용

AWS Glue 시작

다음 코드 예제에서는 AWS Glue의 사용을 시작하는 방법을 보여 줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
            .Build();
```

```
        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListJobs를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                 # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_glue.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
```

```
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome =
 glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListJobs를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListJobs를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
  return JobNames;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListJobs를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import boto3
from botocore.exceptions import ClientError


def hello_glue():
    """
    Lists the job definitions in your AWS Glue account, using the AWS SDK for
 Python (Boto3).
    """
    try:
        # Create the Glue client
        glue = boto3.client("glue")
```

```
        # List the jobs, limiting the results to 10 per page
        paginator = glue.get_paginator("get_jobs")
        response_iterator = paginator.paginate(
            PaginationConfig={"MaxItems": 10, "PageSize": 10}
        )

        # Print the job names
        print("Here are the jobs in your account:")
        for page in response_iterator:
            for job in page["Jobs"]:
                print(f"\t{job['Name']}")

    except ClientError as e:
        print(f"Error: {e}")


if __name__ == "__main__":
    hello_glue()
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListJobs를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
  def initialize(client)
    @client = client
```

```
    @logger = Logger.new($stdout)
  end

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
    @logger.info('Here are the Glue jobs in your account:')

    paginator = @client.get_jobs(max_results: 10)
    jobs = []

    paginator.each_page do |page|
      jobs.concat(page.jobs)
    end

    if jobs.empty?
      @logger.info("You don't have any Glue jobs.")
    else
      jobs.each do |job|
        @logger.info("- #{job.name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListJobs를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- API에 대한 세부 정보는 Rust용 AWS SDK API 참조의 ListJobs를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK가 있는 AWS Glue의 기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 퍼블릭 Amazon S3 버킷을 크롤링하고 CSV 형식의 메타데이터 데이터베이스를 생성하는 크롤러를 생성합니다.
- AWS Glue Data Catalog의 데이터베이스 및 테이블에 대한 정보를 나열합니다.
- 작업을 생성하여 S3 버킷에서 CSV 데이터를 추출하고, 데이터를 변환하며, JSON 형식의 출력을 다른 S3 버킷으로 로드합니다.
- 작업 실행에 대한 정보를 나열하고 변환된 데이터를 확인하며 리소스를 정리합니다.

자세한 내용은 자습서:AWS Glue Studio 시작하기를 참조하세요.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오에서 사용되는 AWS Glue 함수를 래핑하는 클래스를 생성합니다.

```
using System.Net;

namespace GlueActions;

public class GlueWrapper
{
    private readonly IAmazonGlue _amazonGlue;

    /// <summary>
    /// Constructor for the AWS Glue actions wrapper.
    /// </summary>
    /// <param name="amazonGlue"></param>
    public GlueWrapper(IAmazonGlue amazonGlue)
    {
        _amazonGlue = amazonGlue;
    }

    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
 executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
 (Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
```

```
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
```

```
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete the AWS Glue database.
```

```
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
```

```
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
 found.");
        return null;
    }


    /// <summary>
    /// Get information about the state of an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A value describing the state of the crawler.</returns>
    public async Task<CrawlerState> GetCrawlerStateAsync(string crawlerName)
    {
        var response = await _amazonGlue.GetCrawlerAsync(
            new GetCrawlerRequest { Name = crawlerName });
        return response.Crawler.State;
    }


    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };
```

```
        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }


    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }


    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
```

```
            });
        }

        return jobRuns;
    }


    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }


    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }
```

```
        return jobNames;
    }


    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };
```

```
        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }

}
```

시나리오를 실행하는 클래스를 생성합니다.

```
global using Amazon.Glue;
global using GlueActions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.Glue.Model;
using Amazon.S3;
using Amazon.S3.Model;

namespace GlueBasics;

public class GlueBasics
{
    private static ILogger logger = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
```

```
            services.AddAWSService<IAmazonGlue>()
            .AddTransient<GlueWrapper>()
            .AddTransient<UiWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
        .CreateLogger<GlueBasics>();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // These values are stored in settings.json
        // Once you have run the CDK script to deploy the resources,
        // edit the file to set "BucketName", "RoleName", and "ScriptURL"
        // to the appropriate values. Also set "CrawlerName" to the name
        // you want to give the crawler when it is created.
        string bucketName = _configuration["BucketName"]!;
        string bucketUrl = _configuration["BucketUrl"]!;
        string crawlerName = _configuration["CrawlerName"]!;
        string roleName = _configuration["RoleName"]!;
        string sourceData = _configuration["SourceData"]!;
        string dbName = _configuration["DbName"]!;
        string cron = _configuration["Cron"]!;
        string scriptUrl = _configuration["ScriptURL"]!;
        string jobName = _configuration["JobName"]!;

        var wrapper = host.Services.GetRequiredService<GlueWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UiWrapper>();

        uiWrapper.DisplayOverview();
        uiWrapper.PressEnter();

        // Create the crawler and wait for it to be ready.
        uiWrapper.DisplayTitle("Create AWS Glue crawler");
        Console.WriteLine("Let's begin by creating the AWS Glue crawler.");

        var crawlerDescription = "Crawler created for the AWS Glue Basics
scenario.";
```

```
        var crawlerCreated = await wrapper.CreateCrawlerAsync(crawlerName,
crawlerDescription, roleName, cron, sourceData, dbName);
        if (crawlerCreated)
        {
            Console.WriteLine($"The crawler: {crawlerName} has been created. Now
let's wait until it's ready.");
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't create crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.DisplayTitle("Start AWS Glue crawler");
        Console.WriteLine("Now let's wait until the crawler has successfully
started.");
        var crawlerStarted = await wrapper.StartCrawlerAsync(crawlerName);
        if (crawlerStarted)
        {
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't start the crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.PressEnter();
```

```
        Console.WriteLine($"\nLet's take a look at the database: {dbName}");
        var database = await wrapper.GetDatabaseAsync(dbName);

        if (database != null)
        {
            uiWrapper.DisplayTitle($"{database.Name} Details");
            Console.WriteLine($"{database.Name} created on
{database.CreateTime}");
            Console.WriteLine(database.Description);
        }

        uiWrapper.PressEnter();

        var tables = await wrapper.GetTablesAsync(dbName);
        if (tables.Count > 0)
        {
            tables.ForEach(table =>
            {
                Console.WriteLine($"{table.Name}\tCreated:
{table.CreateTime}\tUpdated: {table.UpdateTime}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create AWS Glue job");
        Console.WriteLine("Creating a new AWS Glue job.");
        var description = "An AWS Glue job created using the AWS SDK for .NET";
        await wrapper.CreateJobAsync(dbName, tables[0].Name, bucketUrl, jobName,
roleName, description, scriptUrl);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Starting AWS Glue job");
        Console.WriteLine("Starting the new AWS Glue job...");
        var jobRunId = await wrapper.StartJobRunAsync(jobName, dbName,
tables[0].Name, bucketName);
        var jobRunComplete = false;
        var jobRun = new JobRun();
        do
        {
            jobRun = await wrapper.GetJobRunAsync(jobName, jobRunId);
            if (jobRun.JobRunState == "SUCCEEDED" || jobRun.JobRunState ==
"STOPPED" ||
```

```
                jobRun.JobRunState == "FAILED" || jobRun.JobRunState ==
"TIMEOUT")
            {
                jobRunComplete = true;
            }
        } while (!jobRunComplete);

        uiWrapper.DisplayTitle($"Data in {bucketName}");

        // Get the list of data stored in the S3 bucket.
        var s3Client = new AmazonS3Client();

        var response = await s3Client.ListObjectsAsync(new ListObjectsRequest
{ BucketName = bucketName });
        response.S3Objects.ForEach(s3Object =>
        {
            Console.WriteLine(s3Object.Key);
        });

        uiWrapper.DisplayTitle("AWS Glue jobs");
        var jobNames = await wrapper.ListJobsAsync();
        jobNames.ForEach(jobName =>
        {
            Console.WriteLine(jobName);
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Get AWS Glue job run information");
        Console.WriteLine("Getting information about the AWS Glue job.");
        var jobRuns = await wrapper.GetJobRunsAsync(jobName);

        jobRuns.ForEach(jobRun =>
        {

Console.WriteLine($"{jobRun.JobName}\t{jobRun.JobRunState}\t{jobRun.CompletedOn}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Deleting resources");
        Console.WriteLine("Deleting the AWS Glue job used by the example.");
        await wrapper.DeleteJobAsync(jobName);
```

```
        Console.WriteLine("Deleting the tables from the database.");
        tables.ForEach(async table =>
        {
            await wrapper.DeleteTableAsync(dbName, table.Name);
        });

        Console.WriteLine("Deleting the database.");
        await wrapper.DeleteDatabaseAsync(dbName);

        Console.WriteLine("Deleting the AWS Glue crawler.");
        await wrapper.DeleteCrawlerAsync(crawlerName);

        Console.WriteLine("The AWS Glue scenario has completed.");
        uiWrapper.PressEnter();
    }
}


namespace GlueBasics;

public class UiWrapper
{
    public readonly string SepBar = new string('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Amazon Glue: get started with crawlers and jobs");

        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t 1. Create a crawler, pass it the IAM role and the
 URL to the public S3 bucket that contains the source data");
        Console.WriteLine("\t 2. Start the crawler.");
        Console.WriteLine("\t 3. Get the database created by the crawler and the
 tables in the database.");
        Console.WriteLine("\t 4. Create a job.");
        Console.WriteLine("\t 5. Start a job run.");
        Console.WriteLine("\t 6. Wait for the job run to complete.");
        Console.WriteLine("\t 7. Show the data stored in the bucket.");
        Console.WriteLine("\t 8. List jobs for the account.");
        Console.WriteLine("\t 9. Get job run details for the job that was run.");
```

```
        Console.WriteLine("\t10. Delete the demo job.");
        Console.WriteLine("\t11. Delete the database and tables created for the
 demo.");
        Console.WriteLine("\t12. Delete the crawler.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPlease press <Enter> to continue. ");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to center on the screen.</param>
    /// <returns>The string padded to make it center on the screen.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.

- CreateCrawler
- CreateJob
- DeleteCrawler
- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
 \\sa runGettingStartedWithGlueScenario()
 \param bucketName: An S3 bucket created in the setup.
 \param roleName: An AWS Identity and Access Management (IAM) role created in the
 setup.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
```

```
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String
 &bucketName,
                                                     const Aws::String &roleName,
                                                     const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::Glue::GlueClient client(clientConfig);

    Aws::String roleArn;
    if (!getRoleArn(roleName, roleArn, clientConfig)) {
        std::cerr << "Error getting role ARN for role." << std::endl;
        return false;
    }

    // 1. Upload the job script to the S3 bucket.
    {
        std::cout << "Uploading the job script '"
                  << AwsDoc::Glue::PYTHON_SCRIPT
                  << "'." << std::endl;

        if (!AwsDoc::Glue::uploadFile(bucketName,
                                      AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                      AwsDoc::Glue::PYTHON_SCRIPT,
                                      clientConfig)) {
            std::cerr << "Error uploading the job file." << std::endl;
            return false;
        }
    }

    // 2. Create a crawler.
    {
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);
```

```
        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
        }
    }

    // 3. Get a crawler.
    {
        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 4. Start a crawler.
    {
        Aws::Glue::Model::StartCrawlerRequest request;
```

```
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                        crawlerState)
                              << ". After " << iterations
                              << " seconds elapsed."
                              << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
```

```
                    std::cerr << "Error getting crawler.  "
                              << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 5. Get a database.
    {
        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
```

```
                        clientConfig);
            return false;
        }
    }

    // 6. Get tables.
    Aws::String tableName;
    {
        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        std::vector<Aws::Glue::Model::Table> all_tables;
        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
                all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
```

```
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
    }

    // 7. Create a job.
    {
        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 8. Start a job run.
    {
        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);
```

```
       Aws::Map<Aws::String, Aws::String> arguments;
       arguments["--input_database"] = CRAWLER_DATABASE_NAME;
       arguments["--input_table"] = tableName;
       arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
       request.SetArguments(arguments);

       Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully started the job." << std::endl;

           Aws::String jobRunId = outcome.GetResult().GetJobRunId();

           int iterator = 0;
           bool done = false;
           while (!done) {
               ++iterator;
               std::this_thread::sleep_for(std::chrono::seconds(1));
               Aws::Glue::Model::GetJobRunRequest jobRunRequest;
               jobRunRequest.SetJobName(JOB_NAME);
               jobRunRequest.SetRunId(jobRunId);

               Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                       jobRunRequest);

               if (jobRunOutcome.IsSuccess()) {
                   const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                   Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                   if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                       (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                       (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                       std::cerr << "Error running job. "
                                 << jobRun.GetErrorMessage()
                                 << std::endl;
```

```
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                             jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                         clientConfig);
            return false;
        }
    }

    // 9. List the output data stored in the S3 bucket.
```

```
   {
        Aws::S3::S3Client s3Client;
        Aws::S3::Model::ListObjectsV2Request request;
        request.SetBucket(bucketName);
        request.SetPrefix(OUTPUT_FILE_PREFIX);

        Aws::String continuationToken; // Used for pagination.
        std::vector<Aws::S3::Model::Object> allObjects;
        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome =
s3Client.ListObjectsV2(
                    request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::S3::Model::Object> &objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
                continuationToken =
outcome.GetResult().GetNextContinuationToken();
            }
            else {
                std::cerr << "Error listing objects. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!continuationToken.empty());

        std::cout << "Data from your job is in " << allObjects.size() <<
                  " files in the S3 bucket, " << bucketName << "." << std::endl;

        for (size_t i = 0; i < allObjects.size(); ++i) {
            std::cout << "    " << i + 1 << ". " << allObjects[i].GetKey()
                      << std::endl;
        }

        int objectIndex = askQuestionForIntRange(
                std::string(
                        "Enter the number of a block to download it and see the
first ") +
```

```
                std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
                " lines of JSON output in the block: ", 1,
                static_cast<int>(allObjects.size()));

        Aws::String objectKey = allObjects[objectIndex - 1].GetKey();

        std::stringstream stringStream;
        if (getObjectFromBucket(bucketName, objectKey, stringStream,
                                clientConfig)) {
            for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; +
+i) {
                std::string line;
                std::getline(stringStream, line);
                std::cout << "    " << line << std::endl;
            }
        }
        else {
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                         clientConfig);
            return false;
        }
    }

    // 10. List all the jobs.
    Aws::String jobName;
    {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
 jobNames.end());
```

```
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
            }
        } while (!nextToken.empty());
        std::cout << "Your account has " << allJobNames.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < allJobNames.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobNames[i] << std::endl;
        }
        int jobIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobNames.size()) +
                " to see the list of runs for a job: ",
                1, static_cast<int>(allJobNames.size()));

        jobName = allJobNames[jobIndex - 1];
    }

    // 11. Get the job runs for a job.
    Aws::String jobRunID;
    if (!jobName.empty()) {
        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
jobRuns.end());
```

```
                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "There are " << allJobRuns.size() << " runs in the job '"
                  <<
                  jobName << "'." << std::endl;

        for (size_t i = 0; i < allJobRuns.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobRuns[i].GetJobName()
                      << std::endl;
        }

        int runIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobRuns.size()) +
                " to see details for a run: ",
                1, static_cast<int>(allJobRuns.size()));
        jobRunID = allJobRuns[runIndex - 1].GetId();
    }

    // 12. Get a single job run.
    if (!jobRunID.empty()) {
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
```

```
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
 \\sa deleteAssets()
 \param crawler: Name of an AWS Glue crawler.
 \param database: The name of an AWS Glue database.
 \param job: The name of an AWS Glue job.
 \param bucketName: The name of an S3 bucket.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
 &database,
                                const Aws::String &job, const Aws::String
 &bucketName,
                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::Glue::GlueClient client(clientConfig);
    bool result = true;

    // 13. Delete a job.
    if (!job.empty()) {
        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
 outcome.GetError().GetMessage()
```

```
                    << std::endl;
            result = false;
        }
    }

    // 14. Delete a database.
    if (!database.empty()) {
        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                    << std::endl;
            result = false;
        }
    }

    // 15. Delete a crawler.
    if (!crawler.empty()) {
        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                    << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
```

```
                                                        clientConfig);
    return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
 \\sa uploadFile()
 \param bucketName: An S3 bucket created in the setup.
 \param filePath: The path of the file to upload.
 \param fileName The name for the uploaded file.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                         const Aws::String &filePath,
                         const Aws::String &fileName,
                         const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          filePath.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    else {
        std::cout << "Added object '" << filePath << "' to bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \\sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    bool result = true;
    do {
        if (!continuationToken.empty()) {
            listObjectsRequest.SetContinuationToken(continuationToken);
        }

        Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
 client.ListObjectsV2(
                listObjectsRequest);

        if (listObjectsOutcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
 listObjectsOutcome.GetResult().GetContents();
            if (!objects.empty()) {
                Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
                deleteObjectsRequest.SetBucket(bucketName);

                std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
                for (const Aws::S3::Model::Object &object: objects) {
                    objectIdentifiers.push_back(
                            Aws::S3::Model::ObjectIdentifier().WithKey(
```

```
                                    object.GetKey()));
                }
                Aws::S3::Model::Delete objectsDelete;
                objectsDelete.SetObjects(objectIdentifiers);
                objectsDelete.SetQuiet(true);
                deleteObjectsRequest.SetDelete(objectsDelete);

                Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                        client.DeleteObjects(deleteObjectsRequest);

                if (!deleteObjectsOutcome.IsSuccess()) {
                    std::cerr << "Error deleting objects. " <<
                              deleteObjectsOutcome.GetError().GetMessage() <<
 std::endl;
                    result = false;
                    break;
                }
                else {
                    std::cout << "Successfully deleted the objects." <<
 std::endl;

                }
            }
            else {
                std::cout << "No objects to delete in '" << bucketName << "'."
                          << std::endl;
            }

            continuationToken =
 listObjectsOutcome.GetResult().GetNextContinuationToken();
        }
        else {
            std::cerr << "Error listing objects. "
                      << listObjectsOutcome.GetError().GetMessage() << std::endl;
            result = false;
            break;
        }
    } while (!continuationToken.empty());

    return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
```

```
 \\sa getObjectFromBucket()
 \param bucketName: The S3 bucket name.
 \param objectKey: The object's name.
 \param objectStream: A stream to receive the retrieved data.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);


    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." <<
 std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
    }
    else {
        std::cerr << "Error retrieving object. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 항목을 참조하세요.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase

- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
```

```
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri> <bucketNameSc>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3
 permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
 that contains data (for example, s3://<bucket name>/read).
                cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
                locationUri - The location of the database (you can find this
 file in resources folder).
                bucketNameSc - The Amazon S3 bucket name used when creating a job
                """;
```

```
        if (args.length != 9) {
            System.out.println(usage);
            return;
        }
        Scanner scanner = new Scanner(System.in);
        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println("""
            AWS Glue is a fully managed extract, transform, and load (ETL)
service provided by Amazon
            Web Services (AWS). It is designed to simplify the process of
building, running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known
as the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.

            Lets get started.

            """);
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName,
crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        try {
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
```

```
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
     * Creates a Glue database with the specified name and location URI.
     *
     * @param glueClient  The Glue client to use for the database creation.
     * @param dbName      The name of the database to create.
     * @param locationUri The location URI for the database.
     */
```

```
   public static void createDatabase(GlueClient glueClient, String dbName,
String locationUri) {
       try {
           DatabaseInput input = DatabaseInput.builder()
               .description("Built with the AWS SDK for Java V2")
               .name(dbName)
               .locationUri(locationUri)
               .build();

           CreateDatabaseRequest request = CreateDatabaseRequest.builder()
               .databaseInput(input)
               .build();

           glueClient.createDatabase(request);
           System.out.println(dbName + " was successfully created");

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the
data source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler
will store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   public static void createGlueCrawler(GlueClient glueClient,
                                        String iam,
                                        String s3Path,
                                        String cron,
                                        String dbName,
                                        String crawlerName) {

       try {
           S3Target s3Target = S3Target.builder()
```

```
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to
be in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue
service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            boolean ready = false;
            while (!ready) {
```

```
                GetCrawlerResponse response =
glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact
with the service
     * @param databaseName the name of the database to retrieve
```

```
    * @throws GlueException if there is an error retrieving the database from
the AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " +
createDate);

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table
names from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
```

```
                .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }


    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                                String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
```

```
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    public static void createJob(GlueClient glueClient, String jobName, String
iam, String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
```

```
            throw e;
        }
    }


    /**
     * Retrieves and prints information about all the jobs in the Glue data
catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue
service
     */
    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the job runs for a given Glue job and prints the status of the
job runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();
```

```
            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
```

```
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact
with the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName)
{
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
```

```
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
 DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreateCrawler

- CreateJob
- DeleteCrawler
- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

공용 Amazon Simple Storage Service (S3) 버킷을 크롤링하고 검색한 CSV 형식의 데이터를 설명하는 메타데이터 데이터베이스를 생성하는 크롤러를 만들고 실행합니다.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
```

```
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
    return true;
  } catch {
    return false;
  }
};

/**
 * @param {{ createCrawler: import('../../../actions/create-
crawler.js').createCrawler}} actions
 */
const makeCreateCrawlerStep = (actions) => async (context) => {
```

```
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH,
    );

    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};

/**
 * @param {(name: string) => Promise<import('@aws-sdk/client-
glue').GetCrawlerCommandOutput>} getCrawler
 * @param {string} crawlerName
 */
const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
    return;
  }

  log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
  await wait(waitTimeInSeconds);
  return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
    await startCrawler(process.env.CRAWLER_NAME);
```

```
    log("Crawler started.", { type: "success" });

    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
  };
```

AWS Glue Data Catalog의 데이터베이스 및 테이블에 대한 정보를 나열합니다.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };

/**
```

```
 * @param {{ getTables: () => Promise<import('@aws-sdk/client-
glue').GetTablesCommandOutput}} config
 */
const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };
```

소스 Amazon S3 버킷에서 CSV 데이터를 추출하고, 필드를 제거하고 이름을 변경하여 변환하고, JSON 형식의 출력을 다른 Amazon S3 버킷으로 로드하는 작업을 만들고 실행합니다.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
```

```
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY,
    );
    log("Job created.", { type: "success" });

    return { ...context };
  };

/**
 * @param {(name: string, runId: string) => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput> }  getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "TIMEOUT":
    case "STOPPED":
    case "ERROR":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`,
      );
    case "SUCCEEDED":
      return;
```

```
    default:
      break;
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`,
  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};

/**
 * @param {{ prompter: { prompt: () => Promise<{ shouldOpen: boolean }>} }}
 context
 */
const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });

  if (shouldOpen) {
    return open(
      `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME} to
 view the output.`,
    );
  }
};

const makeStartJobRunStep =
  ({ startJobRun, getJobRun }) =>
  async (context) => {
    log("Starting job.");
    const { JobRunId } = await startJobRun(
      process.env.JOB_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_NAME,
      process.env.BUCKET_NAME,
    );
    log("Job started.", { type: "success" });

    log("Waiting for job to finish running. This can take a while.");
    await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
```

```
    log("Job run succeeded.", { type: "success" });

    await promptToOpen(context);

    return { ...context };
  };
```

작업 실행에 대한 정보를 나열하고 변환된 데이터 중 일부를 볼 수 있습니다.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};

/**
 * @typedef {{ prompter: { prompt: () => Promise<{jobName: string}> } }} Context
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput>} getJobRun
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunsCommandOutput} getJobRuns
 */
```

```
/**
 *
 * @param {getJobRun} getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};

/**
 *
 * @param {{getJobRuns: getJobRuns, getJobRun: getJobRun }} funcs
 */
const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (/** @type { Context } */ context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }

    return { ...context };
  };
```

데모 중에 생성된 모든 리소스를 삭제합니다.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });
```

```
  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};

const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};

/**
 *
 * @param {import('../../../actions/delete-job.js').deleteJob} deleteJobFn
 * @param {string[]} jobNames
 * @param {{ prompter: { prompt: () => Promise<any> }}} context
 */
const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
  /**
   * @type {{ selectedJobNames: string[] }}
```

```
   */
  const { selectedJobNames } = await context.prompter.prompt({
    name: "selectedJobNames",
    type: "checkbox",
    message: "Let's clean up jobs. Select jobs to delete.",
    choices: jobNames,
  });

  if (selectedJobNames.length === 0) {
    log("No jobs selected.");
  } else {
    log("Deleting jobs.");
    await Promise.all(
      selectedJobNames.map((n) => deleteJobFn(n).catch(console.error)),
    );
    log("Jobs deleted.", { type: "success" });
  }
};

/**
 * @param {{
 *   listJobs: import('../../../actions/list-jobs.js').listJobs,
 *   deleteJob: import('../../../actions/delete-job.js').deleteJob
 * }} config
 */
const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-table.js').deleteTable} deleteTable
 * @param {string} databaseName
 * @param {string[]} tableNames
 */
const deleteTables = (deleteTable, databaseName, tableNames) =>
  Promise.all(
    tableNames.map((tableName) =>
```

```
      deleteTable(databaseName, tableName).catch(console.error),
    ),
  );

/**
 * @param {{
 *  getTables: import('../../../actions/get-tables.js').getTables,
 *  deleteTable: import('../../../actions/delete-table.js').deleteTable
 * }} config
 */
const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}}} context
   */
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null }),
    );

    if (TableList && TableList.length > 0) {
      /**
       * @type {{ tableNames: string[] }}
       */
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });

      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
  };

/**
```

```
 * @param {import('../../../actions/delete-database.js').deleteDatabase}
 deleteDatabase
 * @param {string[]} databaseNames
 */
const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error)),
  );

/**
 * @param {{
 *   getDatabases: import('../../../actions/get-databases.js').getDatabases
 *   deleteDatabase: import('../../../actions/delete-database.js').deleteDatabase
 * }} config
 */
const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}} context
   */
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      /** @type {{ dbNames: string[] }} */
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
    }

    return { ...context };
  };
```

```
const cleanUpCrawlerStep = async (context) => {
  log("Deleting crawler.");

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log("Crawler is already deleted.");
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service
 (Amazon S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
```

```
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}
```

```
suspend fun createCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
```

```
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}
```

```
suspend fun startJob(jobNameVal: String?) {
    val runRequest =
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
```

```
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }
```

```
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;
```

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);
```

```
        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);
```

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
```

```
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path):
Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
```

```
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
```

```
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

```
    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 항목을 참조하세요.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables

- ListJobs
- StartCrawler
- StartJobRun

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오에 사용된 AWS Glue 함수를 래핑하는 클래스를 만듭니다.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
```

```
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

```
    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
```

```
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_job_run(self, name, input_database, input_table,
output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
the
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
```

```
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
```

```
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

시나리오를 실행하는 클래스를 생성합니다.

```
class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job
 and use
    them to transform data from CSV to JSON format.
    """

    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM)
 role
                                  that AWS Glue can assume to gain access to the
                                  resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output
 data
                            from AWS Glue job runs.
        """
```

```
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an
animated
        spinner.

        :param seconds: The number of seconds to wait.
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = "|/-\\"
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1 / tick)
            waited += 1

    def upload_job_script(self, job_script):
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the
AWS Glue
        job to transform data.

        :param job_script: The relative path to the job script.
        """
        try:
            self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
            print(f"Uploaded job script '{job_script}' to the example bucket.")
        except S3UploadFailedError as err:
            logger.error("Couldn't upload job script. Here's why: %s", err)
            raise

    def run(self, crawler_name, db_name, db_prefix, data_source, job_script,
job_name):
        """
        Runs the scenario. This is an interactive experience that runs at a
command
        prompt and asks you for input throughout.
```

```
        :param crawler_name: The name of the crawler used in the scenario. If the
                             crawler does not exist, it is created.
        :param db_name: The name to give the metadata database created by the
crawler.
        :param db_prefix: The prefix to give tables added to the database by the
                          crawler.
        :param data_source: The location of the data source that is targeted by
the
                            crawler and extracted during job runs.
        :param job_script: The job script that is used to transform data during
job
                           runs.
        :param job_name: The name to give the job definition that is created
during the
                         scenario.
        """
        wrapper = GlueWrapper(self.glue_client)
        print(f"Checking for crawler {crawler_name}.")
        crawler = wrapper.get_crawler(crawler_name)
        if crawler is None:
            print(f"Creating crawler {crawler_name}.")
            wrapper.create_crawler(
                crawler_name,
                self.glue_service_role.arn,
                db_name,
                db_prefix,
                data_source,
            )
            print(f"Created crawler {crawler_name}.")
            crawler = wrapper.get_crawler(crawler_name)
        pprint(crawler)
        print("-" * 88)

        print(
            f"When you run the crawler, it crawls data stored in {data_source}
and "
            f"creates a metadata database in the AWS Glue Data Catalog that
describes "
            f"the data in the data source."
        )
        print("In this example, the source data is in CSV format.")
        ready = False
        while not ready:
```

```
            ready = Question.ask_question(
                "Ready to start the crawler? (y/n) ", Question.is_yesno
            )
        wrapper.start_crawler(crawler_name)
        print("Let's wait for the crawler to run. This typically takes a few
minutes.")
        crawler_state = None
        while crawler_state != "READY":
            self.wait(10)
            crawler = wrapper.get_crawler(crawler_name)
            crawler_state = crawler["State"]
            print(f"Crawler is {crawler['State']}.")
        print("-" * 88)

        database = wrapper.get_database(db_name)
        print(f"The crawler created database {db_name}:")
        pprint(database)
        print(f"The database contains these tables:")
        tables = wrapper.get_tables(db_name)
        for index, table in enumerate(tables):
            print(f"\t{index + 1}. {table['Name']}")
        table_index = Question.ask_question(
            f"Enter the number of a table to see more detail: ",
            Question.is_int,
            Question.in_range(1, len(tables)),
        )
        pprint(tables[table_index - 1])
        print("-" * 88)

        print(f"Creating job definition {job_name}.")
        wrapper.create_job(
            job_name,
            "Getting started example job.",
            self.glue_service_role.arn,
            f"s3://{self.glue_bucket.name}/{job_script}",
        )
        print("Created job definition.")
        print(
            f"When you run the job, it extracts data from {data_source},
transforms it "
            f"by using the {job_script} script, and loads the output into "
            f"S3 bucket {self.glue_bucket.name}."
        )
        print(
```

```
            "In this example, the data is transformed from CSV to JSON, and only
a few "
            "fields are included in the output."
        )
        job_run_status = None
        if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
            job_run_id = wrapper.start_job_run(
                job_name, db_name, tables[0]["Name"], self.glue_bucket.name
            )
            print(f"Job {job_name} started. Let's wait for it to run.")
            while job_run_status not in ["SUCCEEDED", "STOPPED", "FAILED",
"TIMEOUT"]:
                self.wait(10)
                job_run = wrapper.get_job_run(job_name, job_run_id)
                job_run_status = job_run["JobRunState"]
                print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
        print("-" * 88)

        if job_run_status == "SUCCEEDED":
            print(
                f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':"
            )
            try:
                keys = [
                    obj.key for obj in
self.glue_bucket.objects.filter(Prefix="run-")
                ]
                for index, key in enumerate(keys):
                    print(f"\t{index + 1}: {key}")
                lines = 4
                key_index = Question.ask_question(
                    f"Enter the number of a block to download it and see the
first {lines} "
                    f"lines of JSON output in the block: ",
                    Question.is_int,
                    Question.in_range(1, len(keys)),
                )
                job_data = io.BytesIO()
                self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
                job_data.seek(0)
                for _ in range(lines):
                    print(job_data.readline().decode("utf-8"))
            except ClientError as err:
```

```
                logger.error(
                    "Couldn't get job run data. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
            print("-" * 88)

        job_names = wrapper.list_jobs()
        if job_names:
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list
of runs for "
                f"a job: ",
                Question.is_int,
                Question.in_range(1, len(job_names)),
            )
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index -
1]}:")
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}"
                    )
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int,
                    Question.in_range(1, len(job_runs)),
                )
                pprint(job_runs[run_index - 1])
            else:
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
        print("-" * 88)

        print(
```

```
            f"Let's clean up. During this example we created job definition
 '{job_name}'."
        )
        if Question.ask_question(
            "Do you want to delete the definition and all runs? (y/n) ",
            Question.is_yesno,
        ):
            wrapper.delete_job(job_name)
            print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these
 tables:")
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
            Question.is_yesno,
        ):
            for table in tables:
                wrapper.delete_table(db_name, table["Name"])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno
        ):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
        print("-" * 88)


def parse_args(args):
    """
    Parse command line arguments.

    :param args: The command line arguments.
    :return: The parsed arguments.
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
 scenario. "
        "Before you run this scenario, set up scaffold resources by running "
        "'python scaffold.py deploy'."
```

```
    )
    parser.add_argument(
        "role_name",
        help="The name of an IAM role that AWS Glue can assume. This role must
 grant access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.",
    )
    parser.add_argument(
        "bucket_name",
        help="The name of an S3 bucket that AWS Glue can access to get the job
 script and "
        "put job results.",
    )
    parser.add_argument(
        "--job_script",
        default="flight_etl_job_script.py",
        help="The name of the job script file that is used in the scenario.",
    )
    return parser.parse_args(args)


def main():
    args = parse_args(sys.argv[1:])
    try:
        print("-" * 88)
        print(
            "Welcome to the AWS Glue getting started with crawlers and jobs
 scenario."
        )
        print("-" * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client("glue"),
            boto3.resource("iam").Role(args.role_name),
            boto3.resource("s3").Bucket(args.bucket_name),
        )
        scenario.upload_job_script(args.job_script)
        scenario.run(
            "doc-example-crawler",
            "doc-example-database",
            "doc-example-",
            "s3://crawler-public-us-east-1/flight/2016/csv",
            args.job_script,
            "doc-example-job",
```

```
        )
        print("-" * 88)
        print(
            "To destroy scaffold resources, including the IAM role and S3 bucket
 "
            "used in this scenario, run 'python scaffold.py destroy'."
        )
        print("\nThanks for watching!")
        print("-" * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")
```

AWS Glue에서 사용된 ETL 스크립트를 생성하여 작업 실행 중에 데이터를 추출, 전환, 적재(ETL)합니다.

```
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

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
```

```
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)
```

```
job.commit()
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오에 사용된 AWS Glue 함수를 래핑하는 클래스를 만듭니다.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
 the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
```

```
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
```

```
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
```

```
  # @param output_bucket_name [String] The name of the output S3 bucket for the
job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
```

```
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
```

```
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
                                          body: file,
                                          bucket: bucket_resource.name,
                                          key: file_path
                                        })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end
end
```

시나리오를 실행하는 클래스를 생성합니다.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end

  private

  def setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    new_step(1, 'Create a crawler')
    crawler = wrapper.get_crawler(crawler_name)
    unless crawler
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
 db_prefix, data_source)
      puts "Successfully created #{crawler_name}."
    end
    wrapper.start_crawler(crawler_name)
```

```
    monitor_crawler(wrapper, crawler_name)
  end

  def monitor_crawler(wrapper, crawler_name)
    new_step(2, 'Monitor Crawler')
    crawler_state = nil
    until crawler_state == 'READY'
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]['state']
      print "Crawler status: #{crawler_state}".yellow
    end
  end

  def query_database(wrapper, _crawler_name, db_name)
    new_step(3, 'Query the database.')
    wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    puts "Database contains tables: #{wrapper.get_tables(db_name).map { |t|
 t['name'] }}"
  end

  def create_and_run_job(wrapper, job_script, job_name, db_name)
    new_step(4, 'Create and run job.')
    wrapper.upload_job_script(job_script, @glue_bucket)
    wrapper.create_job(job_name, 'ETL Job', @glue_service_role.arn, "s3://
#{@glue_bucket.name}/#{job_script}")
    run_job(wrapper, job_name, db_name)
  end

  def run_job(wrapper, job_name, db_name)
    new_step(5, 'Run the job.')
    wrapper.start_job_run(job_name, db_name, wrapper.get_tables(db_name)[0]
['name'], @glue_bucket.name)
    job_run_status = nil
    until %w[SUCCEEDED FAILED STOPPED].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]['job_run_state']
      print "Job #{job_name} status: #{job_run_status}".yellow
    end
  end
end
```

```
def main
  banner('../../helpers/banner.txt')
  puts 'Starting AWS Glue demo...'

  # Load resource names from YAML.
  resource_names = YAML.load_file('resource_names.yaml')

  # Setup services and resources.
  iam_role = Aws::IAM::Resource.new(region: 'us-
east-1').role(resource_names['glue_service_role'])
  s3_bucket = Aws::S3::Resource.new(region: 'us-
east-1').bucket(resource_names['glue_bucket'])

  # Instantiate scenario and run.
  scenario = GlueCrawlerJobScenario.new(Aws::Glue::Client.new(region: 'us-
east-1'), iam_role, s3_bucket, @logger)
  random_suffix = rand(10**4)
  scenario.run("crawler-#{random_suffix}", "db-#{random_suffix}", "prefix-
#{random_suffix}-", 's3://data_source',
               'job_script.py', "job-#{random_suffix}")

  puts 'Demo complete.'
end
```

AWS Glue에서 사용된 ETL 스크립트를 생성하여 작업 실행 중에 데이터를 추출, 전환, 적재(ETL)합니다.

```
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

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
```

```
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
```

```
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

공용 Amazon Simple Storage Service (S3) 버킷을 크롤링하고 검색한 CSV 형식의 데이터를 설명하는 메타데이터 데이터베이스를 생성하는 크롤러를 만들고 실행합니다.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;

        let start_crawler =
    glue.start_crawler().name(self.crawler()).send().await;
```

```
        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

AWS Glue Data Catalog의 데이터베이스 및 테이블에 대한 정보를 나열합니다.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
 database".into()))?;

        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

소스 Amazon S3 버킷에서 CSV 데이터를 추출하고, 필드를 제거하고 이름을 변경하여 변환하고, JSON 형식의 출력을 다른 Amazon S3 버킷으로 로드하는 작업을 만들고 실행합니다.

```
        let create_job = glue
            .create_job()
            .name(self.job())
```

```
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;

        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

데모 중에 생성된 모든 리소스를 삭제합니다.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }

        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob

- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`Package.swift` 파일.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "glue-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
```

```
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "glue-scenario",
            dependencies: [
                .product(name: "AWSGlue", package: "aws-sdk-swift"),
                .product(name: "AWSS3", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Swift 코드 파일 `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to demonstrate
// creating and using crawlers and jobs using AWS Glue.
//
// 0. Upload the Python job script to Amazon S3 so it can be used when
//    calling `startJobRun()` later.
// 1. Create a crawler, pass it the IAM role and the URL of the public Amazon
//    S3 bucket that contains the source data:
//    s3://crawler-public-us-east-1/flight/2016/csv.
// 2. Start the crawler. This takes time, so after starting it, use a loop
//    that calls `getCrawler()` until the state is "READY".
// 3. Get the database created by the crawler, and the tables in the
//    database. Display them to the user.
// 4. Create a job. Pass it the IAM role and the URL to a Python ETL script
//    previously uploaded to the user's S3 bucket.
// 5. Start a job run, passing the following custom arguments. These are
//    expected by the ETL script, so must exactly match.
//    * `--input_database: <name of the database created by the crawler>`
//    * `--input_table: <name of the table created by the crawler>`
```

```
//    * `--output_bucket_url: <URL to the scaffold bucket created for the
//      user>`
// 6. Loop and get the job run until it returns one of the following states:
//    "SUCCEEDED", "STOPPED", "FAILED", or "TIMEOUT".
// 7. Output data is stored in a group of files in the user's S3 bucket.
//    Either direct the user to their location or download a file and display
//    the results inline.
// 8. List the jobs for the user's account.
// 9. Get job run details for a job run.
// 10. Delete the demo job.
// 11. Delete the database and tables created by the example.
// 12. Delete the crawler created by the example.

import ArgumentParser
import AWSS3
import Foundation
import Smithy

import AWSClientRuntime
import AWSGlue

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS IAM role to use for AWS Glue calls.")
    var role: String

    @Option(help: "The Amazon S3 bucket to use for this example.")
    var bucket: String

    @Option(help: "The Amazon S3 URL of the data to crawl.")
    var s3url: String = "s3://crawler-public-us-east-1/flight/2016/csv"

    @Option(help: "The Python script to run as a job with AWS Glue.")
    var script: String = "./flight_etl_job_script.py"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(help: "A prefix string to use when naming tables.")
    var tablePrefix = "swift-glue-basics-table"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
```

```
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "glue-scenario",
        abstract: """
        Demonstrates various features of AWS Glue.
        """,
        discussion: """
        An example showing how to use AWS Glue to create, run, and monitor
        crawlers and jobs.
        """
    )

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Upload a file to an Amazon S3 bucket.
    ///
    /// - Parameters:
    ///   - s3Client: The S3 client to use when uploading the file.
    ///   - path: The local path of the source file to upload.
    ///   - toBucket: The name of the S3 bucket into which to upload the file.
    ///   - key: The key (name) to give the file in the S3 bucket.
```

```
    ///
    /// - Returns: `true` if the file is uploaded successfully, otherwise
`false`.
    func uploadFile(s3Client: S3Client, path: String, toBucket: String, key:
String) async -> Bool {
        do {
            let fileData: Data = try Data(contentsOf: URL(fileURLWithPath: path))
            let dataStream = ByteStream.data(fileData)
            _ = try await s3Client.putObject(
                input: PutObjectInput(
                    body: dataStream,
                    bucket: toBucket,
                    key: key
                )
            )
        } catch {
            print("*** An unexpected error occurred uploading the script to the
Amazon S3 bucket \"\(bucket)\".")
            return false
        }

        return true
    }

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
    ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
    ///   - databaseName: The name of an AWS Glue database to operate on.
    ///
    /// - Returns: `true` if the crawler is created successfully, otherwise
`false`.
    func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole:
String,
                       s3Path: String, cronSchedule: String, databaseName:
String) async -> Bool {
        let s3Target = GlueClientTypes.S3Target(path: s3url)
        let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

        do {
```

```
            _ = try await glueClient.createCrawler(
                input: CreateCrawlerInput(
                    databaseName: databaseName,
                    description: "Created by the AWS SDK for Swift Scenario
Example for AWS Glue.",
                    name: crawlerName,
                    role: iamRole,
                    schedule: cronSchedule,
                    tablePrefix: tablePrefix,
                    targets: targetList
                )
            )
        } catch _ as AlreadyExistsException {
            print("*** A crawler named \"\(crawlerName)\" already exists.")
            return false
        } catch _ as OperationTimeoutException {
            print("*** The attempt to create the AWS Glue crawler timed out.")
            return false
        } catch {
            print("*** An unexpected error occurred creating the AWS Glue
crawler: \(error.localizedDescription)")
            return false
        }

        return true
    }

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }
```

```
    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise
`false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
            }
```

```
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
        }
    }

    /// Wait until the specified crawler is ready to run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to wait for.
    ///
    /// - Returns: `true` if the crawler is ready, `false` if the client is
    ///   stopping (and will therefore never be ready).
    func waitUntilCrawlerReady(glueClient: GlueClient, name: String) async ->
Bool {
        while true {
            let state = await getCrawlerState(glueClient: glueClient, name: name)

            if state == .ready {
                return true
            } else if state == .stopping {
                return false
            }
            Thread.sleep(forTimeInterval: 4)
        }
    }

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
```

```
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }

    /// Return a list of the AWS Glue jobs listed on the user's account.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - maxJobs: The maximum number of jobs to return (default: 100).
    ///
    /// - Returns: An array of strings listing the names of all available AWS
    ///   Glue jobs.
    func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
        var jobList: [String] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.listJobs(
                    input: ListJobsInput(
                        maxResults: maxJobs,
                        nextToken: nextToken
                    )
                )
```

```
                guard let jobs = output.jobNames else {
                    return jobList
                }

                jobList = jobList + jobs
                nextToken = output.nextToken
            } catch {
                return jobList
            }
        } while (nextToken != nil)

        return jobList
    }

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }

    /// Create an AWS Glue database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name to give the new database.
    ///   - location: The URL of the source data to use with AWS Glue.
    ///
    /// - Returns: `true` if the database is created successfully, otherwise
`false`.
    func createDatabase(glueClient: GlueClient, name databaseName: String,
location: String) async -> Bool {
        let databaseInput = GlueClientTypes.DatabaseInput(
```

```
            description: "Created by the AWS SDK for Swift Glue basic scenario
example.",
            locationUri: location,
            name: databaseName
        )

        do {
            _ = try await glueClient.createDatabase(
                input: CreateDatabaseInput(
                    databaseInput: databaseInput
                )
            )
        } catch {
            return false
        }

        return true
    }

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
```

```
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async
 -> [GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
```

```
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient:
glueClient, databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables.

            do {
                _ = try await glueClient.batchDeleteTable(
                    input: BatchDeleteTableInput(
                        databaseName: databaseName,
                        tablesToDelete: tableNames
                    )
                )
            } catch {
                print("*** Unable to delete the tables.")
            }
            return true
        }

        // Delete the database itself.

        do {
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
        }
        return true
    }
```

```
   /// Start an AWS Glue job run.
   ///
   /// - Parameters:
   ///   - glueClient: The AWS Glue client to use.
   ///   - jobName: The name of the job to run.
   ///   - databaseName: The name of the AWS Glue database to run the job
against.
   ///   - tableName: The name of the table in the database to run the job
against.
   ///   - outputURL: The AWS S3 URI of the bucket location into which to
   ///     write the resulting output.
   ///
   /// - Returns: `true` if the job run is started successfully, otherwise
`false`.
   func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
String,
                    tableName: String, outputURL: String) async -> String? {
       do {
           let output = try await glueClient.startJobRun(
               input: StartJobRunInput(
                   arguments: [
                       "--input_database": databaseName,
                       "--input_table": tableName,
                       "--output_bucket_url": outputURL
                   ],
                   jobName: jobName,
                   numberOfWorkers: 10,
                   workerType: .g1x
               )
           )

           guard let id = output.jobRunId else {
               return nil
           }

           return id
       } catch {
           return nil
       }
   }

   /// Get information about a specific AWS Glue job run.
   ///
```

```
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String)
 async -> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // A name to give the Python script upon upload to the Amazon S3
        // bucket.
        let scriptName = "jobscript.py"

        // Schedule string in `cron` format, as described here:
        // https://docs.aws.amazon.com/glue/latest/dg/monitor-data-warehouse-
schedule.html
        let cron = "cron(15 12 * * ? *)"

        let glueConfig = try await GlueClient.GlueClientConfiguration(region:
 awsRegion)
        let glueClient = GlueClient(config: glueConfig)

        let s3Config = try await S3Client.S3ClientConfiguration(region:
 awsRegion)
        let s3Client = S3Client(config: s3Config)

        // Create random names for things that need them.
```

```
        let crawlerName = tempName(prefix: "swift-glue-basics-crawler")
        let databaseName = tempName(prefix: "swift-glue-basics-db")

        // Create a name for the AWS Glue job.

        let jobName = tempName(prefix: "scenario-job")

        // The URL of the Python script on S3.

        let scriptURL = "s3://\(bucket)/\(scriptName)"

        print("Welcome to the AWS SDK for Swift basic scenario for AWS Glue!")

        //========================================================================
        // 0. Upload the Python script to the target bucket so it's available
        //    for use by the Amazon Glue service.
        //========================================================================

        print("Uploading the Python script: \(script) as key \(scriptName)")
        print("Destination bucket: \(bucket)")
        if !(await uploadFile(s3Client: s3Client, path: script, toBucket: bucket,
key: scriptName)) {
            return
        }

        //========================================================================
        // 1. Create the database and crawler using the randomized names
        //    generated previously.
        //========================================================================

        print("Creating database \"\(databaseName)\"...")
        if !(await createDatabase(glueClient: glueClient, name: databaseName,
location: s3url)) {
            print("*** Unable to create the database.")
            return
        }

        print("Creating crawler \"\(crawlerName)\"...")
        if !(await createCrawler(glueClient: glueClient, crawlerName:
crawlerName,
                                 iamRole: role, s3Path: s3url, cronSchedule:
cron,
                                 databaseName: databaseName)) {
            return
```

```
        }

        //====================================================================
        // 2. Start the crawler, then wait for it to be ready.
        //====================================================================

        print("Starting the crawler and waiting until it's ready...")
        if !(await startCrawler(glueClient: glueClient, name: crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        if !(await waitUntilCrawlerReady(glueClient: glueClient, name:
crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
        }

        //====================================================================
        // 3. Get the database and table created by the crawler.
        //====================================================================

        print("Getting the crawler's database...")
        let database = await getDatabase(glueClient: glueClient, name:
databaseName)
        let tableList = await getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)

        print("Found \(tableList.count) table(s):")
        for table in tableList {
            print("  \(table.name ?? "<unnamed>")")
        }

        if tableList.count != 1 {
            print("*** Incorrect number of tables found. There should only be
one.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        guard let tableName = tableList[0].name else {
            print("*** Table is unnamed.")
```

```
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //===================================================================
        // 4. Create a job.
        //===================================================================

        print("Creating a job...")
        if !(await createJob(glueClient: glueClient, name: jobName, role: role,
                             scriptLocation: scriptURL)) {
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //===================================================================
        // 5. Start a job run.
        //===================================================================

        print("Starting the job...")

        // Construct the Amazon S3 URL for the job run's output. This is in
        // the bucket specified on the command line, with a folder name that's
        // unique for this job run.

        let timeStamp = Date().timeIntervalSince1970
        let jobPath = "\(jobName)-\(Int(timeStamp))"
        let outputURL = "s3://\(bucket)/\(jobPath)"

        // Start the job run.

        let jobRunID = await startJobRun(glueClient: glueClient, name: jobName,
                                         databaseName: databaseName,
                                         tableName: tableName,
                                         outputURL: outputURL)

        guard let jobRunID else {
            print("*** Job run ID is invalid.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
```

```
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
 withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //========================================================================
        // 6. Wait for the job run to indicate that the run is complete.
        //========================================================================

        print("Waiting for job run to end...")

        var jobRunFinished = false
        var jobRunState: GlueClientTypes.JobRunState

        repeat {
            let jobRun = await getJobRun(glueClient: glueClient, name: jobName,
 id: jobRunID)
            guard let jobRun else {
                print("*** Unable to get the job run.")
                _ = await deleteJob(glueClient: glueClient, name: jobName)
                _ = await deleteDatabase(glueClient: glueClient, name:
 databaseName, withTables: true)
                _ = await deleteCrawler(glueClient: glueClient, name:
 crawlerName)
                return
            }
            jobRunState = jobRun.jobRunState ?? .failed

            //
========================================================================
            // 7. Output where to find the data if the job run was successful.
            //    If the job run failed for any reason, output an appropriate
            //    error message.
            //
========================================================================

            switch jobRunState {
                case .succeeded:
                    print("Job run succeeded. JSON files are in the Amazon S3
 path:")
                    print("    \(outputURL)")
                    jobRunFinished = true
                case .stopped:
```

```
                    jobRunFinished = true
                case .error:
                    print("*** Error: Job run ended in an error.
\(jobRun.errorMessage ?? "")")
                    jobRunFinished = true
                case .failed:
                    print("*** Error: Job run failed. \(jobRun.errorMessage ??
"")")
                    jobRunFinished = true
                case .timeout:
                    print("*** Warning: Job run timed out.")
                    jobRunFinished = true
                default:
                    Thread.sleep(forTimeInterval: 0.25)
            }
        } while jobRunFinished != true

        //=========================================================================
        // 8. List the jobs for the user's account.
        //=========================================================================

        print("\nThe account has the following jobs:")
        let jobs = await listJobs(glueClient: glueClient)

        if jobs.count == 0 {
            print("    <no jobs found>")
        } else {
            for job in jobs {
                print("    \(job)")
            }
        }

        //=========================================================================
        // 9. Get the job run details for a job run.
        //=========================================================================

        print("Information about the job run:")
        let jobRun = await getJobRun(glueClient: glueClient, name: jobName, id:
jobRunID)

        guard let jobRun else {
            print("*** Unable to retrieve the job run.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
```

```
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
 withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        let startDate = jobRun.startedOn ?? Date(timeIntervalSince1970: 0)
        let endDate = jobRun.completedOn ?? Date(timeIntervalSince1970: 0)
        let dateFormatter: DateFormatter = DateFormatter()
        dateFormatter.dateStyle = .long
        dateFormatter.timeStyle = .long

        print("    Started at: \(dateFormatter.string(from: startDate))")
        print("  Completed at: \(dateFormatter.string(from: endDate))")

        //========================================================================
        // 10. Delete the job.
        //========================================================================

        print("\nDeleting the job...")
        _ = await deleteJob(glueClient: glueClient, name: jobName)

        //========================================================================
        // 11. Delete the database and tables created by this example.
        //========================================================================

        print("Deleting the database...")
        _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
 withTables: true)

        //========================================================================
        // 12. Delete the crawler.
        //========================================================================

        print("Deleting the crawler...")
        if !(await deleteCrawler(glueClient: glueClient, name: crawlerName)) {
            return
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
```

```
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Swift API 참조의 다음 주제를 참조하세요.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 AWS Glue 작업

다음 코드 예제는 AWS SDK를 통해 개별 AWS Glue 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 AWS GlueAPI 참조를 참조하세요.

예시

- AWS SDK와 함께 CreateCrawler사용
- AWS SDK 또는 CLI와 함께 CreateJob 사용
- AWS SDK와 함께 DeleteCrawler사용
- AWS SDK와 함께 DeleteDatabase사용
- AWS SDK 또는 CLI와 함께 DeleteJob 사용
- AWS SDK와 함께 DeleteTable사용
- AWS SDK와 함께 GetCrawler사용
- AWS SDK와 함께 GetDatabase사용
- AWS SDK 또는 CLI와 함께 GetDatabases 사용
- AWS SDK 또는 CLI와 함께 GetJob 사용
- AWS SDK 또는 CLI와 함께 GetJobRun 사용
- AWS SDK 또는 CLI와 함께 GetJobRuns 사용
- AWS SDK 또는 CLI와 함께 GetTables 사용
- AWS SDK와 함께 ListJobs사용
- AWS SDK 또는 CLI와 함께 StartCrawler 사용
- AWS SDK 또는 CLI와 함께 StartJobRun 사용

AWS SDK와 함께 **`CreateCrawler`**사용

다음 코드 예시는 `CreateCrawler`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
(Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
```

```
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateCrawler를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);
```

```
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateCrawler를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the
data source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler
will store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
                                         String cron,
                                         String dbName,
                                         String crawlerName) {

       try {
           S3Target s3Target = S3Target.builder()
               .path(s3Path)
               .build();

           List<S3Target> targetList = new ArrayList<>();
           targetList.add(s3Target);
           CrawlerTargets targets = CrawlerTargets.builder()
               .s3Targets(targetList)
               .build();

           CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
               .databaseName(dbName)
               .name(crawlerName)
               .description("Created by the AWS Glue Java API")
               .targets(targets)
               .role(iam)
               .schedule(cron)
               .build();

           glueClient.createCrawler(crawlerRequest);
           System.out.println(crawlerName + " was successfully created");

       } catch (GlueException e) {
```

```
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCrawler를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateCrawler를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
```

```
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateCrawler를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

   public function createCrawler($crawlerName, $role, $databaseName, $path):
Result
   {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
```

```
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateCrawler를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
 data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
 Access
                         Management (IAM) role that grants permission to let AWS
 Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
 crawler.
```

```
        :param db_prefix: The prefix to give any database tables that are created
 by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreateCrawler를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
```

```
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
 the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateCrawler를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreateCrawler을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
    ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
    ///   - databaseName: The name of an AWS Glue database to operate on.
    ///
    /// - Returns: `true` if the crawler is created successfully, otherwise
 `false`.
    func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole:
 String,
                       s3Path: String, cronSchedule: String, databaseName:
 String) async -> Bool {
        let s3Target = GlueClientTypes.S3Target(path: s3url)
        let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

        do {
            _ = try await glueClient.createCrawler(
                input: CreateCrawlerInput(
                    databaseName: databaseName,
                    description: "Created by the AWS SDK for Swift Scenario
 Example for AWS Glue.",
                    name: crawlerName,
                    role: iamRole,
                    schedule: cronSchedule,
                    tablePrefix: tablePrefix,
```

```
                    targets: targetList
                )
            )
        } catch _ as AlreadyExistsException {
            print("*** A crawler named \"\(crawlerName)\" already exists.")
            return false
        } catch _ as OperationTimeoutException {
            print("*** The attempt to create the AWS Glue crawler timed out.")
            return false
        } catch {
            print("*** An unexpected error occurred creating the AWS Glue
crawler: \(error.localizedDescription)")
            return false
        }

        return true
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 CreateCrawler를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateJob** 사용

다음 코드 예시는 CreateJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateJob을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
```

```
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateJob을 참조하십시오.

CLI

AWS CLI

데이터를 변환하는 작업을 생성하려면

다음 `create-job` 예제는 S3에 저장된 스크립트를 실행하는 스트리밍 작업을 생성합니다.

```
aws glue create-job \
    --name my-testing-job \
    --role AWSGlueServiceRoleDefault \
    --command '{ \
        "Name": "gluestreaming", \
        "ScriptLocation": "s3://amzn-s3-demo-bucket/folder/" \
    }' \
    --region us-east-1 \
    --output json \
    --default-arguments '{ \
        "--job-language":"scala", \
        "--class":"GlueApp" \
    }' \
    --profile my-profile \
    --endpoint https://glue.us-east-1.amazonaws.com
```

`test_script.scala`의 콘텐츠:

```
import com.amazonaws.services.glue.ChoiceOption
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.ResolveSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
```

```
import scala.collection.JavaConverters._

object GlueApp {
    def main(sysArgs: Array[String]) {
        val spark: SparkContext = new SparkContext()
        val glueContext: GlueContext = new GlueContext(spark)
        // @params: [JOB_NAME]
        val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME").toArray)
        Job.init(args("JOB_NAME"), glueContext, args.asJava)
        // @type: DataSource
        // @args: [database = "tempdb", table_name = "s3-source",
 transformation_ctx = "datasource0"]
        // @return: datasource0
        // @inputs: []
        val datasource0 = glueContext.getCatalogSource(database = "tempdb",
 tableName = "s3-source", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
        // @type: ApplyMapping
        // @args: [mapping = [("sensorid", "int", "sensorid", "int"),
 ("currenttemperature", "int", "currenttemperature", "int"), ("status", "string",
 "status", "string")], transformation_ctx = "applymapping1"]
        // @return: applymapping1
        // @inputs: [frame = datasource0]
        val applymapping1 = datasource0.applyMapping(mappings = Seq(("sensorid",
 "int", "sensorid", "int"), ("currenttemperature", "int", "currenttemperature",
 "int"), ("status", "string", "status", "string")), caseSensitive = false,
 transformationContext = "applymapping1")
        // @type: SelectFields
        // @args: [paths = ["sensorid", "currenttemperature", "status"],
 transformation_ctx = "selectfields2"]
        // @return: selectfields2
        // @inputs: [frame = applymapping1]
        val selectfields2 = applymapping1.selectFields(paths = Seq("sensorid",
 "currenttemperature", "status"), transformationContext = "selectfields2")
        // @type: ResolveChoice
        // @args: [choice = "MATCH_CATALOG", database = "tempdb", table_name =
 "my-s3-sink", transformation_ctx = "resolvechoice3"]
        // @return: resolvechoice3
        // @inputs: [frame = selectfields2]
        val resolvechoice3 = selectfields2.resolveChoice(choiceOption =
 Some(ChoiceOption("MATCH_CATALOG")), database = Some("tempdb"), tableName =
 Some("my-s3-sink"), transformationContext = "resolvechoice3")
        // @type: DataSink
```

```
        // @args: [database = "tempdb", table_name = "my-s3-sink",
 transformation_ctx = "datasink4"]
        // @return: datasink4
        // @inputs: [frame = resolvechoice3]
        val datasink4 = glueContext.getCatalogSink(database = "tempdb",
 tableName = "my-s3-sink", redshiftTmpDir = "", transformationContext =
 "datasink4").writeDynamicFrame(resolvechoice3)
        Job.commit()
    }
}
```

출력:

```
{
    "Name": "my-testing-job"
}
```

자세한 내용은 AWS 개발자 안내서의 AWS Glue에 작업 작성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateJob을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
```

```
    public static void createJob(GlueClient glueClient, String jobName, String
 iam, String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateJob을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});
```

```
  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateJob을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
```

```
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateJob을 참조하십시오.

PowerShell

Tools for PowerShell V4

예제 1: 이 예제는 AWS Glue에서 새 작업을 생성합니다. 명령 이름 값은 항상 **glueetl**입니다. AWS Glue는 Python 또는 Scala로 작성된 작업 스크립트 실행을 지원합니다. 이 예제에서는 작업 스크립트(MyTestGlueJob.py)가 Python으로 작성되었습니다. Python 파라미터는 **$DefArgs** 변수에 지정된 다음 해시 테이블을 허용하는 **DefaultArguments** 파라미터의 PowerShell 명령에 전달됩니다. **$JobParams** 변수의 파라미터는 AWS Glue API 참조의 작업(https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) 항목에 설명된 CreateJob API에서 가져온 것입니다.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
```

```
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateJob을 참조하세요.

Tools for PowerShell V5

예제 1: 이 예제는 AWS Glue에서 새 작업을 생성합니다. 명령 이름 값은 항상 **glueetl**입니다. AWS Glue는 Python 또는 Scala로 작성된 작업 스크립트 실행을 지원합니다. 이 예제에서는 작업 스크립트(MyTestGlueJob.py)가 Python으로 작성되었습니다. Python 파라미터는 **$DefArgs** 변수에 지정된 다음 해시 테이블을 허용하는 **DefaultArguments** 파라미터의 PowerShell 명령에 전달됩니다. **$JobParams** 변수의 파라미터는 AWS Glue API 참조의 작업(https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) 항목에 설명된 CreateJob API에서 가져온 것입니다.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
```

```
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V5)의 CreateJob을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
```

```
        self.glue_client = glue_client


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreateJob을 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
```

```
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateJob을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreateJob을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
 scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
```

```
            )
        } catch {
            return false
        }
        return true
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 CreateJob을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteCrawler**사용

다음 코드 예시는 DeleteCrawler의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
```

```
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteCrawler를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::DeleteCrawlerRequest request;
       request.SetName(crawler);

       Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully deleted the crawler." << std::endl;
       }
       else {
           std::cerr << "Error deleting the crawler. "
                     << outcome.GetError().GetMessage() << std::endl;
           result = false;
       }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteCrawler를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a specific AWS Glue crawler.
    *
    * @param glueClient  the AWS Glue client object
    * @param crawlerName the name of the crawler to be deleted
    * @throws GlueException if an error occurs during the deletion process
    */
   public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
       try {
           DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                   .name(crawlerName)
                   .build();

           glueClient.deleteCrawler(deleteCrawlerRequest);
           System.out.println(crawlerName + " was deleted");

       } catch (GlueException e) {
           throw e;
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteCrawler를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteCrawler를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
```

```
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteCrawler를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteCrawler를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteCrawler를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteCrawler을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
```

```
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 DeleteCrawler를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteDatabase**사용

다음 코드 예시는 DeleteDatabase의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
```

```
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteDatabase를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
```

```
                    << std::endl;
            result = false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteDatabase를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact
with the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName)
{
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDatabase를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteDatabase를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
```

```
    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteDatabase를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
```

```
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteDatabase를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteDatabase를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteDatabase을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Delete the specified database.
    ///
    /// - Parameters:
```

```
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient:
glueClient, databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables.

            do {
                _ = try await glueClient.batchDeleteTable(
                    input: BatchDeleteTableInput(
                        databaseName: databaseName,
                        tablesToDelete: tableNames
                    )
                )
            } catch {
                print("*** Unable to delete the tables.")
            }
            return true
        }

        // Delete the database itself.

        do {
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
```

```
        } catch {
            print("*** Unable to delete the database.")
        }
        return true
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 DeleteDatabase를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteJob** 사용

다음 코드 예시는 DeleteJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteJob을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteJob을 참조하십시오.

CLI

AWS CLI

작업을 삭제하려면

다음 delete-job 예제에서는 더 이상 필요하지 않은 작업을 삭제합니다.

```
aws glue delete-job \
    --job-name my-testing-job
```

출력:

```
{
    "JobName": "my-testing-job"
}
```

자세한 내용은 AWS Glue 개발자 안내서에서 AWS Glue 콘솔에서 작업 사용을 참조하세요.

- API에 대한 세부 정보는 AWS CLI 명령 참조의 DeleteJob을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Deletes a Glue job.
 *
 * @param glueClient the Glue client to use for the operation
 * @param jobName    the name of the job to be deleted
 * @throws GlueException if there is an error deleting the job
 */
public static void deleteJob(GlueClient glueClient, String jobName) {
```

```
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteJob을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteJob을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteJob을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
```

```
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteJob을 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
```

```
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteJob을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteJob을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 DeleteJob을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **DeleteTable**사용

다음 코드 예시는 DeleteTable의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteTable을 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteTable을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteTable을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteTable를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteTable을 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteTable을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **GetCrawler**사용

다음 코드 예시는 `GetCrawler`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetCrawler를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

       if (outcome.IsSuccess()) {
           Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
           std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                           crawlerState)
                   << "." << std::endl;
       }
       else {
           std::cerr << "Error retrieving a crawler.  "
                     << outcome.GetError().GetMessage() << std::endl;
           deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
           return false;
       }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetCrawler를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves a specific crawler from the AWS Glue service and waits for it to
be in the "READY" state.
    *
    * @param glueClient  the AWS Glue client used to interact with the Glue
service
    * @param crawlerName the name of the crawler to be retrieved
    */
   public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) throws InterruptedException {
       try {
           GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
               .name(crawlerName)
               .build();

           boolean ready = false;
           while (!ready) {
               GetCrawlerResponse response =
glueClient.getCrawler(crawlerRequest);
               String status = response.crawler().stateAsString();
               if (status.compareTo("READY") == 0) {
                   ready = true;
               }
               Thread.sleep(3000);
           }

           System.out.println("The crawler is now ready");

       } catch (GlueException | InterruptedException e) {
           throw e;
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetCrawler를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetCrawler를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
```

```
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetCrawler를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetCrawler를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
                raise
        return crawler
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetCrawler를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
```

```
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetCrawler를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GetCrawler을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
```

```
import AWSGlue

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
 GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
            }
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 GetCrawler를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **GetDatabase**사용

다음 코드 예시는 GetDatabase의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetDatabase를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetDatabaseRequest request;
       request.SetName(CRAWLER_DATABASE_NAME);

       Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

       if (outcome.IsSuccess()) {
           const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

           std::cout << "Successfully retrieve the database\n" <<
                     database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
       }
       else {
           std::cerr << "Error getting the database.  "
                     << outcome.GetError().GetMessage() << std::endl;
           deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
           return false;
       }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetDatabase를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves the specific database from the AWS Glue service.
    *
    * @param glueClient   an instance of the AWS Glue client used to interact
with the service
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from
the AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " +
createDate);

       } catch (GlueException e) {
           throw e;
```

```
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetDatabase를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetDatabase를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetDatabase를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetDatabase를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetDatabase를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetDatabase를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GetDatabase을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue
```

```
    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 GetDatabase를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetDatabases** 사용

다음 코드 예시는 GetDatabases의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

AWS Glue 데이터 카탈로그의 일부 또는 모든 데이터베이스의 정의를 나열하려면

다음 get-databases 예제는 데이터 카탈로그의 데이터베이스에 대한 정보를 반환합니다.

```
aws glue get-databases
```

출력:

```
{
    "DatabaseList": [
        {
            "Name": "default",
            "Description": "Default Hive database",
            "LocationUri": "file:/spark-warehouse",
            "CreateTime": 1602084052.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "flights-db",
            "CreateTime": 1587072847.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
```

```
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "legislators",
            "CreateTime": 1601415625.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "tempdb",
            "CreateTime": 1601498566.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        }
    ]
}
```

자세한 내용은 AWS Glue 개발자 가이드의 데이터 카탈로그에서 데이터베이스 정의를 참조하세요.

- API에 대한 세부 정보는 AWS CLI 명령 참조의 GetDatabases를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const getDatabases = () => {
  const client = new GlueClient({});

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- API에 대한 세부 정보는 AWS SDK for JavaScript API 참조의 GetDatabases를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 `GetJob` 사용

다음 코드 예시는 `GetJob`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

CLI

AWS CLI

작업 정보를 검색하려면

다음 `get-job` 예제는 작업 정보를 검색합니다.

```
aws glue get-job \
    --job-name my-testing-job
```

출력:

```
{
    "Job": {
        "Name": "my-testing-job",
        "Role": "Glue_DefaultRole",
        "CreatedOn": 1602805698.167,
        "LastModifiedOn": 1602805698.167,
        "ExecutionProperty": {
            "MaxConcurrentRuns": 1
        },
        "Command": {
            "Name": "gluestreaming",
            "ScriptLocation": "s3://janetst-bucket-01/Scripts/test_script.scala",
            "PythonVersion": "2"
        },
        "DefaultArguments": {
            "--class": "GlueApp",
            "--job-language": "scala"
        },
        "MaxRetries": 0,
        "AllocatedCapacity": 10,
        "MaxCapacity": 10.0,
        "GlueVersion": "1.0"
    }
}
```

자세한 내용은 AWS Glue 개발자 안내서의 작업을 참조하세요.

- API에 대한 세부 정보는 AWS CLI 명령 참조의 GetJob을 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- API에 대한 세부 정보는 AWS SDK for JavaScript API 참조의 GetJob을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetJobRun** 사용

다음 코드 예시는 GetJobRun의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetJobRun을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
 jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetJobRun을 참조하십시오.

CLI

AWS CLI

작업 실행 정보를 가져오려면

다음 `get-job-run` 예제는 작업 실행 정보를 검색합니다.

```
aws glue get-job-run \
    --job-name "Combine legistators data" \
    --run-id jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e
```

출력:

```
{
```

```
    "JobRun": {
        "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
        "Attempt": 0,
        "JobName": "Combine legistators data",
        "StartedOn": 1602873931.255,
        "LastModifiedOn": 1602874075.985,
        "CompletedOn": 1602874075.985,
        "JobRunState": "SUCCEEDED",
        "Arguments": {
            "--enable-continuous-cloudwatch-log": "true",
            "--enable-metrics": "",
            "--enable-spark-ui": "true",
            "--job-bookmark-option": "job-bookmark-enable",
            "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
        },
        "PredecessorRuns": [],
        "AllocatedCapacity": 10,
        "ExecutionTime": 117,
        "Timeout": 2880,
        "MaxCapacity": 10.0,
        "WorkerType": "G.1X",
        "NumberOfWorkers": 10,
        "LogGroupName": "/aws-glue/jobs",
        "GlueVersion": "2.0"
    }
}
```

자세한 내용은 AWS Glue 개발자 안내서의 작업 실행을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetJobRun을 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetJobRun을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
 ['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
 Result
    {
```

```
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetJobRun을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetJobRun를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
```

```
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetJobRun을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let get_job_run = || async {
            Ok::<JobRun, GlueMvpError>(
                glue.get_job_run()
                    .job_name(self.job())
                    .run_id(job_run_id.to_string())
                    .send()
                    .await
                    .map_err(GlueMvpError::from_glue_sdk)?
                    .job_run()
                    .ok_or_else(|| GlueMvpError::Unknown("Failed to get
job_run".into()))?
                    .to_owned(),
            )
        };

        let mut job_run = get_job_run().await?;
        let mut state =
job_run.job_run_state().unwrap_or(&unknown_state).to_owned();

        while matches!(
            state,
            JobRunState::Starting | JobRunState::Stopping | JobRunState::Running
        ) {
            info!(?state, "Waiting for job to finish");
```

```
            tokio::time::sleep(self.wait_delay).await;

            job_run = get_job_run().await?;
            state = job_run.job_run_state().unwrap_or(&unknown_state).to_owned();
        }
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GetJobRun을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String)
 async -> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
```

```
        } catch {
            return nil
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 GetJobRun을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetJobRuns** 사용

다음 코드 예시는 GetJobRuns의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get information about all AWS Glue runs of a specific job.
/// </summary>
/// <param name="jobName">The name of the job.</param>
/// <returns>A list of JobRun objects.</returns>
public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
{
    var jobRuns = new List<JobRun>();

    var request = new GetJobRunsRequest
```

```
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetJobRuns를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
```

```
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetJobRuns를 참조하십시오.

CLI

AWS CLI

작업에 대한 모든 작업 실행 정보를 가져오려면

다음 `get-job-runs` 예제는 작업에 대한 작업 실행 정보를 검색합니다.

```
aws glue get-job-runs \
    --job-name "my-testing-job"
```

출력:

```
{
    "JobRuns": [
        {
            "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
            "Attempt": 0,
            "JobName": "my-testing-job",
            "StartedOn": 1602873931.255,
            "LastModifiedOn": 1602874075.985,
            "CompletedOn": 1602874075.985,
            "JobRunState": "SUCCEEDED",
            "Arguments": {
                "--enable-continuous-cloudwatch-log": "true",
                "--enable-metrics": "",
                "--enable-spark-ui": "true",
                "--job-bookmark-option": "job-bookmark-enable",
                "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
            },
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 117,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_2",
            "Attempt": 2,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "JobName": "my-testing-job",
            "StartedOn": 1602811168.496,
            "LastModifiedOn": 1602811282.39,
            "CompletedOn": 1602811282.39,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
```

```
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 021AAB703DB20A2D;
                S3 Extended Request ID: teZk24Y09TkXzBvMPG502L5VJBhe9DJuWA9/
TXtuGOqfByajkfL/Tlqt5JBGdEGpigAqzdMDM/U=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 110,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "Attempt": 1,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f",
            "JobName": "my-testing-job",
            "StartedOn": 1602811020.518,
            "LastModifiedOn": 1602811138.364,
            "CompletedOn": 1602811138.364,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 2671D37856AE7ABB;
                S3 Extended Request ID: RLJCJw20brV
+PpC6GpORahyF2fp9flB5SSb2bTGPnUSPVizLXRl1PN3QZldb+v1o9qRVktNYbW8=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 113,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        }
    ]
```

```
}
```

자세한 내용은 AWS Glue 개발자 안내서의 작업 실행을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetJobRuns를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the
job runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
```

```
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetJobRuns를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const getJobRuns = (jobName) => {
```

```
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetJobRuns를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetJobRuns를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetJobRuns를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetJobRuns를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetTables** 사용

다음 코드 예시는 GetTables의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get a list of tables for an AWS Glue database.
/// </summary>
/// <param name="dbName">The name of the database.</param>
/// <returns>A list of Table objects.</returns>
public async Task<List<Table>> GetTablesAsync(string dbName)
{
    var request = new GetTablesRequest { DatabaseName = dbName };
    var tables = new List<Table>();

    // Get a paginator for listing the tables.
    var tablePaginator = _amazonGlue.Paginators.GetTables(request);

    await foreach (var response in tablePaginator.Responses)
    {
        tables.AddRange(response.TableList);
    }

    return tables;
```

```
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetTables를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetTablesRequest request;
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       std::vector<Aws::Glue::Model::Table> all_tables;
       Aws::String nextToken; // Used for pagination.
       do {
           Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

           if (outcome.IsSuccess()) {
               const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
               all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
               nextToken = outcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error getting the tables. "
                         << outcome.GetError().GetMessage()
                         << std::endl;
```

```
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetTables를 참조하십시오.

CLI

AWS CLI

지정된 데이터베이스의 일부 또는 모든 테이블의 정의를 나열하려면

다음 `get-tables` 예제는 지정된 데이터베이스의 테이블에 대한 정보를 반환합니다.

```
aws glue get-tables --database-name 'tempdb'
```

출력:

```
{
```

```
    "TableList": [
        {
            "Name": "my-s3-sink",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730539.0,
            "UpdateTime": 1602730539.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/test-s3-output/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "s3-source",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730658.0,
            "UpdateTime": 1602730658.0,
            "Retention": 0,
```

```
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "test-kinesis-input",
            "DatabaseName": "tempdb",
            "CreateTime": 1601507001.0,
            "UpdateTime": 1601507001.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
```

```
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "my-testing-stream",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "Parameters": {
                    "kinesisUrl": "https://kinesis.us-east-1.amazonaws.com",
                    "streamName": "my-testing-stream",
                    "typeOfData": "kinesis"
                },
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        }
    ]
}
```

자세한 내용은 AWS Glue 개발자 안내서의 AWS Glue 데이터 카탈로그의 테이블 정의를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetTables를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table
names from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
               .build();

           GetTablesResponse response = glueClient.getTables(tableRequest);
           List<Table> tables = response.tableList();
           if (tables.isEmpty()) {
               System.out.println("No tables were returned");
           } else {
               for (Table table : tables) {
                   myTableName = table.name();
                   System.out.println("Table name is: " + myTableName);
               }
           }

       } catch (GlueException e) {
           throw e;
       }
       return myTableName;
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetTables를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetTables를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
```

```
            'DatabaseName' => $databaseName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetTables를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return response["TableList"]
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetTables를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
```

```
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetTables를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GetTables을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
```

```
import AWSGlue

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async
 -> [GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 GetTables를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 함께 **ListJobs**사용

다음 코드 예시는 ListJobs의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListJobs를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::ListJobsRequest listJobsRequest;

       Aws::String nextToken;
       std::vector<Aws::String> allJobNames;

       do {
           if (!nextToken.empty()) {
               listJobsRequest.SetNextToken(nextToken);
           }
           Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                   listJobsRequest);

           if (listRunsOutcome.IsSuccess()) {
               const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
               allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
               nextToken = listRunsOutcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error listing jobs. "
                         << listRunsOutcome.GetError().GetMessage()
                         << std::endl;
```

```
            }
        } while (!nextToken.empty());
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListJobs를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListJobs를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $jobs = $glueService->listJobs();
```

```
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
 Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListJobs를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListJobs를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
```

```
def initialize(glue_client, logger)
  @glue_client = glue_client
  @logger = logger
end

# Retrieves a list of jobs in AWS Glue.
#
# @return [Aws::Glue::Types::ListJobsResponse]
def list_jobs
  @glue_client.list_jobs
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not list jobs: \n#{e.message}")
  raise
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListJobs를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    let mut list_jobs = glue.list_jobs().into_paginator().send();
    while let Some(list_jobs_output) = list_jobs.next().await {
        match list_jobs_output {
            Ok(list_jobs) => {
                let names = list_jobs.job_names();
                info!(?names, "Found these jobs")
            }
            Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
        }
    }
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListJobs을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the AWS Glue jobs listed on the user's account.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - maxJobs: The maximum number of jobs to return (default: 100).
    ///
    /// - Returns: An array of strings listing the names of all available AWS
    ///   Glue jobs.
    func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
        var jobList: [String] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.listJobs(
                    input: ListJobsInput(
                        maxResults: maxJobs,
                        nextToken: nextToken
                    )
                )

                guard let jobs = output.jobNames else {
                    return jobList
                }

                jobList = jobList + jobs
                nextToken = output.nextToken
            } catch {
                return jobList
```

```
            }
        } while (nextToken != nil)

        return jobList
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 ListJobs를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **StartCrawler** 사용

다음 코드 예시는 `StartCrawler`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Start an AWS Glue crawler.
/// </summary>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StartCrawlerAsync(string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
    {
        Name = crawlerName,
```

```
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NETAPI 참조의 SStartCrawler를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::StartCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


       if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                   outcome.GetError().GetErrorType())) {
           if (!outcome.IsSuccess()) {
               std::cout << "Crawler was already started." << std::endl;
           }
           else {
               std::cout << "Successfully started crawler." << std::endl;
```

```
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                      crawlerState)
                              << ". After " << iterations
                              << " seconds elapsed."
                              << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                              << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
```

```
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++API 참조의 SStartCrawler를 참조하십시오.

CLI

AWS CLI

크롤러를 시작하려면

다음 `start-crawler` 예제에서는 크롤러를 시작합니다.

```
aws glue start-crawler --name my-crawler
```

출력:

```
None
```

자세한 내용은 AWS Glue 개발자 안내서의 크롤러 정의 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartCrawler를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.xAPI 참조의 SStartCrawler를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });
```

```
  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScriptAPI 참조의 SStartCrawler를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StartCrawler를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- API 세부 정보는 AWS SDK for PHPAPI 참조의 SStartCrawler를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
```

```
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 StartCrawler를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for RubyAPI 참조의 SStartCrawler를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let start_crawler =
glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
```

```
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

- API 세부 정보는 AWS SDK for Rust API 참조의 StartCrawler를 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise
 `false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 StartCrawler를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **StartJobRun** 사용

다음 코드 예시는 StartJobRun의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Start an AWS Glue job run.
/// </summary>
/// <param name="jobName">The name of the job.</param>
/// <returns>A string representing the job run Id.</returns>
public async Task<string> StartJobRunAsync(
    string jobName,
    string inputDatabase,
    string inputTable,
    string bucketName)
{
    var request = new StartJobRunRequest
    {
        JobName = jobName,
        Arguments = new Dictionary<string, string>
```

```
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 StartJobRun을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);
```

```
        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
```

```
                                  " seconds elapsed." << std::endl;
                          done = true;
                      }
                      else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                          std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                              jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                      }
                  }
                  else {
                      std::cerr << "Error retrieving job run state. "
                                << jobRunOutcome.GetError().GetMessage()
                                << std::endl;
                      deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                   bucketName, clientConfig);
                      return false;
                  }
              }
          }
          else {
              std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                        << std::endl;
              deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                           clientConfig);
              return false;
          }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 StartJobRun을 참조하십시오.

CLI

AWS CLI

작업을 실행하기 시작하려면

다음 `start-job-run` 예제에서는 작업을 시작합니다.

```
aws glue start-job-run \
    --job-name my-job
```

출력:

```
{
    "JobRunId":
 "jr_22208b1f44eb5376a60569d4b21dd20fcb8621e1a366b4e7b2494af764b82ded"
}
```

자세한 내용은 AWS Glue 개발자 안내서의 작업 작성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartJobRun을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Starts a job run in AWS Glue.
    *
    * @param glueClient    the AWS Glue client to use for the job run
    * @param jobName       the name of the Glue job to run
    * @param inputDatabase the name of the input database
    * @param inputTable    the name of the input table
    * @param outBucket     the URL of the output S3 bucket
    * @throws GlueException if there is an error starting the job run
    */
   public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                               String outBucket) {
       try {
           Map<String, String> myMap = new HashMap<>();
           myMap.put("--input_database", inputDatabase);
```

```
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
 response.responseMetadata().requestId());

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartJobRun을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
```

```
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 StartJobRun을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
```

```
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 StartJobRun을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_job_run(self, name, input_database, input_table,
 output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
 tables
                               that describe the source data. This is typically
 created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
```

```
        try:
            # The custom Arguments that are passed to this function are used by
the
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 StartJobRun를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
 job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 StartJobRun을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

- API 세부 정보는 AWS SDK for Rust API 참조의 StartJobRun을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import AWSClientRuntime
import AWSGlue

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to run.
    ///   - databaseName: The name of the AWS Glue database to run the job
 against.
    ///   - tableName: The name of the table in the database to run the job
 against.
    ///   - outputURL: The AWS S3 URI of the bucket location into which to
    ///     write the resulting output.
    ///
    /// - Returns: `true` if the job run is started successfully, otherwise
 `false`.
    func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
 String,
                     tableName: String, outputURL: String) async -> String? {
        do {
            let output = try await glueClient.startJobRun(
                input: StartJobRunInput(
                    arguments: [
                        "--input_database": databaseName,
                        "--input_table": tableName,
                        "--output_bucket_url": outputURL
                    ],
                    jobName: jobName,
                    numberOfWorkers: 10,
                    workerType: .g1x
                )
```

```
            )

            guard let id = output.jobRunId else {
                return nil
            }

            return id
        } catch {
            return nil
        }
    }
```

- API 세부 정보는 AWS SDK for Swift API 참조의 StartJobRun을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS Glue의 보안

AWS는 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객으로서 여러분은 가장 높은 보안 요구 사항을 충족하기 위해 설계된 데이터 센터 및 네트워크 아키텍처의 혜택을 받게 됩니다.

보안은 AWS과 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS클라우드에서 AWS서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사원은 정기적으로 AWS 규제 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. AWS Glue에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내의 AWS서비스를 참조하세요.
- 클라우드 내 보안 – 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS Glue 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 AWS Glue을(를) 구성하는 방법을 보여줍니다. 또한 AWS Glue 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스 사용 방법을 알아봅니다.

주제

- AWS Glue의 데이터 보호
- AWS Glue의 Identity and Access Management
- AWS Lake Formation 액세스 제어 모델
- AWS Glue를 통해 Amazon S3 Access Grants 사용
- AWS Glue의 로깅 및 모니터링
- AWS Glue의 규정 준수 확인
- AWS Glue의 복원성
- AWS Glue에서 인프라 보안

AWS Glue의 데이터 보호

AWS Glue는 데이터를 보호하기 위해 설계된 여러 기능을 제공합니다.

주제

- 저장 데이터 암호화
- 전송 중 데이터 암호화
- FIPS 규정 준수
- 키 관리
- 기타 AWS 서비스에서 AWS Glue 종속성
- 개발 엔드포인트

저장 데이터 암호화

AWS Glue는 AWS Glue Studio를 사용하여 시각적 ETL 작업 구축 및 개발 엔드포인트를 사용하여 스크립트 개발에 대한 미사용 데이터 암호화를 지원합니다. AWS Key Management Service(AWS KMS) 키를 사용하여 암호화된 미사용 데이터를 기록하도록 ETL(추출, 변환 및 로드) 작업 및 개발 엔드포인트를 구성할 수 있습니다. AWS KMS로 관리하는 키를 사용하여 AWS Glue Data Catalog에 저장된 메타데이터도 암호화할 수 있습니다. 또한 AWS KMS 키를 사용하여 크롤러 및 ETL 작업에서 생성된 로그와 작업 북마크를 암호화할 수 있습니다.

작업, 크롤러 및 개발 엔드포인트를 통해 Amazon Simple Storage Service(S3)와 Amazon CloudWatch Logs에 기록된 데이터 외에 AWS Glue Data Catalog의 메타데이터 객체도 암호화할 수 있습니다. AWS Glue에 작업, 크롤러 및 개발 엔드포인트를 만들 때 보안 구성을 연결하여 암호화 설정을 제공할 수 있습니다. 보안 구성에는 AWS KMS(SSE-KMS)에 저장된 고객 마스터 키(CMK)나 Amazon S3 관리형 서버 측 암호화 키(SSE-S3)가 포함됩니다. AWS Glue 콘솔에서 보안 구성을 만들 수 있습니다.

계정에서 전체 데이터 카탈로그에 대한 암호화를 설정할 수도 있습니다. 이렇게 하려면 AWS KMS에 저장된 CMK를 지정합니다.

> **Important**
>
> AWS Glue에서는 대칭 고객 관리형 키만 지원합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 관리형 키(CMK)를 참조하세요.

암호화를 설정한 상태에서 데이터 카탈로그 객체를 추가하거나, 크롤러를 실행하거나, 작업을 실행하거나, 개발 엔드포인트를 시작할 때 SSE-S3 또는 SSE-KMS 키를 사용하여 유휴 데이터를 씁니다. 그리고 신뢰할 수 있는 전송 계층 보안(TLS) 프로토콜을 통해서만 Java 데이터베이스 연결(JDBC) 데이터 스토어에 액세스하도록 AWS Glue를 구성할 수 있습니다.

AWS Glue에서 다음 위치의 암호화 설정을 제어합니다.

- 데이터 카탈로그의 설정입니다.
- 사용자가 생성하는 보안 구성입니다.
- 사용자의 AWS Glue ETL(추출, 변환 및 로드) 작업에 파라미터로 전달되는 서버 측 암호화 설정(SSE-S3 또는 SSE-KMS)입니다.

암호화를 설정하는 방법에 대한 자세한 내용은 AWS Glue에서 암호화 설정 단원을 참조하십시오.

주제

- 데이터 카탈로그 암호화
- 연결 암호 암호화
- AWS Glue에서 작성한 데이터 암호화

데이터 카탈로그 암호화

AWS Glue Data Catalog 암호화는 민감한 데이터에 대한 보안을 강화합니다. AWS Glue는 AWS Key Management Service(AWS KMS)와 통합되어 데이터 카탈로그에 저장된 메타데이터를 암호화합니다. AWS Glue 콘솔 또는 AWS CLI를 사용하여 데이터 카탈로그의 리소스에 대한 암호화 설정을 활성화하거나 비활성화할 수 있습니다.

데이터 카탈로그의 암호화를 활성화하면 새로 만드는 모든 객체가 암호화됩니다. 암호화를 비활성화하면 새로 만든 객체는 암호화되지 않지만 기존의 암호화된 객체는 암호화된 상태로 유지됩니다.

AWS 관리형 암호화 키 또는 고객 관리 암호화 키를 사용하여 전체 데이터 카탈로그를 암호화할 수 있습니다. 키 유형 및 상태에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS Key Management Service 개념을 참조하세요.

> **Note**
>
> 암호화된 Data Catalog를 크롤러와 함께 사용하는 경우 암호화 설정을 유지해야 합니다. 크롤러가 암호화된 Data Catalog를 처리한 후 암호화 설정을 제거하면 오류가 발생합니다. 암호화 설정을 제거해야 하는 경우 기존 크롤러를 수정하는 대신 새 크롤러를 생성하세요.

AWS 관리형 키

AWS 관리형 키는 AWS KMS와 통합된 AWS 서비스가 고객의 계정에서 고객 대신 생성하고 관리하고 사용하는 KMS 키입니다. 계정에서 AWS 관리형 키를 확인하고 키 정책을 확인하며 AWS CloudTrail

로그에서의 사용을 감사할 수 있습니다. 하지만 이러한 키를 관리하거나 이들의 권한을 변경할 수는 없습니다.

저장된 암호화는 메타데이터를 암호화하는 데 사용되는 AWS Glue에 대한 AWS 관리형 키를 관리하기 위해 AWS KMS와 자동으로 통합됩니다. 메타데이터 암호화를 활성화할 때 AWS 관리형 키가 없는 경우 AWS KMS가 자동으로 새 키를 생성합니다.

자세한 내용은 AWS 관리형 키를 참조하세요.

고객 관리형 키

고객 관리형 키는 사용자가 생성, 소유 및 관리하는 AWS 계정의 KMS 키입니다. 이러한 KMS 키를 완전히 제어할 수 있습니다. 다음을 할 수 있습니다.

- 키 정책, IAM 정책, 권한 부여 설정 및 관리
- 활성화 및 비활성화
- 암호화 자료 교체
- 태그 추가
- 이를 참조하는 별칭 생성
- 이를 삭제하도록 예약

고객 관리형 키의 권한 관리에 대한 자세한 내용은 고객 관리형 키를 참조하세요.

> **Important**
>
> AWS Glue에서는 대칭 고객 관리형 키만 지원합니다. KMS 키 목록에는 대칭 키만 표시됩니다. 그러나 Choose a KMS key ARN(KMS 키 ARN 선택)을 선택하면 콘솔에서 모든 키 유형에 대해 ARN을 입력할 수 있습니다. 대칭 키에 대한 ARN만 입력해야 합니다.
> 대칭 고객 관리형 키를 생성하려면 AWS Key Management Service 개발자 안내서의 대칭 고객 관리형 키 생성 단계를 따르세요.

저장된 데이터 카탈로그 암호화를 활성화하면 다음 리소스 유형이 KMS 키를 사용하여 암호화됩니다.

- 데이터베이스 수
- 표
- 파티션
- 테이블 버전

- 열 통계값
- 사용자 정의 함수
- 데이터 카탈로그 보기

AWS Glue 암호화 컨텍스트

암호화 컨텍스트는 데이터에 대한 추가 컨텍스트 정보를 포함하는 키-값 페어의 선택적 집합입니다. AWS KMS는 암호화 컨텍스트를 인증된 암호화를 지원하기 위한 추가 인증 데이터로 사용합니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하는 경우, AWS KMS은(는) 암호화된 데이터에 암호화 컨텍스트를 바인딩합니다. 데이터의 암호를 해독하려면 요청에 동일한 암호화 컨텍스트를 포함합니다. AWS Glue는 키가 `glue_catalog_id`이고 값이 `catalogId`인 모든 AWS KMS 암호화 연산에서 동일한 암호화 컨텍스트를 사용합니다.

```
"encryptionContext": {
    "glue_catalog_id": "111122223333"
}
```

AWS 관리형 키 또는 대칭 고객 관리형 키를 사용하여 데이터 카탈로그를 암호화하는 경우 감사 기록 및 로그의 암호화 컨텍스트를 사용하여 키가 어떻게 사용되고 있는지 파악할 수도 있습니다. 암호화 컨텍스트는 AWS CloudTrail 또는 Amazon CloudWatch 로그에서 생성되는 로그에도 나타납니다.

암호화 활성화

AWS Glue 콘솔의 데이터 카탈로그 설정에서 또는 AWS CLI를 사용하여 AWS Glue Data Catalog 객체에 대한 암호화를 활성화할 수 있습니다.

Console

콘솔을 사용하여 암호화 활성화

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창의 데이터 카탈로그를 선택합니다.
3. 데이터 카탈로그 설정 페이지에서 메타데이터 암호화 확인란을 선택하고 AWS KMS 키를 선택합니다.

   암호화를 사용 설정할 때 고객 관리 키를 지정하지 않으면 암호화 설정에서 AWS 관리형 KMS 키를 사용합니다.

4. (선택 사항) 고객 관리형 키를 사용하여 데이터 카탈로그를 암호화하는 경우, 데이터 카탈로그는 리소스를 암호화 및 해독하기 위해 IAM 역할을 등록할 수 있는 옵션을 제공합니다. AWS Glue가 사용자를 대신하여 수행할 수 있는 IAM 역할 권한을 부여해야 합니다. 여기에는 데이터 암호화 및 암호 해독을 위한 AWS KMS 권한이 포함됩니다.

   데이터 카탈로그에서 새 리소스를 생성하면 AWS Glue가 데이터 암호화를 위해 제공된 IAM 역할을 맡습니다. 마찬가지로 소비자가 리소스에 액세스할 때 AWS Glue가 데이터 암호 해독을 위해 IAM 역할을 맡습니다. 필요한 권한이 있는 IAM 역할을 등록하면 직접 호출 주체는 더 이상 키에 액세스하고 데이터를 해독하는 데 권한이 필요하지 않습니다.

   > **Important**
   >
   > 고객 관리형 키를 사용하여 데이터 카탈로그 리소스를 암호화하는 경우에만 KMS 작업을 IAM 역할에 위임할 수 있습니다. 현재 KMS 역할 위임 기능은 데이터 카탈로그 리소스를 암호화하는 데 AWS 관리형 키를 사용하는 것을 지원하지 않습니다.

   > **Warning**
   >
   > IAM 역할이 KMS 작업을 위임하도록 설정하면 이전에 AWS 관리형 키를 사용하여 암호화되었던 데이터 카탈로그 리소스에 더 이상 액세스할 수 없습니다.

   a. AWS Glue가 사용자를 대신하여 데이터를 암호화하고 암호를 해독할 수 있는 IAM 역할을 활성화하려면 KMS 작업을 IAM 역할에 위임 옵션을 선택합니다.

   b. 다음으로 IAM 역할을 선택합니다.

      IAM 역할을 생성하려면 AWS Glue의 IAM 역할 생성을 참조하십시오.

      AWS Glue가 데이터 카탈로그에 액세스하기 위해 가정하는 IAM 역할에는 데이터 카탈로그의 메타데이터를 암호화 및 암호 해독을 위한 권한이 있어야 합니다. IAM 역할을 생성하여 다음 인라인 정책을 연결할 수 있습니다.

      - 다음 정책을 추가하여 데이터 카탈로그의 암호화 및 암호 해독을 위한 AWS KMS 권한을 포함하세요.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:<region>:<account-id>:key/<key-id>"
        }
    ]
}
```

- 다음으로 AWS Glue 서비스에 대한 역할에 다음 신뢰 정책을 추가하여 IAM 역할을 위임합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
 }
```

- 그런 다음 IAM 역할에 `iam:PassRole` 권한을 추가합니다.

```
                    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
```

```
                "arn:aws:iam::<account-id>:role/<encryption-role-name>"
            ]
        }
    ]
}
```

암호화를 사용 설정할 때 AWS Glue가 맡을 IAM 역할을 지정하지 않은 경우 데이터 카탈로그에 액세스하는 주체는 다음 API 작업을 수행할 수 있는 권한이 있어야 합니다.

- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKey`

AWS CLI

SDK 또는 AWS CLI을 사용하여 암호화 활성화

- `PutDataCatalogEncryptionSettings` API 작업을 사용합니다. 키를 지정하지 않으면 AWS Glue는 고객 계정에 대해 AWS 관리형 암호화 키를 사용하여 데이터 카탈로그를 암호화합니다.

```
aws glue put-data-catalog-encryption-settings \
  --data-catalog-encryption-settings '{
      "EncryptionAtRest": {
      "CatalogEncryptionMode": "SSE-KMS-WITH-SERVICE-ROLE",
      "SseAwsKmsKeyId": "arn:aws:kms:<region>:<account-id>:key/<key-id>",
      "CatalogEncryptionServiceRole":"arn:aws:iam::<account-
id>:role/<encryption-role-name>"
    }

  }'
```

암호화를 활성화하면 데이터 카탈로그 객체에서 생성하는 모든 객체가 암호화됩니다. 이 설정을 삭제하면 데이터 카탈로그에서 생성하는 객체가 더 이상 암호화되지 않습니다. 필요한 KMS 권한을 사용하여 데이터 카탈로그의 기존 암호화된 객체에 계속 액세스할 수 있습니다.

> **Important**
>
> AWS KMS 키는 데이터 카탈로그에 함께 암호화된 객체의 AWS KMS 키 스토어에서 계속 사용할 수 있어야 합니다. 키를 제거하면 객체의 암호를 해독할 수 없습니다. 일부 시나리오에서는 데이터 카탈로그 메타데이터에 대한 액세스를 방지하기 위에 이것이 필요할 수 있습니다.

AWS Glue에 대한 KMS 키 모니터링

데이터 카탈로그 리소스와 함께 KMS 키를 사용하는 경우 AWS CloudTrail 또는 Amazon CloudWatch 로그를 사용하여 AWS Glue가 AWS KMS로 보내는 요청을 추적할 수 있습니다. AWS CloudTrail은 AWS Glue가 KMS 키로 암호화된 데이터에 액세스하기 위해 직접적으로 호출하는 KMS 작업을 모니터링하고 기록합니다.

다음 예제는 `Decrypt` 및 `GenerateDataKey` 작업에 대한 AWS CloudTrail 이벤트입니다.

Decrypt

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAXPHTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAXPHTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-01-10T14:33:56Z",
                "mfaAuthenticated": "false"
            }
        },
```

```
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-01-10T15:18:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "eu-west-2",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "glue_catalog_id": "111122223333"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "43b019aa-34b8-4798-9b98-ee968b2d63df",
    "eventID": "d7614763-d3fe-4f84-a1e1-3ca4d2a5bbd5",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:<region>:111122223333:key/<key-id>"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
}
```

GenerateDataKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId":
 "AROAXPHTESTANDEXAMPLE:V_00_GLUE_KMS_GENERATE_DATA_KEY_111122223333",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/
V_00_GLUE_KMS_GENERATE_DATA_KEY_111122223333",
```

```
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAXPHTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-01-05T21:15:47Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-01-05T21:15:47Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "eu-west-2",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:eu-west-2:AKIAIOSFODNN7EXAMPLE:key/
AKIAIOSFODNN7EXAMPLE",
        "encryptionContext": {
            "glue_catalog_id": "111122223333"
        },
        "keySpec": "AES_256"
    },
    "responseElements": null,
    "requestID": "64d1783a-4b62-44ba-b0ab-388b50188070",
    "eventID": "1c73689b-2ef2-443b-aed7-8c126585ca5e",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-west-2:111122223333:key/AKIAIOSFODNN7EXAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

연결 암호 암호화

`GetConnection` 및 `GetConnections` API 작업을 사용하여 AWS Glue Data Catalog에서 연결 암호를 검색할 수 있습니다. 이러한 암호는 데이터 카탈로그 연결에 저장되고 AWS Glue가 Java Database Connectivity(JDBC) 데이터 스토어에 연결할 때 사용됩니다. 연결이 생성되거나 업데이트되면 데이터 카탈로그 설정의 옵션이 암호가 암호화되었는지 여부를 확인하고, 암호화된 경우 지정된 AWS Key Management Service(AWS KMS) 키를 확인합니다.

AWS Glue 콘솔의 [데이터 카탈로그 설정(Data catalog settings)] 페이지에서 이 옵션을 설정할 수 있습니다.

연결 암호를 암호화하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. 데이터 카탈로그 설정 페이지에서 Encrypt connection passwords(연결 암호 암호화)를 선택하고 AWS KMS 키를 선택합니다.

> **Important**
>
> AWS Glue에서는 대칭 고객 마스터 키(CMK)만 지원합니다. AWS KMS 키(key) 목록에는 대칭 키만 표시됩니다. 그러나 Choose a AWS KMS key ARN(KMS 키 ARN 선택)을 선택하면 콘솔에서 모든 키 유형의 ARN을 입력할 수 있습니다. 대칭 키에 대한 ARN만 입력해야 합니다.

자세한 내용은 데이터 카탈로그 설정 단원을 참조하십시오.

AWS Glue에서 작성한 데이터 암호화

보안 구성은 AWS Glue에서 사용할 수 있는 보안 속성의 집합입니다. 보안 구성을 사용하여 미사용 데이터를 암호화할 수 있습니다. 다음 시나리오는 보안 구성을 사용할 수 있는 몇 가지 방법을 보여줍니다.

- 암호화된 Amazon CloudWatch Logs를 기록할 AWS Glue 크롤러에 보안 구성을 연결합니다. 크롤러에 보안 구성을 첨부하는 방법에 대한 자세한 내용은 the section called “보안 설정을 구성합니다” 섹션을 참조하세요.
- 암호화된 Amazon Simple Storage Service(Amazon S3) 대상과 암호화된 CloudWatch Logs를 기록할 추출, 변환 및 로드(ETL) 작업에 보안 구성을 연결합니다.
- 암호화된 Amazon S3 데이터로 작업 북마크를 기록할 ETL 작업에 보안 구성을 연결합니다.
- 암호화된 Amazon S3 대상을 기록할 개발 엔드포인트에 보안 구성을 연결합니다.

> **Important**
>
> 현재 보안 구성은 ETL 작업 파라미터로 전달되는 서버 측 암호화(SSE-S3)를 재정의합니다. 따라서 보안 구성과 SSE-S3 파라미터가 모두 작업과 연결된 경우 SSE-S3 파라미터는 무시됩니다.

보안 구성에 대한 자세한 내용은 AWS Glue 콘솔에서 보안 구성 관리 단원을 참조하세요.

주제

- 보안 구성을 사용하도록 AWS Glue 설정
- VPC 작업 및 크롤러용 AWS KMS로 가는 경로 생성
- AWS Glue 콘솔에서 보안 구성 관리

보안 구성을 사용하도록 AWS Glue 설정

다음 단계에 따라 보안 구성을 사용하도록 AWS Glue 환경을 설정합니다.

1. AWS Key Management Service(AWS KMS) 키를 생성하거나 업데이트하여 CloudWatch Logs를 암호화할 AWS Glue 크롤러 및 작업에 전달되는 IAM 역할에 AWS KMS 권한을 부여합니다. 자세한 내용은 Amazon CloudWatch Logs User Guide의 Encrypt Log Data in CloudWatch Logs Using AWS KMS를 참조하세요.

   다음 예에서 *"role1"*, *"role2"*, *"role3"*은 크롤러와 작업에 전달되는 IAM 역할입니다.

   ```
   {
          "Effect": "Allow",
          "Principal": { "Service": "logs.region.amazonaws.com",
          "AWS": [
                   "role1",
                   "role2",
                   "role3"
                ] },
                      "Action": [
                             "kms:Encrypt*",
                             "kms:Decrypt*",
                             "kms:ReEncrypt*",
                             "kms:GenerateDataKey*",
                             "kms:Describe*"
                      ],
                      "Resource": "*"
   }
   ```

   키를 사용하여 CloudWatch Logs를 암호화하는 경우 `"Service": "logs.region.amazonaws.com"`과 같이 `Service` 문이 필요합니다.

2. 사용 전에 AWS KMS 키가 `ENABLED` 상태인지 확인합니다.

> **Note**
>
> Iceberg를 데이터 레이크 프레임워크로 사용하는 경우 Iceberg 테이블에는 서버 측 암호화를 활성화하는 자체 메커니즘이 있습니다. AWS Glue의 보안 구성과 함께 이러한 구성을 활성화해야 합니다. Iceberg 테이블에서 서버 측 암호화를 활성화하려면 Iceberg 설명서의 지침을 검토하세요.

VPC 작업 및 크롤러용 AWS KMS로 가는 경로 생성

인터넷을 통해 연결하지 않고 Virtual Private Cloud(VPC)의 프라이빗 엔드포인트를 통해 AWS KMS에 직접 연결할 수 있습니다. VPC 엔드포인트를 사용하는 경우 VPC와 AWS KMS 사이의 통신은 모두 AWS 네트워크에서 수행됩니다.

VPC 안에 AWS KMS VPC 엔드포인트를 생성할 수 있습니다. 이 단계를 거치지 않으면 작업의 `kms timeout` 또는 크롤러의 `internal service exception`에서 작업이나 크롤러가 실패할 수 있습니다. 자세한 지침은 AWS Key Management Service Developer Guide의 Connecting to AWS KMS Through a VPC Endpoint를 참조하세요.

이러한 설명에 따라 VPC 콘솔에서 다음을 수행해야 합니다.

- 프라이빗 DNS 이름 활성화를 선택합니다.
- Java Database Connectivity(JDBC)에 액세스하는 작업 또는 크롤러에 사용할 보안 그룹(자체 참조 규칙 포함)을 선택합니다. AWS Glue 연결에 대한 자세한 정보는 데이터에 연결 섹션을 참조하세요.

JDBC 데이터 스토어에 액세스하는 크롤러 또는 작업에 보안 구성을 추가할 때 AWS Glue에 AWS KMS 엔드포인트로 가는 경로가 있어야 합니다. 네트워크 주소 변환(NAT) 게이트웨이 또는 AWS KMS VPC 엔드포인트에 경로를 제공할 수 있습니다. NAT 게이트웨이를 생성하려면 Amazon VPC 사용 설명서의 NAT 게이트웨이를 참조하세요.

AWS Glue 콘솔에서 보안 구성 관리

> **Warning**
>
> AWS Glue 보안 구성은 현재 Ray 작업에서 지원되지 않습니다.

AWS Glue의 보안 구성에는 암호화된 데이터를 쓸 때 필요한 속성이 포함되어 있습니다. AWS Glue 콘솔에서 보안 구성을 생성하여 크롤러, 작업 및 개발 엔드포인트에서 사용되는 암호화 속성을 제공합니다.

생성한 보안 구성 목록을 모두 보려면 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 열고 탐색 창에서 [보안 구성(Security configurations)]을 선택합니다.

보안 구성 목록은 각 구성에 대한 다음 속성을 표시합니다.

명칭

구성 생성 시 제공한 고유 이름입니다. 이름은 문자(A~Z), 숫자(0~9), 하이픈(-) 또는 밑줄(_) 포함할 수 있으며 최대 255자로 지정할 수 있습니다.

Amazon S3 암호화 활성화

설정된 경우 SSE-KMS 또는 SSE-S3와 같은 Amazon Simple Storage Service(S3) 암호화 모드가 데이터 카탈로그의 메타데이터 스토어에 대해 활성화합니다.

Amazon CloudWatch Logs 암호화 활성화

설정된 경우 로그를 Amazon CloudWatch에 작성할 때 SSE-KMS와 같은 Amazon S3 암호화 모드가 활성화됩니다.

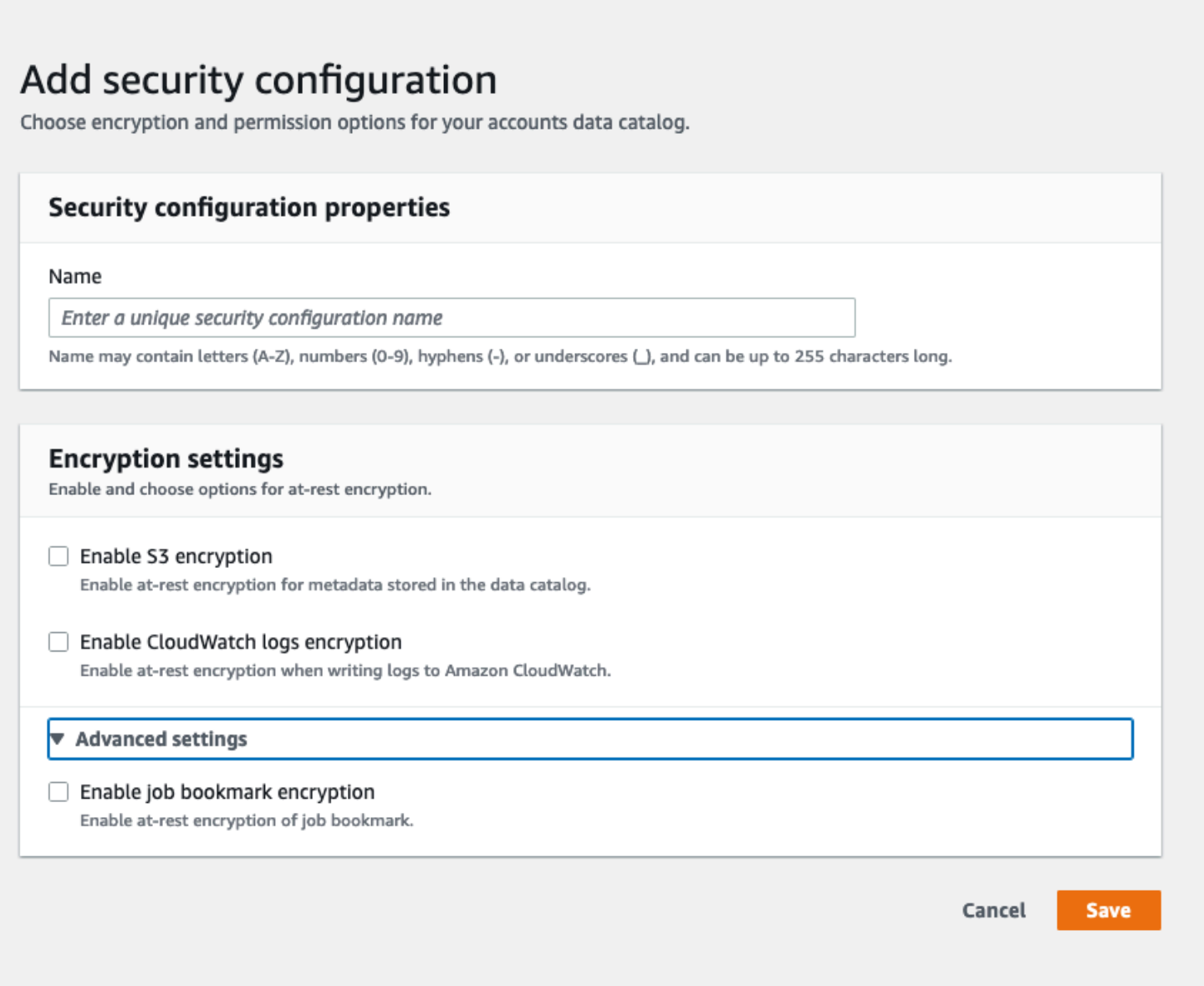
고급 설정: 작업 북마크 암호화 활성화

설정된 경우 작업이 북마크될 때 CSE-KMS와 같은 Amazon S3 암호화 모드가 활성화됩니다.

콘솔의 보안 구성 섹션에서 구성을 추가하거나 삭제할 수 있습니다. 목록에서 구성 이름을 선택하여 구성에 대한 더 자세한 정보를 알아봅니다. 세부 정보는 구성을 생성할 시 정의한 정보를 포함합니다.

보안 구성 추가

AWS Glue 콘솔을 사용하여 보안 구성을 추가하려면 보안 구성 페이지에서 Add security configuration(보안 구성 추가)을 선택합니다.



보안 구성 속성

고유한 보안 구성 이름을 입력합니다. 이름은 문자(A~Z), 숫자(0~9), 하이픈(-) 또는 밑줄(_) 포함할 수 있으며 최대 255자로 지정할 수 있습니다.

암호화 설정

Amazon S3의 데이터 카탈로그에 저장된 메타데이터와 Amazon CloudWatch의 로그에 저장된 메타데이터에 대해 저장 중 암호화를 활성화할 수 있습니다. AWS Glue 콘솔에서 AWS Key Management Service(AWS KMS) 키를 통해 데이터 및 메타데이터의 암호화를 설정하려면 콘솔 사용자에게 정책을 추가합니다. 다음 예와 같이 이 정책은 허용된 리소스를 Amazon S3 데이터 스토어 암호화에 사용되는 주요 Amazon 리소스 이름(ARN)으로 지정해야 합니다.

```
{
"Version": "2012-10-17",
  "Statement": {
  "Effect": "Allow",
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt",
    "kms:Encrypt"],
  "Resource": "arn:aws:kms:region:account-id:key/key-id"}
}
```

> **Important**
>
> 보안 구성이 크롤러나 작업에 연결되면 전달되는 IAM 역할에 AWS KMS 권한이 있어야 합니다. 자세한 내용은 AWS Glue에서 작성한 데이터 암호화 섹션을 참조하세요.

구성을 정의할 때 필요한 다음 속성에 대한 값을 제공할 수 있습니다.

S3 암호화 활성화

Amazon S3 데이터를 작성할 때 Amazon S3 관리형 키(SSE-S3)를 사용하는 서버 측 암호화 또는 AWS KMS 관리형 키(SSE-KMS)를 사용하는 서버 측 암호화를 사용합니다. 이 필드는 선택 사항입니다. Amazon S3 액세스를 허용하려면 AWS KMS 키를 선택하거나 [키 ARN 입력(Enter a key ARN)]을 선택하고 키의 ARN을 제공합니다. `arn:aws:kms:region:account-id:key/key-id` 형식에 ARN 이름을 입력합니다. 또한 `arn:aws:kms:region:account-id:alias/alias-name` 같은 키 별칭으로 ARN을 제공할 수도 있습니다.

작업에 대해 Spark UI를 활성화하면 Amazon S3에 업로드된 Spark UI 로그 파일이 동일한 암호화로 적용됩니다.

> **Important**
>
> AWS Glue에서는 대칭 고객 마스터 키(CMK)만 지원합니다. AWS KMS 키(key) 목록에는 대칭 키만 표시됩니다. 그러나 Choose a AWS KMS key ARN(KMS 키 ARN 선택)을 선택하면 콘솔에서 모든 키 유형의 ARN을 입력할 수 있습니다. 대칭 키에 대한 ARN만 입력해야 합니다.

CloudWatch Logs 암호화 활성화

서버 측(SSE-KMS) 암호화는 CloudWatch Logs 암호화에 사용됩니다. 이 필드는 선택 사항입니다. 이를 설정하려면 AWS KMS 키를 선택하거나 [키 ARN 입력(Enter a key ARN)]을 선택하고 키에 대한 ARN 이름을 제공합니다. `arn:aws:kms:region:account-id:key/key-id` 형식에 ARN 이름을 입력합니다. 또한 `arn:aws:kms:region:account-id:alias/alias-name` 같은 키 별칭으로 ARN을 제공할 수도 있습니다.

고급 설정: 작업 북마크 암호화

클라이언트 측(CSE-KMS) 암호화는 작업 북마크를 암호화하는 데 사용됩니다. 이 필드는 선택 사항입니다. 북마크 데이터는 저장을 위해 Amazon S3에 전송되기 전에 암호화됩니다. 이를 설정하려면 AWS KMS 키를 선택하거나 [키 ARN 입력(Enter a key ARN)]을 선택하고 키에 대한 ARN 이름을 제공합니다. `arn:aws:kms:region:account-id:key/key-id` 형식에 ARN 이름을 입력합니다. 또한 `arn:aws:kms:region:account-id:alias/alias-name` 같은 키 별칭으로 ARN을 제공할 수도 있습니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 다음 주제를 참조하세요.

- `SSE-S3`에 대한 자세한 내용은 Amazon S3가 관리하는 암호화 키(SSE-S3)를 사용하는 서버 측 암호화로 데이터 보호를 참조하십시오.
- `SSE-KMS`에 대한 자세한 내용은 AWS KMS keys를 사용하는 서버 측 암호화로 데이터 보호를 참조하세요.
- `CSE-KMS`에 대한 자세한 내용은 AWS KMS에 저장된 KMS 키 사용을 참조하세요.

전송 중 데이터 암호화

AWS는 이동 중인 데이터에 대해 전송 계층 보안(TLS) 암호화를 제공합니다. AWS Glue의 보안 구성을 사용하여 크롤러, ETL 작업 및 개발 엔드포인트에 대한 암호화 설정을 구성할 수 있습니다. 데이터 카탈로그에 대한 설정을 통해 AWS Glue Data Catalog 암호화를 설정할 수 있습니다.

2018년 9월 4일부터 AWS Glue ETL 및 AWS Glue Data Catalog에 대한 AWS KMS(기존 보유 키 사용 및 서버 측 암호화)가 지원됩니다.

FIPS 규정 준수

명령행 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하십시오.

키 관리

AWS Identity and Access Management(IAM)를 AWS Glue와 함께 사용하여 사용자, AWS 리소스, 그룹, 역할 및 액세스, 거부 등에 대한 세분화된 정책을 정의할 수 있습니다.

조직의 필요에 따라 리소스 기반 및 자격 증명 기반 정책을 모두 사용하여 메타데이터에 대한 액세스를 정의할 수 있습니다. 리소스 기반 정책은 교차 계정 액세스와 같은 정책을 설정할 수 있도록 리소스에 대한 액세스가 허용되거나 거부되는 원칙을 나열합니다. 자격 증명 기반 정책은 특히 IAM 내의 사용자, 그룹 및 역할에 연결됩니다.

단계별 예제는 AWS 빅 데이터 블로그의 리소스 레벨 IAM 권한 및 리소스 기반 정책으로 AWS Glue Data Catalog에 대한 액세스 제한을 참조하세요.

정책의 세분화된 액세스 부분은 `Resource` 절 내에서 정의됩니다. 이 부분은 작업을 수행할 수 있는 AWS Glue Data Catalog 객체 및 해당 작업에서 반환되는 결과 객체를 모두 정의합니다.

개발 엔드포인트는 AWS Glue 스크립트를 개발하고 테스트할 수 있는 환경입니다. 개발 엔드 포인트의 SSH 키를 추가하거나 삭제, 회전할 수 있습니다.

2018년 9월 4일부터 AWS KMS ETL 및 AWS Glue에 대한 AWS Glue Data Catalog(기존 보유 키 사용 및 서버 측 암호화)가 지원됩니다.

기타 AWS 서비스에서 AWS Glue 종속성

사용자가 AWS Glue 콘솔로 작업하려면 해당 사용자는 AWS 계정에 대한 AWS Glue 리소스로 작업하도록 허용하는 최소 권한 집합이 있어야 합니다. 이 AWS Glue 권한 이외에 콘솔은 다음 서비스로부터 권한을 필요로 합니다.

- 로그를 표시할 수 있는 Amazon CloudWatch Logs 권한.
- 역할을 나열하고 전달할 수 있는 AWS Identity and Access Management(IAM) 권한.
- 스택으로 작업할 수 있는 AWS CloudFormation 권한.
- Virtual Private Cloud(VPC), 서브넷, 보안 그룹, 인스턴스 및 기타 객체를 나열할 수 있는(작업, 크롤러를 생성하고 개발 엔드포인트를 생성할 때 VPC와 같은 Amazon EC2 항목을 설정할 수 있는) Amazon Elastic Compute Cloud(Amazon EC2) 권한.

- 버킷과 객체를 나열하고 스크립트를 검색하고 저장할 수 있는 Amazon Simple Storage Service(Amazon S3) 권한.
- 클러스터 작업을 위한 Amazon Redshift 권한.
- 인스턴스를 나열할 수 있는 Amazon Relational Database Service(Amazon RDS) 권한.

개발 엔드포인트

개발 엔드포인트는 AWS Glue 스크립트를 개발하고 테스트할 수 있는 환경입니다. AWS Glue를 사용하여 개발 엔드포인트를 생성, 편집 및 삭제할 수 있습니다. 생성된 모든 개발 엔드포인트를 나열할 수 있습니다. 개발 엔드 포인트의 SSH 키를 추가하거나 삭제, 회전할 수 있습니다. 개발 엔드포인트를 사용하는 노트북을 만들 수도 있습니다.

구성 값을 제공하여 개발 환경을 제공합니다. 이러한 값은 네트워크를 설정하는 방법을 AWS Glue에 알립니다. 따라서 안전하게 개발 엔드포인트에 액세스할 수 있으며 엔드포인트가 데이터 스토어에 액세스할 수 있습니다. 그런 다음, 개발 엔드포인트에 연결하는 노트북을 생성할 수 있습니다. 노트북을 사용하여 ETL 스크립트를 작성하고 테스트할 수 있습니다.

AWS Glue ETL 작업을 실행하는 데 사용하는 IAM 역할과 비슷한 권한을 통해 AWS Identity and Access Management(IAM) 역할을 사용합니다. Virtual Private Cloud(VPC), 서브넷 및 보안 그룹을 사용하여 데이터 리소스에 안전하게 연결할 수 있는 개발 엔드포인트를 생성합니다. SSH를 사용하여 개발 엔드포인트에 연결할 SSH 키 페어를 생성합니다.

JDBC를 사용하여 데이터 집합에 액세스하는 데 사용할 수 있는 VPC 내에서 Amazon S3 데이터에 대한 개발 엔드포인트를 생성할 수 있습니다.

로컬 시스템에 Jupyter Notebook 클라이언트를 설치하고 노트북을 사용하여 개발 엔드포인트에서 ETL 스크립트를 디버깅하고 테스트할 수 있습니다. 아니면 Sagemaker 노트북을 사용하여 AWS에 대한 JupyterLab에서 ETL 스크립트를 작성할 수 있습니다. 개발 엔드포인트와 SageMaker 노트북 함께 사용하기를 참조하세요.

AWS Glue는 `aws-glue-dev-endpoint`를 통해 접두사가 지정되는 이름으로 Amazon EC2 인스턴스를 태깅합니다.

개발 엔드포인트에서 노트북 서버를 설정하여 AWS Glue 확장 프로그램을 통해 PySpark를 실행할 수 있습니다.

AWS Glue의 Identity and Access Management

AWS Identity and Access Management(IAM)은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 AWS Glue 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

> **Note**
>
> AWS Glue 메서드 또는 AWS Lake Formation 권한 부여를 사용하여 AWS Glue 데이터 카탈로그 내 데이터에 대한 액세스 권한을 부여할 수 있습니다. AWS Identity and Access Management(IAM) 정책을 사용하여 AWS Glue 메서드로 세분화된 액세스 제어를 수행합니다. Lake Formation은 관계형 데이터베이스 시스템의 `GRANT/REVOKE` 명령과 유사한 더 간단한 `GRANT/REVOKE` 권한 모델을 사용합니다.
> 이 섹션에는 AWS Glue 메서드 사용 방법에 대한 정보를 포함합니다. Lake Formation 권한 부여 사용에 대한 자세한 내용은 AWS Lake Formation Developer Guide의 Granting Lake Formation permissions를 참조하세요.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- AWS Glue의 작동 방식 IAM
- AWS Glue에 대한 IAM 권한 구성
- AWS Glue 액세스 제어 정책 예제
- AWS Glue에 대한 AWS 관리형 정책 부여
- AWS Glue 리소스 ARN 지정
- 교차 계정 액세스 권한 부여
- AWS Glue 자격 증명 및 액세스 문제 해결

대상

AWS Identity and Access Management(IAM)를 사용하는 방법은 AWS Glue에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 - AWS Glue 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 AWS Glue 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. AWS Glue의 기능에 액세스할 수 없는 경우 AWS Glue 자격 증명 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 - 회사에서 AWS Glue 리소스를 책임지고 있는 경우 AWS Glue에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 AWS Glue 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해합니다. 회사가 AWS Glue에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 AWS Glue의 작동 방식 IAM 섹션을 참조하세요.

IAM 관리자 - IAM 관리자라면 AWS Glue에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 AWS Glue 자격 증명 기반 정책 예제를 보려면 AWS Glue 자격 증명 기반 정책 예제 섹션을 참조하세요.

ID를 통한 인증

인증은 ID 자격 증명을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자이나 IAM 사용자로, 또는 IAM 역할을 수임하여 인증(AWS에 로그인)받아야 합니다.

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예시입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 연동을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법을 참조하세요.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우, 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 API 요청용 AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS은(는) 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 IAM의 AWS Multi-Factor Authentication을 참조하세요.

AWS 계정 루트 사용자

AWS 계정(을)를 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 ID는 AWS 계정루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

관리자 액세스가 필요한 사용자 등 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스에 액세스하도록 요구하는 것이 가장 좋습니다.

페더레이션 ID는 엔터프라이즈 사용자 디렉터리, 웹 ID 공급자, AWS Directory Service, Identity Center 디렉터리의 사용자 또는 보안 인증 정보 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션 ID는 AWS 계정에 액세스할 때 역할을 수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center(을)를 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 AWS 계정 및 애플리케이션에서 사용하기 위해 고유한 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 ID입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 자격 증명이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있

지만, 역할은 임시 자격 증명만 제공합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. AWS Management Console에서 일시적으로 IAM 역할을 수임하려면 사용자에서 IAM 역할(콘솔)로 전환하면 됩니다. AWS CLI 또는 AWS API 작업을 직접적으로 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 자격 증명이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 ID 공급자의 역할 만들기(페더레이션)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 하지만 일부 AWS 서비스을(를) 사용하면 리소스에 정책을 직접 연결할 수 있습니다(역할을 프록시로서 사용하는 대신) . 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부 AWS 서비스은(는) 다른 AWS 서비스의 특성을 사용합니다. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스을(를) 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스을(를) 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
- 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.

- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 자격 증명을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 생성하고 AWS ID 또는 리소스에 연결하여 AWS에서 내 액세스를 제어합니다. 정책은 ID 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS은(는) 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI또는 AWSAPI에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지

를 제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 IAM 사용 설명서의 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우, 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스가 포함될 수 있습니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3, AWS WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS는 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포

함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.

- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations는 기업이 소유하는 여러 개의 AWS 계정을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 특성을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔티티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.
- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 관계없이 AWS 계정 루트 사용자를 포함한 ID에 대한 유효 권한에 영향을 줄 수 있습니다. RCP를 지원하는 AWS 서비스 목록을 포함하여 Organizations 및 RCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS Glue의 작동 방식 IAM

IAM 를 사용하여 AWS Glue에 대한 액세스를 관리하기 전에 AWS Glue에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

IAM AWS Glue와 함께 사용할 수 있는 기능

IAM 기능	AWS Glue 지원
ID 기반 정책	예

IAM 기능	AWS Glue 지원
리소스 기반 정책	부분
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	아니요
ABAC (정책의 태그)	부분
임시 보안 인증	예
보안 주체 권한	아니요
서비스 역할	예
서비스 연결 역할	아니요

AWS Glue 및 기타 AWS 서비스가 대부분의 IAM 기능에서 작동하는 방식을 개괄적으로 알아보려면 IAM 사용 설명서의 에서 AWS 로 작업하는 서비스를 IAM 참조하세요.

AWS Glue에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 고객 관리형 정책을 사용하여 사용자 지정 IAM 권한 정의를 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부된 작업 및 리소스와 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

AWS Glue 는 모든 에 대해 자격 증명 기반 정책(IAM 정책)을 지원합니다.AWS Glue 작업. 정책을 연결하여 를 생성, 액세스 또는 수정할 수 있는 권한을 부여할 수 있습니다.AWS Glue 리소스, 의 테이블 등 AWS Glue Data Catalog.

AWS Glue에 대한 자격 증명 기반 정책 예제

AWS Glue 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS Glue 자격 증명 기반 정책 예제.

AWS Glue 내의 리소스 기반 정책

리소스 기반 정책 지원: 부분적

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려면 리소스 기반 정책의 보안 주체로 전체 계정 또는 다른 계정의 IAM엔터티를 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 에 있는 경우 신뢰할 수 AWS 계정있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 에서 크로스 계정 리소스 액세스를 IAM 참조하세요.

> **Note**
>
> 만 사용할 수 있습니다.AWS Glue Data Catalog 리소스에 대한 권한을 관리하기 위한 리소스 정책입니다. 다른 에 연결할 수 없습니다.AWS Glue 작업, 트리거, 개발 엔드포인트, 크롤러 또는 분류기와 같은 리소스.
> 카탈로그당 하나의 리소스 정책만 허용되며, 크기는 10KB로 제한됩니다.

AWS Glue에서 리소스 정책은 앞서 언급한 모든 종류의 Data Catalog 리소스를 위한 가상 컨테이너인 카탈로그 에 연결됩니다. 각 AWS 계정은 카탈로그 ID가 AWS 계정 ID와 동일한 AWS 리전의 단일 카탈로그를 소유합니다. 카탈로그는 삭제하거나 수정할 수 없습니다.

발신자 보안 주체가 정책 문서의 "Principal" 블록에 포함된 카탈로그에 대한 모든 API 호출에 대해 리소스 정책이 평가됩니다.

AWS Glue 리소스 기반 정책의 예를 보려면 섹션을 참조하세요AWS Glue용 리소스 기반 정책 예제.

AWS Glue에 대한 정책 작업

정책 작업 지원: 예

관리자는 정책을 사용하여 AWS JSON 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업과 같은 몇 가지 예외가 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

AWS Glue 작업 목록을 보려면 서비스 승인 참조의 AWS Glue에서 정의한 작업을 참조하세요.

AWS Glue의 정책 작업은 작업 전에 다음 접두사를 사용합니다.

```
glue
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "glue:action1",
      "glue:action2"
         ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Get라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "glue:Get*"
```

정책 예시를 보려면 AWS Glue 액세스 제어 정책 예제를 참조하세요.

AWS Glue에 대한 정책 리소스

정책 리소스 지원: 예

관리자는 정책을 사용하여 AWS JSON 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 객체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 가장 좋은 방법은 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정하는 것입니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

를 사용하여 AWS Glue 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 섹션을 ARNs참조하세요AWS Glue 리소스 ARN 지정.

AWS Glue 리소스 유형 및 해당 의 목록을 보려면 서비스 승인 참조의 AWS Glue에서 정의한 리소스를 ARNs참조하세요. 각 리소스ARN의 를 지정하는 데 사용할 수 있는 작업을 알아보려면 AWS Glue에서 정의한 작업을 참조하세요.

AWS Glue의 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 정책을 사용하여 AWS JSON 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 작업을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어 IAM 리소스에 사용자 IAM 이름으로 태그가 지정된 경우에만 사용자에게 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

AWS Glue 조건 키 목록을 보려면 서비스 승인 참조의 AWS Glue에 대한 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면 AWS Glue에서 정의한 작업을 참조하세요.

정책 예시를 보려면 조건 키 또는 컨텍스트 키를 사용하여 설정 제어를 참조하세요.

ACLs AWS Glue 내

지원 ACLs: 아니요

액세스 제어 목록(ACLs)은 리소스에 액세스할 수 있는 권한이 있는 보안 주체(계정 멤버, 사용자 또는 역할)를 제어합니다. ACLs 는 리소스 기반 정책과 유사하지만 JSON 정책 문서 형식을 사용하지는 않습니다.

ABAC AWS Glue 사용

지원ABAC(정책의 태그): 부분

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그 라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS 리소스에 태그를 연결할 수 있습니다. 엔터티 및 리소스에 태그를 지정하는 것은 의 첫 번째 단계입니다ABAC. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계합니다.

ABAC 는 빠르게 성장하는 환경에서 도움이 되며 정책 관리가 번거로워지는 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 사용하여 권한 정의를 ABAC참조하세요. 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어 사용(ABAC)을 ABAC참조하세요.

> **Important**
>
> 조건 컨텍스트 키는 에만 적용됩니다.AWS Glue API 크롤러, 작업, 트리거 및 개발 엔드포인트에 대한 작업. 영향을 받는 API 작업에 대한 자세한 내용은 AWS Glue 의 조건 키를 참조하세요.
>
> 는 AWS Glue Data Catalog API 작업은 현재 `aws:referer` 및 `aws:UserAgent` 전역 조건 컨텍스트 키를 지원하지 않습니다.

리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예시는 태그를 사용한 액세스 권한 부여에서 확인할 수 있습니다.

AWS Glue에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 보안 인증 정보를 사용하여 로그인할 때 일부는 작동하지 AWS 서비스 않습니다. 임시 자격 증명으로 AWS 서비스 작업하는 것을 비롯한 자세한 내용은 IAM 사용 설명서의 AWS 서비스 에서 작업하는 IAM 섹션을 참조하세요.

사용자 이름 및 암호를 제외한 방법을 AWS Management Console 사용하여 에 로그인하는 경우 임시 보안 인증 정보를 사용합니다. 예를 들어 회사의 Single Sign-On(SSO) 링크를 AWS 사용하여 에 액세스하면 해당 프로세스가 임시 자격 증명을 자동으로 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 사용자에서 IAM 역할(콘솔)로 전환을 참조하세요.

AWS CLI 또는 를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS API. 그런 다음 이러한 임시 자격 증명을 사용하여 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 AWS. AWS recommends에 액세스할 수 있습니다. 자세한 내용은 의 임시 보안 자격 증명을 IAM참조하세요.

AWS Glue에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션 지원(FAS): 아니요

IAM 사용자 또는 역할을 사용하여 에서 작업을 수행 AWS하면 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS 는 를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할

때만 수행됩니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 액세스 세션 전달을 참조하세요.

AWS Glue의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM 역할입니다. IAM 관리자는 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다IAM. 자세한 내용은 IAM 사용 설명서의 에 권한을 위임할 역할 생성을 AWS 서비스 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 AWS Glue 기능이 중단될 수 있습니다. AWS Glue가 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

AWS Glue에 대한 서비스 역할 생성에 대한 자세한 지침은 1단계: AWS Glue 서비스를 위한 IAM 정책 생성 및 섹션을 참조하세요2단계: AWS Glue에 대한 IAM 역할 생성.

AWS Glue에 대한 서비스 연결 역할

서비스 링크 역할 지원: 아니요

서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 AWS 에서 작동하는 서비스를 참조하세요IAM. 서비스 연결 역할 열에서 Yes(이)가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

AWS Glue에 대한 IAM 권한 구성

AWS Identity and Access Management(IAM)를 사용하여 AWS Glue에서 리소스에 액세스하는 데 사용하는 정책과 역할을 정의합니다. 다음 단계는 AWS Glue에 대한 권한 설정을 위한 다양한 옵션을 소개합니다. 비즈니스의 요구 사항에 따라 리소스 액세스를 추가하거나 줄여야 할 경우도 있습니다.

> **Note**
>
> 대신 AWS Glue에 대한 기본 IAM 권한으로 시작하려면 AWS Glue에 대한 IAM 권한 설정 섹션을 참조하세요.

1. AWS Glue 서비스를 위한 IAM 정책 생성: AWS Glue 리소스에 액세스하도록 허용하는 서비스 정책을 생성합니다.
2. AWS Glue에 대한 IAM 역할 생성: IAM 역할을 생성하고 AWS Glue에서 사용하는 Amazon Simple Storage Service(S3) 리소스에 대한 AWS Glue 서비스 정책 및 정책을 연결합니다.
3. AWS Glue에 액세스하는 사용자 또는 그룹에 정책 연결: AWS Glue 콘솔에 로그인하는 모든 사용자 또는 그룹에 정책을 연결합니다.
4. 노트북용 IAM 정책 생성: 개발 엔드포인트에서 노트북 서버 생성에 사용할 노트북 서버 정책을 생성합니다.
5. 노트북용 IAM 역할 생성: IAM 역할을 생성하고 노트북 서버 정책을 연결합니다.
6. Amazon SageMaker AI 노트북에 대한 IAM 정책 생성: 개발 엔드포인트에서 Amazon SageMaker AI 노트북을 생성할 때 사용할 IAM 정책을 생성합니다.
7. Amazon SageMaker AI 노트북용 IAM 역할 생성: IAM 역할을 생성하고 개발 엔드포인트에서 Amazon SageMaker AI 노트북을 생성할 때 권한을 부여하는 정책을 연결합니다.

1단계: AWS Glue 서비스를 위한 IAM 정책 생성

Amazon S3 객체에 액세스하는 것처럼 다른 AWS 리소스의 데이터에 액세스하는 작업의 경우, 사용자를 대신하여 리소스에 액세스할 수 있는 권한이 AWS Glue에 있어야 합니다. AWS Identity and Access Management(IAM)을 사용하여 그러한 권한을 제공합니다.

> **Note**
>
> AWS 관리형 정책 **`AWSGlueServiceRole`**을 사용하면 이 단계를 생략해도 좋습니다.

이 단계에서는`AWSGlueServiceRole`과 비슷한 정책을 만듭니다. IAM 콘솔에서 `AWSGlueServiceRole`의 최신 버전을 확인할 수 있습니다.

AWS Glue를 위한 IAM 정책을 생성하려면

이 정책은 몇 가지 Amazon S3 작업에 대한 권한을 부여하여 이 정책에 따른 역할을 AWS Glue에서 맡을 때 필요한 계정 내 리소스를 관리할 수 있도록 합니다. 이 정책에 지정된 몇 가지 리소스는 Amazon S3 버킷, Amazon S3 ETL 스크립트, CloudWatch Logs 및 Amazon EC2 리소스에 대해 AWS Glue에서 사용하는 기본 이름을 나타냅니다. 간소화를 위해 AWS Glue는 기본적으로 `aws-glue-*`가 접두사인 계정에서 몇 가지 Amazon S3 객체를 버킷에 작성합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 생성(Create Policy)을 선택합니다.
4. [Create Policy(정책 생성)] 화면에서 탭으로 이동하여 JSON을 편집합니다. 다음 JSON 설명으로 정책 문서를 만든 다음 [Review policy(정책 보기)]를 선택합니다.

> **Note**
>
> Amazon S3 리소스에 필요한 모든 액세스를 추가합니다. 필요한 리소스만의 액세스 정책에 대한 리소스 섹션의 범위를 정하고자 할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeRouteTables",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
```

```
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*/*",
                "arn:aws:s3:::*/*aws-glue-*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::crawler-public*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:AssociateKmsKey"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws-glue/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "aws-glue-service-resource"
                    ]
                }
            },
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:instance/*"
            ]
        }
    ]
}
```

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
"glue:*"	"*"	모든 AWS Glue API 작업을 실행할 수 있는 권한을 부여합니다.

작업	리소스	설명
"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl",	"*"	크롤러, 작업, 개발 엔드포인트 및 노트북 서버의 Amazon S3 버킷 목록을 허용합니다.
"ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:CreateNetworkInterface", "ec2:DeleteNetworkInterface", "ec2:DescribeNetworkInterfaces", "ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcAttribute",	"*"	작업, 크롤러 및 개발 엔드포인트를 실행할 때 Virtual Private Cloud(VPC)와 같은 Amazon EC2 네트워크 항목을 설정할 수 있도록 허용합니다.
"iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy"	"*"	크롤러, 작업, 개발 엔드포인트 및 노트북 서버의 IAM 역할 목록을 허용합니다.
"cloudwatch:PutMetricData"	"*"	작업에 대한 CloudWatch 지표 작성을 허용합니다.
"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"	"arn:aws:s3:::aws-glue-*"	작업과 노트북 서버로부터 계정의 Amazon S3 버킷 생성을 허용합니다. 명명 규칙: aws-glue-라는 Amazon S3 폴더를 사용합니다. AWS Glue를 사용하여 퍼블릭 액세스를 차단하는 버킷을 생성합니다.

작업	리소스	설명
"s3:GetObject", "s3:PutObject", "s3:DeleteObject"	"arn:aws:s3:::aws-glue-*/*", "arn:aws:s3:::*/*aws-glue-*/*"	ETL 스크립트 및 노트북 서버 위치와 같은 객체를 저장할 때 Amazon S3 객체를 계정에서 얻고 넣고 삭제하는 것을 허용합니다. 명명 규칙: 이름이 aws-glue-로 시작하는 Amazon S3 버킷 또는 폴더에 권한을 부여합니다.
"s3:GetObject"	"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"	크롤러 및 작업의 예제 및 튜토리얼이 사용한 Amazon S3 객체 얻기를 허용합니다. 명명 규칙: Amazon S3 버킷 이름은 crawler-public 및 aws-glue-로 시작합니다.
"logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents"	"arn:aws:logs:*:*:log-group:/aws-glue/*"	CloudWatch Logs에 로그를 쓸 수 있습니다. 이름 전환: AWS Glue는 aws-glue로 시작하는 로그 투 로그 그룹 이름을 작성합니다.
"ec2:CreateTags", "ec2:DeleteTags"	"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"	개발 엔드포인트를 생성하는 Amazon EC2 리소스 태깅을 허용합니다. 이름 전환: AWS Glue는 Amazon EC2 네트워크 인터페이스, 보안 그룹 및 인스턴스를 aws-glue-service-resource에 태그 지정합니다.

5. 정책 검토 화면에서 정책 이름을 입력합니다(예: GlueServiceRolePolicy). 조건부 설명을 입력하고 정책에 만족하면 정책 생성을 선택합니다.

2단계: AWS Glue에 대한 IAM 역할 생성

AWS Glue가 사용자 대신 다른 서비스를 호출할 때 이용할 수 있도록 IAM 역할 권한을 부여해야 합니다. 여기에는 AWS Glue에서 사용하는 소스, 대상, 스크립트 및 임시 디렉터리에 대한 Amazon S3 액세스 권한이 포함됩니다. 크롤러, 작업 및 개발 엔드포인트는 권한이 필요합니다.

AWS Identity and Access Management(IAM)을 사용하여 그러한 권한을 제공합니다. IAM 역할에 정책을 추가하여 AWS Glue에 전달합니다.

AWS Glue용 IAM 역할 생성

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 신뢰할 수 있는 엔터티 유형으로서 AWS 서비스를 선택합니다. 그런 다음, 서비스 또는 사용 사례의 경우 AWS Glue을(를) 찾아서 선택합니다. 다음을 선택합니다.
5. 권한 추가 페이지에서 필요한 권한이 포함된 정책을 선택합니다. 예를 들어 일반 AWS Glue 권한에 대한 AWS 관리형 정책 `AWSGlueServiceRole`와(과) Amazon S3 리소스에 대한 액세스에 대한 AWS 관리형 정책 AmazonS3FullAccess가 있습니다. 그런 다음 다음을 선택합니다.

> **Note**
>
> 이 역할의 정책 중 하나가 Amazon S3 소스 및 대상에 대한 권한을 보장한다는 것을 확인합니다. 특정 Amazon S3 리소스에 액세스를 위한 자체 정책을 제공하고자 할 수 있습니다. 데이터 원본은 `s3:ListBucket` 및 `s3:GetObject` 권한을 요구합니다. 데이터 대상은 `s3:ListBucket`, `s3:PutObject` 및 `s3:DeleteObject` 권한을 요구합니다. 리소스에 대한 Amazon S3 정책을 생성하는 방법은 정책에서 리소스 지정을 참조하세요. Amazon S3 정책 예제는 IAM 정책 작성하기: Amazon S3 버킷으로의 액세스를 보장하는 방법을 참조하세요.
> SSE-KMS로 암호화된 Amazon S3 소스 및 대상에 액세스하고자 할 경우, AWS Glue 크롤러, 작업 및 개발 엔드포인트가 데이터를 복호화할 수 있게 허용하는 정책을 연결합니다. 자세한 내용은 AWS KMS 관리형 키(SSE-KMS)를 사용하는 서버 측 암호화로 데이터 보호를 참조하세요.

다음은 예입니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
         ]
      }
   ]
}
```

6. 역할 이름을 지정하고 설명(선택 사항)을 추가한 다음, 신뢰 정책 및 권한을 검토하세요. [역할 이름(Role name)]에 역할 이름을 입력합니다(예: `AWSGlueServiceRoleDefault`). 이름의 접두사가 문자열 `AWSGlueServiceRole`인 역할을 생성하고, 콘솔 사용자가 서비스로 해당 역할을 전달할 수 있도록 합니다. AWS Glue가 제공한 정책에서는 IAM 서비스 역할이 `AWSGlueServiceRole`로 시작될 것으로 예상합니다. 그렇지 않으면 정책을 추가하여 IAM 역할에 대한 `iam:PassRole` 권한을 사용자에게 부여하고 이름 규정과 일치시켜야 합니다. 역할 생성을 선택합니다.

> **Note**
>
> 역할이 있는 노트북을 생성하면 해당 역할이 대화형 세션으로 전달되므로 두 위치에서 동일한 역할을 사용할 수 있습니다. 따라서 `iam:PassRole` 권한은 역할의 정책의 일부여야 합니다.
> 다음 예제를 사용하여 역할에 대한 새 정책을 생성합니다. 계정 번호를 사용자의 고유한 역할 이름으로 바꿉니다.
>
> ```
> {
>     "Version": "2012-10-17",
>     "Statement": [
>         {
>             "Effect": "Allow",
>             "Action": "iam:PassRole",
>             "Resource": "arn:aws:iam::090000000210:role/<role_name>"
> ```

```
        }
    ]
}
```

7. 규칙에 태그를 추가합니다(선택 사항). 태그는 리소스를 식별, 구성 또는 검색하는 데 도움이 되는 AWS 리소스에 추가할 수 있는 키-값 쌍입니다. 그런 다음 역할 생성을 선택합니다.

3단계: AWS Glue에 액세스하는 사용자 또는 그룹에 정책 연결

관리자는 AWS Glue 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 모든 사용자, 그룹 또는 역할에 권한을 할당해야 합니다. 정책을 통해 AWS Identity and Access Management(IAM)를 사용하여 권한을 제공합니다. 이 단계에서는 사용자 또는 그룹에 권한을 할당하는 방법을 설명합니다.

이 단계가 완료되면 사용자 또는 그룹은 다음과 같은 정책이 연결됩니다.

- AWS 관리형 정책 `AWSGlueConsoleFullAccess` 또는 사용자 정책 GlueConsoleAccessPolicy
- **`AWSGlueConsoleSageMakerNotebookFullAccess`**
- **`CloudWatchLogsReadOnlyAccess`**
- **`AWSCloudFormationReadOnlyAccess`**
- **`AmazonAthenaFullAccess`**

인라인 정책을 연결하여 사용자 또는 그룹에 추가하는 방법

AWS 관리형 정책 또는 인라인 정책을 사용자 또는 그룹에 연결하여 AWS Glue 콘솔에 액세스할 수 있습니다. 이 정책에 지정된 몇 가지 리소스는 Amazon S3 버킷, Amazon S3 ETL 스크립트, CloudWatch Logs, AWS CloudFormation 및 Amazon EC2 리소스에 대해 AWS Glue에서 사용하는 기본 이름을 나타냅니다. 간소화를 위해 AWS Glue는 기본적으로 `aws-glue-*`가 접두사인 계정에서 몇 가지 Amazon S3 객체를 버킷에 작성합니다.

> **Note**
>
> AWS 관리형 정책 **`AWSGlueConsoleFullAccess`**를 사용하면 이 단계를 생략해도 좋습니다.

> **Important**
>
> AWS Glue는 사용자를 대신하여 작업을 수행하는 데 사용되는 역할에 대한 권한이 필요합니다. 이 과정을 완료하려면 **`iam:PassRole`** 권한을 AWS Glue 사용자 또는 그룹에 부여합니다. 이 정책은 AWS Glue 서비스 역할의 경우 `AWSGlueServiceRole`로 시작되는 역할에 권한을 부여하고, 노트북 서버를 생성할 때 필요한 역할 중에는 `AWSGlueServiceNotebookRole`로 시작하는 역할에 권한을 부여합니다. 이름 전환에 따라 `iam:PassRole` 권한에 대한 자체 정책을 생성할 수도 있습니다.
>
> 보안 모범 사례에 따라 Amazon S3 버킷 및 Amazon CloudWatch 로그 그룹에 대한 액세스를 추가로 제한하는 정책을 강화하여 액세스를 제한하는 것이 좋습니다. Amazon S3 정책 예제는 IAM 정책 작성하기: Amazon S3 버킷으로의 액세스를 보장하는 방법을 참조하세요.

이 단계에서는`AWSGlueConsoleFullAccess`과 비슷한 정책을 만듭니다. IAM 콘솔에서 `AWSGlueConsoleFullAccess`의 최신 버전을 확인할 수 있습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 또는 사용자 그룹을 선택합니다.
3. 목록에서 정책을 삽입할 사용자 또는 그룹 이름을 선택합니다.
4. 권한 탭을 선택하고 필요하다면 Permissions policies(권한 정책) 섹션을 확장합니다.
5. [Add Inline Policy(인라인 정책 추가)] 링크를 선택합니다.
6. [Create Policy(정책 생성)] 화면에서 탭으로 이동하여 JSON을 편집합니다. 다음 JSON 설명으로 정책 문서를 만든 다음 [Review policy(정책 보기)]를 선택합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "redshift:DescribeClusters",
                "redshift:DescribeClusterSubnetGroups",
                "iam:ListRoles",
                "iam:ListUsers",
                "iam:ListGroups",
                "iam:ListRolePolicies",
```

```
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeRouteTables",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeInstances",
                "rds:DescribeDBInstances",
                "rds:DescribeDBClusters",
                "rds:DescribeDBSubnetGroups",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "cloudformation:DescribeStacks",
                "cloudformation:GetTemplateSummary",
                "dynamodb:ListTables",
                "kms:ListAliases",
                "kms:DescribeKey",
                "cloudwatch:GetMetricData",
                "cloudwatch:ListDashboards"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*/*aws-glue-*/*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPublicAccessBlock"           ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:/aws-glue/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack"
            ],
            "Resource": "arn:aws:cloudformation:*:*:stack/aws-glue*/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:image/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:subnet/*",
```

```
                "arn:aws:ec2:*:*:volume/*"
            ]
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceNotebookRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/service-role/AWSGlueServiceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
```

```
                }
            }
        }
    ]
}
```

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
"glue:*"	"*"	모든 AWS Glue API 작업을 실행할 수 있는 권한을 부여합니다. "glue:*" 작업 없이 이전에 정책을 생성했다면 정책에 다음과 같은 개별 권한을 추가해야 합니다. • "glue:ListCrawlers" • "glue:BatchGetCrawlers" • "glue:ListTriggers" • "glue:BatchGetTriggers" • "glue:ListDevEndpoints" • "glue:BatchGetDevEndpoints" • "glue:ListJobs" • "glue:BatchGetJobs"
"redshift:DescribeClusters", "redshift:DescribeClusterSubnetGroups"	"*"	Amazon Redshift로 연결을 생성하도록 허용합니다.

작업	리소스	설명
"iam:ListRoles", "iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy", "iam:ListAttachedRolePolicies"	"*"	크롤러, 작업, 개발 엔드포인트 및 노트북 서버와 작업할 경우 IAM 역할 목록을 허용합니다.
"ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcs", "ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:DescribeVpcAttribute", "ec2:DescribeKeyPairs", "ec2:DescribeInstances"	"*"	작업, 크롤러 및 개발 엔드포인트를 실행할 때 VPC와 같은 Amazon EC2 네트워크 항목의 설치를 허용합니다.
"rds:DescribeDBInstances"	"*"	Amazon RDS로 연결을 생성하도록 허용합니다.
"s3:ListAllMyBuckets", "s3:ListBucket", "s3:GetBucketAcl", "s3:GetBucketLocation"	"*"	크롤러, 작업, 개발 엔드포인트 및 노트북 서버와 작업할 경우, Amazon S3 버킷 목록을 허용합니다.
"dynamodb:ListTables"	"*"	DynamoDB 테이블 나열을 허용합니다.
"kms:ListAliases", "kms:DescribeKey"	"*"	KMS 키 작업을 허용합니다.
"cloudwatch:GetMetricData", "cloudwatch:ListDashboards"	"*"	CloudWatch 지표 작업을 허용합니다.

작업	리소스	설명
"s3:GetObject", "s3:PutObject"	"arn:aws:s3::: aws-glue-*/*", "arn:aws:s3::: */*aws-glue-*/*", "arn:aws:s3::: aws-glue-*"	ETL 스크립트 및 노트북 서버 위치와 같은 객체를 저장할 때 Amazon S3 객체를 계정에서 얻고 넣는 것을 허용합니다. 명명 규칙: 이름이 aws-glue-로 시작하는 Amazon S3 버킷 또는 폴더에 권한을 부여합니다.
"tag:GetResources"	"*"	AWS 태그 검색을 허용합니다.
"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"	"arn:aws:s3::: aws-glue-*"	ETL 스크립트 및 노트북 서버 위치와 같은 객체를 저장할 때 Amazon S3 버킷을 계정에 생성하는 것을 허용합니다. 명명 규칙: 이름이 aws-glue-로 시작하는 Amazon S3 버킷에 권한을 부여합니다. AWS Glue를 사용하여 퍼블릭 액세스를 차단하는 버킷을 생성합니다.

작업	리소스	설명
"logs:GetLogEvents"	"arn:aws:logs:*:*: /aws-glue/*"	CloudWatch Logs의 검색 허용 이름 전환: AWS Glue는 aws-glue로 시작하는 로그 투 로그 그룹 이름을 작성합니다.
"cloudformation:CreateStack", "cloudformation:DeleteStack"	"arn:aws:cloudformation:*:*:stack/aws-glue*/*"	노트북 서버로 작업할 때 AWS CloudFormation 스택 관리를 허용합니다. 이름 전환: AWS Glue는 aws-glue로 시작하는 스택 이름을 생성합니다.
"ec2:RunInstances"	"arn:aws:ec2:*:*:instance/*", "arn:aws:ec2:*:*:key-pair/*", "arn:aws:ec2:*:*:image/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:subnet/*", "arn:aws:ec2:*:*:volume/*"	개발 엔드포인트 및 노트북 서버의 실행을 허용합니다.
"iam:PassRole"	"arn:aws:iam::*:role/ AWSGlueServiceRole*"	AWS Glue가 AWSGlueServiceRole 로 시작하는 역할에 대해 PassRole 권한을 수임하도록 허용합니다.

작업	리소스	설명
`"iam:PassRole"`	`"arn:aws:iam::*:role/ AWSGlueServiceNotebookRole*"`	Amazon EC2가 `AWSGlueServiceNotebookRole` 로 시작하는 역할에 대해 `PassRole` 권한을 수임하도록 허용합니다.
`"iam:PassRole"`	`"arn:aws:iam::*:role/service-role/AWSGlueServiceRole*"`	AWS Glue가 `service-role/AWSGlueServiceRole` 로 시작하는 역할에 대해 `PassRole` 권한을 수임하도록 허용합니다.

7. 정책 검토 화면에서 정책의 이름을 입력합니다(예: GlueConsoleAccessPolicy). 정책에 만족하면 정책 생성을 선택합니다. 화면 상단에 있는 빨간색 상자에 표시되는 오류가 있지 않은지 확인합니다. 표시되는 오류가 있다면 수정합니다.

> **Note**
>
> [Use autoformatting]을 선택하면 정책을 열거나 [Validate Policy]를 선택할 때마다 정책의 형식이 다시 지정됩니다.

AWSGlueConsoleFullAccess 관리 정책을 연결하는 방법

`AWSGlueConsoleFullAccess` 정책을 추가하여 AWS Glue 콘솔 사용자가 필요한 권한을 제공합니다.

> **Note**
>
> AWS Glue 콘솔 액세스용 자체 정책을 생성했다면 이 단계를 건너뛸 수 있습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 AWSGlueConsoleFullAccess 옆의 확인란을 선택합니다. [Filter] 메뉴와 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 조치를 선택한 후 연결을 선택합니다.
5. 정책을 연결하려는 사용자를 선택합니다. 필터 메뉴와 검색 상자를 사용하면 보안 주체 개체 목록을 필터링할 수 있습니다. 정책을 추가할 사용자를 선택한 다음 [Attach policy(정책 추가)]를 선택합니다.

`AWSGlueConsoleSageMakerNotebookFullAccess` 관리형 정책을 연결하려면

`AWSGlueConsoleSageMakerNotebookFullAccess` 정책을 사용자에게 연결해 AWS Glue 콘솔에서 생성한 SageMaker AI 노트북을 관리할 수 있습니다. 필요한 다른 AWS Glue 콘솔 권한 이외에도 이 정책은 SageMaker AI 노트북 관리에 필요한 리소스에 대한 액세스 권한을 부여합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 AWSGlueConsoleSageMakerNotebookFullAccess 옆의 확인란을 선택합니다. [Filter] 메뉴와 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 조치를 선택한 후 연결을 선택합니다.
5. 정책을 연결하려는 사용자를 선택합니다. 필터 메뉴와 검색 상자를 사용하면 보안 주체 개체 목록을 필터링할 수 있습니다. 정책을 추가할 사용자를 선택한 다음 [Attach policy(정책 추가)]를 선택합니다.

CloudWatchLogsReadOnlyAccess 관리 정책을 연결하는 방법

CloudWatchLogsReadOnlyAccess 정책을 사용자에게 연결하고 CloudWatch Logs 콘솔에서 AWS Glue가 생성한 로그를 볼 수 있습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 [CloudWatchLogsReadOnlyAccess] 이름 옆의 확인란을 선택합니다. [Filter] 메뉴와 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 조치를 선택한 후 연결을 선택합니다.

5. 정책을 연결하려는 사용자를 선택합니다. 필터 메뉴와 검색 상자를 사용하면 보안 주체 개체 목록을 필터링할 수 있습니다. 정책을 추가할 사용자를 선택한 다음 [Attach policy(정책 추가)]를 선택합니다.

AWSCloudFormationReadOnlyAccess 관리 정책을 연결하는 방법

AWSCloudFormationReadOnlyAccess 정책을 사용자에게 연결하고 AWS Glue가 사용한 AWS CloudFormation 스택을 AWS CloudFormation 콘솔에서 볼 수 있습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 AWSCloudFormationReadOnlyAccess 이름 옆의 확인란을 선택합니다. [Filter] 메뉴와 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 조치를 선택한 후 연결을 선택합니다.
5. 정책을 연결하려는 사용자를 선택합니다. 필터 메뉴와 검색 상자를 사용하면 보안 주체 개체 목록을 필터링할 수 있습니다. 정책을 추가할 사용자를 선택한 다음 [Attach policy(정책 추가)]를 선택합니다.

AmazonAthenaFullAccess 관리 정책을 추가하는 방법

AmazonAthenaFullAccess 정책을 사용자에게 연결하고 Amazon S3 데이터를 Athena 콘솔에서 볼 수 있습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 [AmazonAthenaFullAccess(Amazon Athena 완전한 액세스)] 옆의 확인란을 선택합니다. [Filter] 메뉴와 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 조치를 선택한 후 연결을 선택합니다.
5. 정책을 연결하려는 사용자를 선택합니다. 필터 메뉴와 검색 상자를 사용하면 보안 주체 개체 목록을 필터링할 수 있습니다. 정책을 추가할 사용자를 선택한 다음 [Attach policy(정책 추가)]를 선택합니다.

4단계: 노트북 서버용 IAM 정책 생성

개발 엔드포인트로 노트북을 사용하고자 한다면 노트북 서버를 생성할 때 권한을 지정해야 합니다. AWS Identity and Access Management(IAM)을 사용하여 그러한 권한을 제공합니다.

이 정책은 몇 가지 Amazon S3 작업에 대한 권한을 부여하여 이 정책에 따른 역할을 AWS Glue에서 맡을 때 필요한 계정 내 리소스를 관리할 수 있도록 합니다. 이 정책에 지정된 몇 가지 리소스는 Amazon S3 버킷, Amazon S3 ETL 스크립트, Amazon EC2 리소스에 대해 AWS Glue에서 사용하는 기본 이름을 나타냅니다. 간소화를 위해 AWS Glue는 기본적으로 접두사가 `aws-glue-*`인 계정에서 몇 가지 Amazon S3 객체를 버킷에 작성합니다.

> **Note**
>
> AWS 관리형 정책 **`AWSGlueServiceNotebookRole`**를 사용하면 이 단계를 생략해도 좋습니다.

이 단계에서는`AWSGlueServiceNotebookRole`과 비슷한 정책을 만듭니다. IAM 콘솔에서 `AWSGlueServiceNotebookRole`의 최신 버전을 확인할 수 있습니다.

노트북에 대한 IAM 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 생성(Create Policy)을 선택합니다.
4. [Create Policy(정책 생성)] 화면에서 탭으로 이동하여 JSON을 편집합니다. 다음 JSON 설명으로 정책 문서를 만든 다음 [Review policy(정책 보기)]를 선택합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:CreateDatabase",
            "glue:CreatePartition",
            "glue:CreateTable",
            "glue:DeleteDatabase",
            "glue:DeletePartition",
```

```
            "glue:DeleteTable",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:GetTable",
            "glue:GetTableVersions",
            "glue:GetTables",
            "glue:UpdateDatabase",
            "glue:UpdatePartition",
            "glue:UpdateTable",
            "glue:GetJobBookmark",
            "glue:ResetJobBookmark",
            "glue:CreateConnection",
            "glue:CreateJob",
            "glue:DeleteConnection",
            "glue:DeleteJob",
            "glue:GetConnection",
            "glue:GetConnections",
            "glue:GetDevEndpoint",
            "glue:GetDevEndpoints",
            "glue:GetJob",
            "glue:GetJobs",
            "glue:UpdateJob",
            "glue:BatchDeleteConnection",
            "glue:UpdateConnection",
            "glue:GetUserDefinedFunction",
            "glue:UpdateUserDefinedFunction",
            "glue:GetUserDefinedFunctions",
            "glue:DeleteUserDefinedFunction",
            "glue:CreateUserDefinedFunction",
            "glue:BatchGetPartition",
            "glue:BatchDeletePartition",
            "glue:BatchCreatePartition",
            "glue:BatchDeleteTable",
            "glue:UpdateDevEndpoint",
            "s3:GetBucketLocation",
            "s3:ListBucket",
            "s3:ListAllMyBuckets",
            "s3:GetBucketAcl"
        ],
        "Resource":[
            "*"
        ]
```

```
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::crawler-public*",
            "arn:aws:s3:::aws-glue*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:DeleteObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-glue*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:CreateTags",
            "ec2:DeleteTags"
         ],
         "Condition":{
            "ForAllValues:StringEquals":{
               "aws:TagKeys":[
                  "aws-glue-service-resource"
               ]
            }
         },
         "Resource":[
            "arn:aws:ec2:*:*:network-interface/*",
            "arn:aws:ec2:*:*:security-group/*",
            "arn:aws:ec2:*:*:instance/*"
         ]
      }
   ]
}
```

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
`"glue:*"`	`"*"`	모든 AWS Glue API 작업을 실행할 수 있는 권한을 부여합니다.
`"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl"`	`"*"`	노트북 서버의 Amazon S3 버킷 목록을 허용합니다.
`"s3:GetObject"`	`"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"`	노트북의 예제 및 튜토리얼이 사용한 Amazon S3 객체 얻기를 허용합니다. 명명 규칙: Amazon S3 버킷 이름은 crawler-public 및 aws-glue-로 시작합니다.
`"s3:PutObject", "s3:DeleteObject"`	`"arn:aws:s3:::aws-glue*"`	노트북의 계정으로 Amazon S3 객체를 넣고 삭제하는 것을 허용합니다. 명명 규칙: aws-glue라는 Amazon S3 폴더를 사용합니다.

작업	리소스	설명
`"ec2:CreateTags", "ec2:DeleteTags"`	`"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"`	노트북 서버를 생성하는 Amazon EC2 리소스 태깅을 허용합니다. 이름 전환: AWS Glue는 Amazon EC2 인스턴스를 aws-glue-service-resource에 태그 지정합니다.

5. 정책 검토 화면에서 정책 이름을 입력합니다(예: GlueServiceNotebookPolicyDefault). 조건부 설명을 입력하고 정책에 만족하면 정책 생성을 선택합니다.

5단계: 노트북 서버용 IAM 역할 생성

개발 엔드포인트로 노트북을 사용하고자 한다면 IAM 역할 권한을 부여해야 합니다. IAM 역할을 통해 AWS Identity and Access Management IAM을 사용하여 권한을 제공합니다.

> **Note**
>
> IAM 콘솔을 사용하여 IAM 역할을 생성하면 인스턴스 프로파일이 자동으로 생성되고 해당 역할과 동일한 이름이 지정됩니다.

노트북용 IAM 역할을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 역할 유형의 경우, AWS 서비스를 선택하고 EC2를 찾아 선택한 다음 EC2 사용 사례를 선택하고 다음: 권한을 선택합니다.
5. 권한 정책 연결(Attach permissions policy) 페이지에서 필요한 권한이 포함된 정책을 선택합니다. 예를 들어 일반 AWS Glue 권한에 대한 AWSGlueServiceNotebookRole과 Amazon S3 리소스에

대한 액세스에 대한 AWS 관리형 정책 AmazonS3FullAccess가 있습니다. 그런 다음 다음: 검토(Next: Review)를 선택합니다.

> **Note**
>
> 이 역할의 정책 중 하나가 Amazon S3 소스 및 대상에 대한 권한을 보장한다는 것을 확인합니다. 정책이 노트북 서버를 생성할 때 노트북을 저장하는 위치로 완전한 액세스가 허용되는지 확인합니다. 특정 Amazon S3 리소스에 액세스를 위한 자체 정책을 제공하고자 할 수 있습니다. 리소스에 대한 Amazon S3 정책을 생성하는 방법은 정책에서 리소스 지정을 참조하세요.
> SSE-KMS로 암호화된 Amazon S3 소스 및 대상에 액세스하고자 할 경우, 노트북이 데이터를 복호화할 수 있게 허용하는 정책을 연결합니다. 자세한 내용은 AWS KMS 관리형 키(SSE-KMS)를 사용하는 서버 측 암호화로 데이터 보호를 참조하세요.
> 다음은 예입니다.
>
> ```
> {
>    "Version":"2012-10-17",
>    "Statement":[
>       {
>          "Effect":"Allow",
>          "Action":[
>             "kms:Decrypt"
>          ],
>          "Resource":[
>             "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
>          ]
>       }
>    ]
> }
> ```

6. 역할 이름에 역할의 이름을 입력합니다. 이름의 접두사가 문자열 `AWSGlueServiceNotebookRole`인 역할을 생성하고, 콘솔 사용자가 노트북 서버로 해당 역할을 전달할 수 있도록 합니다. AWS Glue가 제공한 정책에서는 IAM 서비스 역할이 `AWSGlueServiceNotebookRole`로 시작될 것으로 예상합니다. 그렇지 않으면 사용자에게 정책을 추가하여 IAM 역할의 `iam:PassRole` 권한이 이름 규정과 일치하도록 해야 합니다. 예를 들면 `AWSGlueServiceNotebookRoleDefault`를(을) 입력합니다. 그런 다음 역할 생성을 선택합니다.

6단계: SageMaker AI 노트북용 IAM 정책 생성

개발 엔드포인트에서 SageMaker AI 노트북을 사용할 계획이라면 노트북을 생성할 때 권한을 지정해야 합니다. AWS Identity and Access Management(IAM)을 사용하여 그러한 권한을 제공합니다.

SageMaker AI 노트북용 IAM 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 생성(Create Policy)을 선택합니다.
4. 정책 생성 페이지에서 탭으로 이동하여 JSON을 편집합니다. 다음 JSON 문을 사용해 정책 문서를 생성합니다. 환경에 대한 *bucket-name*, *region-code* 및 *account-id*를 편집합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucket-name"
            ]
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucket-name*"
            ]
        },
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
```

```
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:region-code:account-id:log-group:/aws/sagemaker/*",
                "arn:aws:logs:region-code:account-id:log-group:/aws/sagemaker/
*:log-stream:aws-glue-*"
            ]
        },
        {
            "Action": [
                "glue:UpdateDevEndpoint",
                "glue:GetDevEndpoint",
                "glue:GetDevEndpoints"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:glue:region-code:account-id:devEndpoint/*"
            ]
        },
        {
            "Action": [
                "sagemaker:ListTags"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:sagemaker:region-code:account-id:notebook-instance/*"
            ]
         }
    ]
}
```

그런 다음 정책 검토를 선택합니다.

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
"s3:ListBucket*"	"arn:aws:s3::: *bucket-name* "	Amazon S3 버킷을 나열할 수 있는 권한을 부여합니다.

작업	리소스	설명
"s3:GetObject"	"arn:aws:s3::: *bucket-name* *"	SageMaker AI 노트북에서 사용하는 Amazon S3 객체를 가져오도록 권한을 부여합니다.
"logs:CreateLogStream", "logs:DescribeLogStreams", "logs:PutLogEvents", "logs:CreateLogGroup"	"arn:aws:logs: *region-code* :*account-id* :log-group:/aws/sagemaker/*", "arn:aws:logs: *region-code* :*account-id* :log-group:/aws/sagemaker/*:log-stream:aws-glue-*"	노트북에서 Amazon CloudWatch Logs에 로그를 쓸 수 있는 권한을 부여합니다. 이름 지정 규칙: 이름이 aws-glue로 시작하는 로그 그룹에 씁니다.
"glue:UpdateDevEndpoint", "glue:GetDevEndpoint", "glue:GetDevEndpoints"	"arn:aws:glue: *region-code* :*account-id* :devEndpoint/*"	SageMaker AI 노트북에서 개발 엔드포인트를 사용할 수 있는 권한을 부여합니다.
"sagemaker:ListTags"	"arn:aws:sagemaker : *region-code* :*account-id* :notebook-instance /*"	SageMaker AI 리소스에 대한 태그를 반환할 수 있는 권한을 부여합니다. aws-glue-dev-endpoint 태그는 SageMaker AI 노트북을 개발 엔드포인트에 연결하기 위해 필요합니다.

5. [정책 검토(Review Policy)] 화면에서 [정책 이름(Policy Name)]을 입력합니다(예: AWSGlueSageMakerNotebook). 조건부 설명을 입력하고 정책에 만족하면 정책 생성을 선택합니다.

7단계: SageMaker AI 노트북용 IAM 역할 생성

개발 엔드포인트로 SageMaker AI 노트북을 사용하고자 한다면 IAM 역할 권한을 부여해야 합니다. IAM 역할을 통해 AWS Identity and Access Management(IAM)를 사용하여 권한을 제공합니다.

SageMaker AI 노트북용 IAM 역할을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 역할 유형의 경우, AWS 서비스를 선택하고 SageMaker를 찾아 선택한 다음 SageMaker - 실행 사용 사례를 선택합니다. 그런 다음 다음: 권한을 선택합니다.
5. Attach permissions policy(권한 정책 연결) 페이지에서 필요한 권한을 포함한 정책을 선택합니다(예: AmazonSageMakerFullAccess). 다음: 검토를 선택합니다.

   SSE-KMS로 암호화된 Amazon S3 소스 및 대상에 액세스하고자 할 경우, 다음 예에 표시된 것처럼 노트북이 데이터를 복호화하도록 허용하는 정책을 연결합니다. 자세한 내용은 AWS KMS 관리형 키(SSE-KMS)를 사용하는 서버 측 암호화로 데이터 보호를 참조하세요.

   ```
   {
      "Version":"2012-10-17",
      "Statement":[
         {
            "Effect":"Allow",
            "Action":[
               "kms:Decrypt"
            ],
            "Resource":[
               "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
            ]
         }
      ]
   }
   ```

6. 역할 이름에 역할의 이름을 입력합니다. 콘솔 사용자로부터 SageMaker AI로 역할이 전달되도록 하려면 문자열 `AWSGlueServiceSageMakerNotebookRole`로 접두사가 지정된 이름을 사용합니다. AWS Glue에서 제공한 정책은 IAM 역할이 `AWSGlueServiceSageMakerNotebookRole`로 시작할 것이라고 기대합니다. 그렇지 않으면 사용자에게 정책을 추가하여 IAM 역할의 `iam:PassRole` 권한이 이름 규정과 일치하도록 해야 합니다.

   예를 들어 `AWSGlueServiceSageMakerNotebookRole-Default`를 입력한 다음 [역할 생성(Create role)]을 선택합니다.

7. 역할을 생성한 후 AWS Glue에서 SageMaker AI 노트북을 생성하는 데 필요한 추가 권한을 허용하는 정책을 연결합니다.

   방금 생성한 역할인 `AWSGlueServiceSageMakerNotebookRole-Default`를 열고 [정책 연결(Attach policies)]을 선택합니다. 역할에 생성한 `AWSGlueSageMakerNotebook` 정책을 연결합니다.

AWS Glue 액세스 제어 정책 예제

이 섹션에는 자격 증명 기반(IAM) 액세스 제어 정책과 AWS Glue 리소스 정책 모두의 예제가 포함되어 있습니다.

목차

- AWS Glue 자격 증명 기반 정책 예제
  - 정책 모범 사례
  - 리소스 수준 권한은 특정 AWS Glue 객체에만 적용됨
  - AWS Glue Console에서 사용
  - 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
  - 테이블에 대한 읽기 전용 권한 부여
  - GetTables 권한으로 테이블 필터링
  - 테이블 및 모든 파티션에 전체 액세스 권한 부여
  - 이름 접두사 및 명시적 거부로 액세스 제어
  - 태그를 사용한 액세스 권한 부여
  - 태그를 사용한 액세스 거부
  - 목록 및 배치 API 작업과 함께 태그 사용
  - 조건 키 또는 컨텍스트 키를 사용하여 설정 제어
    - 조건 키를 사용하여 설정을 제어하는 정책 제어
    - 컨텍스트 키를 사용하여 설정을 제어하는 정책 제어
  - ID가 데이터 미리 보기 세션을 생성하지 못하게 하기
- AWS Glue용 리소스 기반 정책 예제
  - AWS Glue에서 리소스 기반 정책 사용 시 고려 사항
  - 리소스 정책을 사용하여 동일한 계정에서 액세스 제어

AWS Glue 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 AWS Glue 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWS API를 사용해 태스크를 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 AWS Glue에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 AWS Glue에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

> **Note**
>
> 이 섹션에 제공된 예제는 모두 `us-west-2` 리전을 사용합니다. 이 리전을 사용하려는 AWS 리전으로 바꿀 수 있습니다.

주제

- 정책 모범 사례
- 리소스 수준 권한은 특정 AWS Glue 객체에만 적용됨
- AWS Glue Console에서 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 테이블에 대한 읽기 전용 권한 부여
- GetTables 권한으로 테이블 필터링
- 테이블 및 모든 파티션에 전체 액세스 권한 부여
- 이름 접두사 및 명시적 거부로 액세스 제어
- 태그를 사용한 액세스 권한 부여
- 태그를 사용한 액세스 거부
- 목록 및 배치 API 작업과 함께 태그 사용
- 조건 키 또는 컨텍스트 키를 사용하여 설정 제어
- ID가 데이터 미리 보기 세션을 생성하지 못하게 하기

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 AWS Glue 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS CloudFormation와 같이, 특정 AWS 서비스를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 – AWS 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우, 추가 보안을 위해 MFA를 설정합니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

리소스 수준 권한은 특정 AWS Glue 객체에만 적용됨

AWS Glue에서 특정 객체에 대한 세분화된 제어만 정의할 수 있습니다. 따라서 `Resource` 문에 대한 Amazon 리소스 이름(ARN)을 허용하는 API 작업이 ARN을 허용하지 않는 API 작업과 혼합되지 않도록 클라이언트의 IAM 정책을 작성해야 합니다.

예를 들어, 다음 IAM 정책은 `GetClassifier` 및 `GetJobRun`에 대한 API 작업을 허용합니다. AWS Glue는 식별자와 작업 실행에 ARN을 허용하지 않기 때문에 이 정책은 `Resource`를 *로 정의합니다. ARN은 `GetDatabase` 및 `GetTable` 같은 특정 API 작업에 대해 허용이 되기 때문에 정책의 후반부에서 ARN을 지정할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetClassifier*",
                "glue:GetJobRun*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:Get*"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:123456789012:catalog",
                "arn:aws:glue:us-east-1:123456789012:database/default",
                "arn:aws:glue:us-east-1:123456789012:table/default/e*1*",
                "arn:aws:glue:us-east-1:123456789012:connection/connection2"
            ]
        }
    ]
}
```

ARN을 허용하는 AWS Glue 객체 목록은 AWS Glue 리소스 ARN 지정 단원을 참조하십시오.

AWS Glue Console에서 사용

AWS Glue Console에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은 AWS 계정에서 AWS Glue 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 직접 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다. 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 AWS Glue Console을 여전히 사용할 수 있도록 하려면 AWS Glue *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책을 엔터티에 추가합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 AWS Glue 콘솔로 작업하려면 해당 사용자는 AWS 계정에 대한 AWS Glue 리소스로 작업하도록 허용하는 최소 권한 집합이 있어야 합니다. 이 AWS Glue 권한 이외에 콘솔은 다음 서비스로부터 권한을 필요로 합니다.

- 로그를 표시할 수 있는 Amazon CloudWatch Logs 권한.
- 역할을 나열하고 전달할 수 있는 AWS Identity and Access Management(IAM) 권한.
- 스택으로 작업할 수 있는 AWS CloudFormation 권한.
- VPC, 서브넷, 보안 그룹, 인스턴스 및 기타 객체를 나열할 수 있는 Amazon Elastic Compute Cloud(Amazon EC2) 권한.
- 버킷과 객체를 나열하고 스크립트를 검색하고 저장할 수 있는 Amazon Simple Storage Service(Amazon S3) 권한.
- 클러스터 작업을 위한 Amazon Redshift 권한.
- 인스턴스를 나열할 수 있는 Amazon Relational Database Service(Amazon RDS) 권한.

사용자가 AWS Glue 콘솔을 보고 작업하는 데 필요한 권한에 대한 자세한 내용은 3단계: AWS Glue에 액세스하는 사용자 또는 그룹에 정책 연결 단원을 참조하세요.

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔은 해당 IAM 정책에 연결된 사용자에 대해 의도대로 작동하지 않습니다. 이 사용자가 AWS Glue Console을 사용할 수 있도록 하려면 `AWSGlueConsoleFullAccess` 관리형 정책을 사용자에게 연결합니다(AWS Glue에 대한 AWS 관리형(미리 정의된) 정책 참조).

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로 이 태스크를 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

테이블에 대한 읽기 전용 권한 부여

다음 정책은 데이터베이스 db1에 있는 테이블 books에 대한 읽기 전용 권한을 부여합니다. Amazon 리소스 이름(ARN)에 대한 자세한 내용은 Data Catalog ARN 단원을 참조하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetTablesActionOnBooks",
      "Effect": "Allow",
      "Action": [
        "glue:GetTables",
        "glue:GetTable"
      ],
      "Resource": [
```

```
        "arn:aws:glue:us-west-2:123456789012:catalog",
        "arn:aws:glue:us-west-2:123456789012:database/db1",
        "arn:aws:glue:us-west-2:123456789012:table/db1/books"
      ]
    }
  ]
}
```

이 정책은 데이터베이스 db1에 있는 테이블 books에 읽기 전용 권한을 부여합니다. 테이블에 Get 권한을 부여하려면 카탈로그 및 데이터베이스 리소스에 대한 권한도 필요합니다.

다음 정책은 데이터베이스 db1에서 테이블 tb1을 생성하기 위해 필요한 최소 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:123456789012:table/db1/tbl1",
        "arn:aws:glue:us-west-2:123456789012:database/db1",
        "arn:aws:glue:us-west-2:123456789012:catalog"
      ]
    }
  ]
}
```

GetTables 권한으로 테이블 필터링

데이터베이스 db1에 customers, stores 및 store_sales와 같은 3개의 테이블이 있다고 가정합니다. 다음 정책은 customers는 제외하고 stores 및 store_sales에 대한 GetTables 권한을 부여합니다. 이 정책으로 GetTables을 호출하면 결과에는 인증된 테이블 두 개만 포함됩니다. customers 테이블은 반환되지 않습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesExample",
```

```
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/store_sales",
                "arn:aws:glue:us-west-2:123456789012:table/db1/stores"
            ]
        }
    ]
}
```

store로 시작하는 모든 테이블 이름과 일치하도록 store*를 사용하여 이전 정책을 간소화할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesExample2",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/store*"
            ]
        }
    ]
}
```

마찬가지로, db1의 모든 테이블과 일치하도록 /db1/*을 사용하면 다음 정책은 db1의 모든 테이블에 대한 GetTables 액세스 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "GetTablesReturnAll",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/*"
            ]
        }
    ]
}
```

테이블 ARN을 제공하지 않으면 GetTables 호출은 성공하지만 빈 목록이 반환됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesEmptyResults",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1"
            ]
        }
    ]
}
```

데이터베이스 ARN이 정책에 누락된 경우에는 GetTables 호출이 실패하고 AccessDeniedException이 발생합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesAccessDeny",
            "Effect": "Allow",
            "Action": [
```

```
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:table/db1/*"
            ]
        }
    ]
}
```

테이블 및 모든 파티션에 전체 액세스 권한 부여

다음 정책은 데이터베이스 db1에 있는 books 테이블에 대한 모든 권한을 부여합니다. 여기에는 테이블 자체, 테이블의 보관된 버전 및 테이블의 모든 파티션에 대한 읽기 및 쓰기 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessOnTable",
            "Effect": "Allow",
            "Action": [
                "glue:CreateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:GetTableVersion",
                "glue:GetTableVersions",
                "glue:DeleteTableVersion",
                "glue:BatchDeleteTableVersion",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:UpdatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                            "arn:aws:glue:us-west-2:123456789012:catalog",
```

```
                        "arn:aws:glue:us-west-2:123456789012:database/db1",
                        "arn:aws:glue:us-west-2:123456789012:table/db1/books"
                        ]
        }
    ]
}
```

실제로 이전 정책은 다음과 같이 간소화할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessOnTable",
            "Effect": "Allow",
            "Action": [
                "glue:*Table*",
                "glue:*Partition*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/books"
            ]
        }
    ]
}
```

세분화된 액세스 제어의 최소 세부 수준은 테이블 수준입니다. 즉, 사용자에게 테이블의 파티션 중 일부 또는 테이블 열 중 일부에 대한 권한을 부여할 수는 없습니다. 사용자는 테이블 전체에 대한 액세스 권한을 갖거나 해당 테이블에 대한 권한을 갖지 못하거나 둘 중 하나입니다.

이름 접두사 및 명시적 거부로 액세스 제어

이 예제에서는 AWS Glue Data Catalog의 데이터베이스 및 테이블이 이름 접두사를 사용하여 구성되었다고 가정합니다. 개발 스테이지의 데이터 베이스에 이름 접두사 `dev-`가 있고, 프로덕션 스테이지의 데이터베이스에는 이름 접두사 `prod-`가 있습니다. 다음 정책을 사용하여 이름 접두사가 `dev-`인 모든 데이터베이스, 테이블, UDF 등에 대한 모든 액세스를 개발자에게 부여할 수 있습니다. 하지만 이름 접두사가 `prod-`인 모든 항목에는 읽기 전용 액세스 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "DevAndProdFullAccess",
            "Effect": "Allow",
            "Action": [
                "glue:*Database*",
                "glue:*Table*",
                "glue:*Partition*",
                "glue:*UserDefinedFunction*",
                "glue:*Connection*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/dev-*",
                "arn:aws:glue:us-west-2:123456789012:database/prod-*",
                "arn:aws:glue:us-west-2:123456789012:table/dev-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/dev-*",
                "arn:aws:glue:us-west-2:123456789012:table/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/dev-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/dev-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:connection/dev-*",
                "arn:aws:glue:us-west-2:123456789012:connection/prod-*"
            ]
        },
        {
            "Sid": "ProdWriteDeny",
            "Effect": "Deny",
            "Action": [
                "glue:*Create*",
                "glue:*Update*",
                "glue:*Delete*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:database/prod-*",
                "arn:aws:glue:us-west-2:123456789012:table/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:connection/prod-*"
            ]
        }
```

```
    ]
}
```

이전 정책의 두 번째 문에서는 명시적 deny를 사용합니다. 명시적 deny를 사용해 보안 주체에 부여된 모든 allow 권한을 덮어쓸 수 있습니다. 그러면 중요한 리소스에 대한 액세스를 잠그고 다른 정책이 이러한 리소스에 대해 실수로 액세스 권한을 부여하지 않도록 방지할 수 있습니다.

이전 예제에서는 첫 번째 문이 prod- 리소스에 대한 모든 액세스 권한을 부여하더라도 두 번째 문이 리소스에 대한 쓰기 액세스 권한을 명시적으로 취소해 prod- 리소스에 대해 일기 전용 액세스만 남아 있게 됩니다.

태그를 사용한 액세스 권한 부여

예를 들어 트리거 t2에 대한 액세스를 계정에 있는 Tom이라는 특정 사용자로 제한하려 한다고 가정합니다. Sam을 포함해 다른 모든 사용자들은 t1을 트리거하기 위한 액세스 권한을 갖습니다. 트리거 t1 및 t2는 다음과 같은 속성을 갖추고 있습니다.

```
aws glue get-triggers
{
    "Triggers": [
        {
            "State": "CREATED",
            "Type": "SCHEDULED",
            "Name": "t1",
            "Actions": [
                {
                    "JobName": "j1"
                }
            ],
            "Schedule": "cron(0 0/1 * * ? *)"
        },
        {
            "State": "CREATED",
            "Type": "SCHEDULED",
            "Name": "t2",
            "Actions": [
                {
                    "JobName": "j1"
                }
            ],
            "Schedule": "cron(0 0/1 * * ? *)"
        }
```

```
    ]
}
```

t2를 트리거하기 위한 태그 값 Tom(aws:ResourceTag/Name": "Tom")에 AWS Glue 관리자가 연결되어 있습니다. 또한 AWS Glue 관리자는 태그를 기준으로 조건 문과 함께 IAM 정책을 Tom에게 제공했습니다. 결과적으로 Tom은 태그 값 Tom으로 리소스에서 수행되는 AWS Glue 작업만 사용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
         "Effect": "Allow",
         "Action": "glue:*",
         "Resource": "*",
         "Condition": {
             "StringEquals": {
                "aws:ResourceTag/Name": "Tom"
              }
          }
         }
    ]
}
```

Tom이 트리거 t1에 대한 액세스를 시도하면 액세스 거부 메시지가 수신됩니다. 한편, Tom은 트리거 t2를 성공적으로 검색할 수 있습니다.

```
aws glue get-trigger --name t1

An error occurred (AccessDeniedException) when calling the GetTrigger operation:
 User: Tom is not authorized to perform: glue:GetTrigger on resource: arn:aws:glue:us-
east-1:123456789012:trigger/t1

aws glue get-trigger --name t2
{
    "Trigger": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j1"
```

```
            }
        ],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}
```

이 작업은 태그에 대한 필터링을 지원하지 않기 때문에 Tom은 여러 개의 `GetTriggers` API 작업을 사용하여 트리거를 나열할 수 없습니다.

Tom에게 `GetTriggers`에 대한 액세스 권한을 부여하기 위해 AWS Glue 관리자는 권한을 두 개의 섹션으로 분할하는 정책을 생성합니다. 첫 번째 섹션에서는 Tom에게 `GetTriggers` API 작업에서 모든 트리거에 액세스할 수 있도록 허용합니다. 두 번째 섹션에서는 Tom에게 `Tom`으로 태그 값이 지정된 API 작업에 액세스할 수 있도록 허용합니다. 이 정책에 따라 Tom에게는 트리거 `t2`에 대한 `GetTriggers` 및 `GetTrigger` 액세스가 모두 허용됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "glue:GetTriggers",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "glue:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Name": "Tom"
                }
            }
        }
    ]
}
```

태그를 사용한 액세스 거부

또 다른 리소스 정책 접근 방식은 리소스에 대한 액세스를 명시적으로 거부하는 것입니다.

> **Important**
>
> 명시적 거부 정책은 GetTriggers와 같은 여러 개의 API 작업에 적용되지 않습니다.

다음 예제 정책에서는 모든 AWS Glue 작업이 허용됩니다. 그러나 두 번째 Effect 명령문은 Team 키와 Special 값으로 태그가 지정된 작업에 대한 액세스를 명시적으로 거부합니다.

관리자가 자격 증명에 다음 정책을 연결하면 해당 자격 증명은 Team 키와 Special 값이 태그로 지정된 작업을 제외한 모든 작업에 액세스할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "glue:*",
      "Resource": "arn:aws:glue:us-east-1:123456789012:job/*"
    },
    {
      "Effect": "Deny",
      "Action": "glue:*",
      "Resource": "arn:aws:glue:us-east-1:123456789012:job/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Team": "Special"
        }
      }
    }
  ]
}
```

목록 및 배치 API 작업과 함께 태그 사용

리소스 정책을 작성하는 세 번째 접근 방식은 태그 값에 대해 리소스를 나열하도록 List API 작업을 사용해 리소스에 대한 액세스를 허용하는 것입니다. 그런 다음, 해당되는 Batch API 작업을 사용하여 특정 리소스의 세부 정보에 대한 액세스를 허용합니다. 이러한 접근 방식에서는 관리자가 여러 개의 GetCrawlers, GetDevEndpoints, GetJobs 또는 GetTriggers API 작업에 대한 액세스를 허용할 필요가 없습니다. 대신에 다음 API 작업을 통해 리소스를 나열할 수 있도록 허용할 수 있습니다.

- ListCrawlers

- ListDevEndpoints
- ListJobs
- ListTriggers

그리고 다음 API 작업을 통해 개별 리소스에 대한 세부 정보를 확인할 수 있도록 허용할 수 있습니다.

- BatchGetCrawlers
- BatchGetDevEndpoints
- BatchGetJobs
- BatchGetTriggers

관리자는 이러한 접근 방식을 사용하기 위해 다음을 수행할 수 있습니다.

1. 크롤러, 개발 엔드포인트, 작업 및 트리거에 태그를 추가합니다.
2. GetCrawlers, GetDevEndponts, GetJobs, GetTriggers 같은 Get API 작업에 대한 사용자 액세스를 거부합니다.
3. 사용자가 액세스 권한을 가진 태그 지정 리소스를 확인할 수 있도록 ListCrawlers, ListDevEndponts, ListJobs, ListTriggers 같은 List API 작업에 대한 사용자 액세스를 허용합니다.
4. TagResource 및 UntagResource 같은 AWS Glue 태그 지정 API에 대한 사용자 액세스를 거부합니다.
5. BatchGetCrawlers, BatchGetDevEndponts, BatchGetJobs, BatchGetTriggers 같은 BatchGet API 작업을 통해 리소스 세부 정보에 대한 사용자 액세스를 허용합니다.

예를 들어, ListCrawlers 작업을 호출할 때 사용자 이름과 일치하도록 태그 값을 제공합니다. 그러면 결과는 제공된 태그 값과 일치하는 크롤러의 목록이 됩니다. 주어진 태그를 사용해 각 크롤러에 대한 세부 정보를 얻을 수 있도록 BatchGetCrawlers에 이름 목록을 제공합니다.

예를 들어 Tom이 Tom으로 태그가 지정된 트리거의 세부 정보만 검색할 수 있도록 하려면 관리자가 Tom에서 트리거에 태그를 추가하고, 모든 사용자에 대해 GetTriggers API 작업 액세스를 거부하고, 모든 사용자에 대해 ListTriggers 및 BatchGetTriggers 액세스를 허용할 수 있습니다.

다음은 AWS Glue 관리자가 Tom에게 부여하는 리소스 정책입니다. 이 정책의 첫 번째 섹션에서는 GetTriggers에 대한 AWS Glue API 작업이 거부됩니다. 정책의 두 번째 섹션에서는 모든 리소스에

대해ListTriggers가 허용됩니다. 그러나 세 번째 섹션에서는 Tom이라고 태그가 지정된 리소스에게 BatchGetTriggers 액세스가 허용됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "glue:GetTriggers",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListTriggers"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:BatchGetTriggers"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Name": "Tom"
                }
            }
        }
    ]
}
```

Tom은 이전 예제와 똑같은 트리거를 사용하여 트리거 t2에 액세스할 수 있지만, 트리거 t1에는 액세스할 수 없습니다. 다음 예제는 Tom이 BatchGetTriggers를 통해 t1 및 t2에 액세스를 시도할 때의 결과를 보여줍니다.

```
aws glue batch-get-triggers --trigger-names t2
```

```
{
    "Triggers": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j2"
            }
        ],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}

aws glue batch-get-triggers  --trigger-names t1

An error occurred (AccessDeniedException) when calling the BatchGetTriggers operation:
 No access to any requested resource.
```

다음 예제는 Tom이 동일한 BatchGetTriggers 호출에서 트리거 t2와 트리거 t3(존재하지 않음) 모두에 액세스를 시도할 때의 결과를 보여줍니다. Tom은 트리거 t2에 대한 액세스 권한을 가지고 있고 이 트리거가 존재하기 때문에 t2만 반환됩니다. Tom에게 트리거 t3에 대한 액세스가 허용되지만 트리거 t3가 존재하지 않기 때문에 "TriggersNotFound": [] 목록에서 응답으로 t3가 반환됩니다.

```
aws glue batch-get-triggers --trigger-names t2 t3
{
    "Triggers": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j2"
            }
        ],
        "TriggersNotFound": ["t3"],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}
```

조건 키 또는 컨텍스트 키를 사용하여 설정 제어

작업 생성 및 업데이트 권한을 부여할 때 조건 키 또는 컨텍스트 키를 사용할 수 있습니다. 다음 섹션에서는 키에 대해 설명합니다.

- 조건 키를 사용하여 설정을 제어하는 정책 제어
- 컨텍스트 키를 사용하여 설정을 제어하는 정책 제어

조건 키를 사용하여 설정을 제어하는 정책 제어

AWS Glue는 `glue:VpcIds`, `glue:SubnetIds` 및 `glue:SecurityGroupIds`의 세 가지 IAM 조건 키를 제공합니다. 작업 생성 및 업데이트 권한을 부여할 때 조건 키를 IAM 정책에 사용할 수 있습니다. 이 설정을 사용하여 원하는 VPC 환경 외부에서 실행하려는 경우 작업 또는 세션을 생성하거나 업데이트하지 않도록 할 수 있습니다. VPC 설정 정보는 `CreateJob` 요청에서 직접 입력하지 않고 AWS Glue 연결을 가리키는 작업 '연결' 필드에서 유추됩니다.

사용 예

원하는 VpcId 'vpc-id1234', SubnetIds, SecurityGroupIds를 사용하여 '트래픽 모니터링 연결'이라는 AWS Glue 네트워크 유형 연결을 생성합니다.

IAM 정책에서 `CreateJob` 및 `UpdateJob` 작업에 대한 조건 키 조건을 지정합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "glue:CreateJob",
    "glue:UpdateJob"
  ],
  "Resource": [
    "*"
  ],
  "Condition": {
    "ForAnyValue:StringLike": {
      "glue:VpcIds": [
        "vpc-id1234"
      ]
    }
  }
}
```

연결 정보를 지정하지 않고 AWS Glue 작업을 생성할 수 없도록 하는 유사한 IAM 정책을 생성할 수 있습니다.

VPC의 세션 제한

지정된 VPC 내에서 생성된 세션을 실행하도록 하려면 glue:vpc-id가 vpc-<123>이 아니라는 조건에서 `glue:CreateSession` 작업에 `Deny` 적용을 추가하여 역할 권한을 제한합니다. 예시:

```
"Effect": "Deny",
"Action": [
    "glue:CreateSession"
 ],
"Condition": {
    "StringNotEquals" : {"glue:VpcIds" : ["vpc-123"]}
}
```

또한 `glue:vpc-id`가 null이라는 조건에서 `glue:CreateSession` 작업에 `Deny` 적용을 추가하여 VPC 내에서 생성된 세션을 실행하도록 할 수도 있습니다. 예시:

```
{
    "Effect": "Deny",
    "Action": [
        "glue:CreateSession"
    ],
      "Condition": {
        "Null": {"glue:VpcIds": true}
    }
},
{
    "Effect": "Allow",
    "Action": [
        "glue:CreateSession"
    ],
    "Resource": ["*"]
}
```

컨텍스트 키를 사용하여 설정을 제어하는 정책 제어

AWS Glue는 작업 및 개발자 엔드포인트에서 AWS Glue를 통해 사용할 수 있는 각 역할 세션에 컨텍스트 키(`glue:CredentialIssuingService= glue.amazonaws.com`)를 제공합

니다. 이렇게 하면 AWS Glue 스크립트에서 수행하는 작업에 대한 보안 제어를 구현할 수 있습니다. AWS Glue는 고객을 대신하여 AWS Glue가 다른 AWS 서비스를 호출(작업/개발 엔드포인트가 아닌 AWS Glue 서비스를 통해 직접 호출)하는 각 역할 세션에 다른 컨텍스트 키(`glue:RoleAssumedBy=glue.amazonaws.com`)를 제공합니다.

사용 예

IAM 정책에서 조건부 권한을 지정하고 AWS Glue 작업에서 사용할 역할에 이 권한을 연결합니다. 이렇게 하면 역할 세션이 AWS Glue 작업 런타임 환경에 사용되는지 여부에 따라 특정 작업이 허용/거부됩니다.

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:CredentialIssuingService": "glue.amazonaws.com"
        }
    }
}
```

ID가 데이터 미리 보기 세션을 생성하지 못하게 하기

이 섹션에는 ID가 데이터 미리 보기 세션을 생성하지 못하게 하는 데 사용되는 IAM 정책 예제가 포함되어 있습니다. 실행 중 데이터 미리 보기 세션에서 사용되는 역할과 별개인 ID에 이 정책을 연결합니다.

```
{
    "Sid": "DatapreviewDeny",
    "Effect": "Deny",
     "Action": [
           "glue:CreateSession"
     ],
     "Resource": [
          "arn:aws:glue:*:*:session/glue-studio-datapreview*"
      ]
 }
```

AWS Glue용 리소스 기반 정책 예제

이 섹션에는 크로스 계정 액세스 권한을 부여하는 정책을 비롯해 리소스 기반 정책 예제가 포함되어 있습니다.

다음 예제에서는 AWS Command Line Interface(AWS CLI)를 사용하여 AWS Glue 서비스 API 작업과 상호 작용합니다. AWS Glue 콘솔에서 또는 AWS SDK 중 하나를 사용하여 동일한 작업을 수행할 수 있습니다.

> **Important**
>
> AWS Glue 리소스 정책을 변경하면 계정 내 기존 AWS Glue 사용자에 대한 권한을 실수로 취소해 예기치 않은 중단을 일으킬 수 있습니다. 다음 예제는 개발 또는 테스트 계정에서만 시도해 보고, 변경하기 전에 이러한 예제가 기존 워크플로우에 문제를 일으키지 않는지 확인하십시오.

주제

- AWS Glue에서 리소스 기반 정책 사용 시 고려 사항
- 리소스 정책을 사용하여 동일한 계정에서 액세스 제어

AWS Glue에서 리소스 기반 정책 사용 시 고려 사항

> **Note**
>
> IAM 정책과 AWS Glue 리소스 정책을 둘 다 전파하려면 몇 초 가량 걸립니다. 새 정책을 연결하더라도 새 정책이 시스템 전체에서 전파되기까지 이전 정책이 계속해서 적용됩니다.

JSON 형식으로 작성된 정책 문서를 사용해 리소스 정책을 생성하거나 수정할 수 있습니다. 정책 구문은 자격 증명 기반 IAM 정책에 대해 동일하지만(IAM JSON 정책 참조 확인) 다음과 같은 예외가 있습니다.

- 각 정책 설명에는 `"Principal"` 또는 `"NotPrincipal"` 블록이 필요합니다.
- `"Principal"` 또는 `"NotPrincipal"`은 유효한 기존 보안 주체를 식별해야 합니다. 와일드카드 패턴(예: `arn:aws:iam::`*`account-id`*`:user/*`)은 허용되지 않습니다.
- 정책의 `"Resource"` 블록에는 다음 정규식 구문과 일치하는 모든 리소스 ARN이 필요합니다. 여기서 첫 번째 `%s`는 *`region`*이고 두 번째 `%s`는 *`account-id`*입니다.

```
*arn:aws:glue:%s:%s:(\*|[a-zA-Z\*]+\/?.*)
```

예를 들어, arn:aws:glue:us-west-2:*account-id*:*와 arn:aws:glue:us-west-2:*account-id*:database/default 둘 다 허용되지만 *는 허용되지 않습니다.

- 자격 증명 기반 정책과 달리 AWS Glue 리소스 정책에는 정책이 연결된 카탈로그에 속한 리소스의 Amazon 리소스 이름(ARN)만 포함되어야 합니다. 이러한 ARN은 항상 arn:aws:glue:로 시작합니다.
- 정책은 자격 증명이 추가 정책 생성 또는 수정을 수행할 수 없도록 잠기게 만들 수는 없습니다.
- 리소스 정책 JSON 문서는 크기가 10KB를 초과할 수 없습니다.

리소스 정책을 사용하여 동일한 계정에서 액세스 제어

이 예제에서 계정 A의 관리 사용자는 계정 A의 IAM 사용자인 Alice에게 카탈로그에 대한 모든 액세스 권한을 부여하는 리소스 정책을 생성합니다. Alice에게는 연결된 IAM 정책이 없습니다.

이렇게 하기 위해 관리 사용자가 다음 AWS CLI 명령을 실행합니다.

```
# Run as admin of Account A
$ aws glue put-resource-policy --profile administrator-name --region us-west-2 --
policy-in-json '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": [
          "arn:aws:iam::account-A-id:user/Alice"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:*"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:account-A-id:*"
      ]
    }
  ]
}'
```

JSON 정책 문서를 AWS CLI 명령의 일부로 입력하는 대신 정책 문서를 파일로 저장한 다음 AWS CLI 명령에서 파일 경로(`file://`로 접두사가 지정됨)를 참조할 수 있습니다. 다음은 이렇게 할 수 있는 방법을 보여주는 예제입니다.

```
$ echo '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": [
          "arn:aws:iam::account-A-id:user/Alice"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:*"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:account-A-id:*"
      ]
    }
  ]
}' > /temp/policy.json

$ aws glue put-resource-policy --profile admin1 \
    --region us-west-2 --policy-in-json file:///temp/policy.json
```

리소스 정책이 전파된 후 Alice는 다음과 같이 계정 A에 있는 모든 AWS Glue 리소스에 액세스할 수 있습니다.

```
# Run as user Alice
$ aws glue create-database --profile alice --region us-west-2 --database-input '{
    "Name": "new_database",
    "Description": "A new database created by Alice",
    "LocationUri": "s3://amzn-s3-demo-bucket"
}'

$ aws glue get-table --profile alice --region us-west-2 --database-name "default" --
table-name "tbl1"}
```

Alice의 `get-table` 호출에 대한 응답으로 AWS Glue 서비스는 다음을 반환합니다.

```
{
  "Table": {
    "Name": "tbl1",
    "PartitionKeys": [],
    "StorageDescriptor": {
        ......
    },
    ......
  }
}
```

AWS Glue에 대한 AWS 관리형 정책 부여

AWS 관리형 정책은 AWS에 의해 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. 만약 AWS가 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 위탁자 ID(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새 AWS 서비스을(를) 시작하거나 새 API 작업을 기존 서비스에 이용하는 경우, AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS Glue에 대한 AWS 관리형(미리 정의된) 정책

AWS는 AWS에서 생성하고 관리하는 독립형 IAM 정책을 제공하여 많은 일반 사용 사례를 처리합니다. 이러한 AWS 관리형 정책은 사용자가 필요한 권한을 조사할 필요가 없도록 일반 사용 사례에 필요한 권한을 부여합니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

계정의 자격 증명에 연결할 수 있는 다음 AWS 관리형 정책은 AWS Glue에 고유하며, 사용 사례 시나리오를 기준으로 그룹화되어 있습니다.

- AWSGlueConsoleFullAccess - 정책이 연결된 자격 증명이 AWS Management Console을 사용하는 경우 AWS Glue 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 보통 AWS Glue 콘솔의 사용자에게 해당됩니다.

- AWSGlueServiceRole – 다양한 AWS Glue 프로세스를 대신 실행하는 데 필요한 리소스에 대한 액세스 권한을 부여합니다. 이러한 리소스에는 AWS Glue, Amazon S3, IAM, CloudWatch Logs 및 Amazon EC2가 포함됩니다. 이 정책에 지정된 리소스의 이름 변환을 따르고자 한다면 AWS Glue 절차는 필요한 권한을 소유합니다. 이 정책은 크롤러, 작업 및 개발 엔드포인트를 정의할 때 지정된 역할에 일반적으로 추가됩니다.
- AwsGlueSessionUserRestrictedServiceRole – 세션을 제외한 모든 AWS Glue 리소스에 대한 전체 액세스 권한을 제공합니다. 사용자와 연결된 대화형 세션만 사용자가 생성하고 사용할 수 있도록 허용합니다. 이 정책에는 다른 AWS 서비스에서 AWS Glue 리소스를 관리하는 데 AWS Glue에서 필요한 기타 권한이 포함됩니다. 또한 이 정책은 다른 AWS 서비스의 AWS Glue 리소스에 태그를 추가할 수 있도록 허용합니다.

  > **Note**
  >
  > 전체 보안 이점을 얻으려면 `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` 또는 `AWSGlueConsoleSageMakerNotebookFullAccess` 정책이 할당된 사용자에게 이 정책을 부여하지 않습니다.

- AwsGlueSessionUserRestrictedPolicy – 태그 키 'owner' 및 담당자의 AWS 사용자 ID와 일치하는 값이 제공된 경우에만 `CreateSession` API 작업을 사용하여 AWS Glue 대화형 세션을 생성할 수 있는 액세스 권한을 제공합니다. 이 자격 증명 정책은 `CreateSession` API 작업을 호출하는 IAM 사용자에게 연결됩니다. 또한 이 정책은 'owner' 태그 및 AWS 사용자 ID와 일치하는 값으로 생성된 AWS Glue 대화형 세션 리소스를 담당자가 조작할 수 있도록 허용합니다. 이 정책은 세션이 생성된 후 AWS Glue 세션 리소스에서 'owner' 태그를 변경하거나 제거할 수 있는 권한을 거부합니다.

  > **Note**
  >
  > 전체 보안 이점을 얻으려면 `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` 또는 `AWSGlueConsoleSageMakerNotebookFullAccess` 정책이 할당된 사용자에게 이 정책을 부여하지 않습니다.

- AwsGlueSessionUserRestrictedNotebookServiceRole - AWS Glue Studio 노트북 세션에 대해 충분한 액세스 권한을 제공하여 특정 AWS Glue 대화형 세션 리소스와 상호 작용합니다. 노트북을 생성하는 보안 주체(IAM 사용자 또는 역할)의 AWS 사용자 ID와 일치하는 'owner' 태그 값으로 생성된 리소스입니다. 이러한 태그에 대한 자세한 내용은 IAM 사용 설명서의 보안 주체 키 값 차트를 참조하세요.

이 서비스-역할 정책은 노트북 내에서 매직 명령문으로 지정되었거나 `CreateSession` API 작업에 역할로 전달된 역할에 연결됩니다. 또한 이 정책은 태그 키 'owner' 및 보안 주체의 AWS 사용자 ID와 일치하는 값이 제공된 경우에만 보안 주체가 AWS Glue Studio 노트북 인터페이스에서 AWS Glue 대화형 세션을 생성할 수 있도록 허용합니다. 이 정책은 세션이 생성된 후 AWS Glue 세션 리소스에서 'owner' 태그를 변경하거나 제거할 수 있는 권한을 거부합니다. 이 정책에는 Amazon S3 버킷에서 읽고 쓰기, CloudWatch 로그 쓰기, AWS Glue에서 사용되는 Amazon EC2 리소스에 대한 태그 생성 및 삭제 권한도 포함됩니다.

> **Note**
>
> 전체 보안 이점을 얻으려면 `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` 또는 `AWSGlueConsoleSageMakerNotebookFullAccess` 정책이 할당된 역할에 이 정책을 부여하지 않습니다.

- AwsGlueSessionUserRestrictedNotebookPolicy – 태그 키 'owner' 및 노트북을 생성하는 보안 주체(IAM 사용자 또는 역할)의 AWS 사용자 ID와 일치하는 값이 있는 경우에만 AWS Glue Studio 노트북 인터페이스에서 AWS Glue 대화형 세션을 생성할 수 있는 액세스 권한을 제공합니다. 이러한 태그에 대한 자세한 내용은 IAM 사용 설명서의 보안 주체 키 값 차트를 참조하세요.

  이 정책은 AWS Glue Studio 노트북 인터페이스에서 세션을 생성하는 보안 주체(IAM 사용자 또는 역할)에 연결됩니다. 또한 이 정책은 AWS Glue Studio 노트북에 충분히 액세스하여 특정 AWS Glue 대화형 세션 리소스와 상호 작용할 수 있도록 허용합니다. 이러한 리소스는 보안 주체의 AWS 사용자 ID와 일치하는 'owner' 태그 값으로 생성됩니다. 이 정책은 세션이 생성된 후 AWS Glue 세션 리소스에서 'owner' 태그를 변경하거나 제거할 수 있는 권한을 거부합니다.

- AWSGlueServiceNotebookRole – AWS Glue Studio 노트북에서 시작된 AWS Glue 세션에 액세스 권한을 부여합니다. 이 정책은 모든 세션에 대한 세션 정보를 나열하고 가져올 수 있도록 허용하지만 사용자가 AWS 사용자 ID로 태깅된 세션을 생성하고 사용할 수 있도록만 허용합니다. 이 정책은 AWS ID로 태깅된 AWS Glue 세션 리소스에서 'owner' 태그를 변경하거나 제거할 수 있는 권한을 거부합니다.

  AWS Glue Studio에서 노트북 인터페이스를 사용하여 작업을 생성하는 AWS 사용자에게 이 정책을 할당합니다.

- AWSGlueConsoleSageMakerNotebookFullAccess – 정책이 연결된 자격 증명이 AWS Management Console을 사용할 때 AWS Glue 및 SageMaker AI 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 SageMaker AI 노트북을 관리하는 AWS Glue 콘솔의 사용자에게 연결됩니다.

- AWSGlueSchemaRegistryFullAccess – 정책이 연결된 자격 증명이 AWS Management Console 또는 AWS CLI를 사용할 때 AWS Glue Schema Registry 리소스에 대한 전체 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue Schema Registry를 관리하는 AWS Glue 콘솔 또는 AWS CLI의 사용자에게 연결됩니다.
- AWSGlueSchemaRegistryReadonlyAccess – 정책이 연결된 자격 증명이 AWS Management Console 또는 AWS CLI를 사용할 때 AWS Glue Schema Registry 리소스에 대한 읽기 전용 액세스 권한을 부여합니다. 이 정책에 지정된 리소스의 이름 변환을 따르면 사용자는 콘솔 전체 용량을 소유합니다. 이 정책은 일반적으로 AWS Glue Schema Registry를 사용하는 AWS Glue 콘솔 또는 AWS CLI의 사용자에게 연결됩니다.

> **Note**
>
> IAM 콘솔에 로그인하고 이 콘솔에서 특정 정책을 검색하여 이러한 권한 정책을 검토할 수 있습니다.

AWS Glue 작업 및 리소스에 대한 권한을 허용하는 고유의 사용자 정의 IAM 정책을 생성할 수도 있습니다. 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

사용자 지정 IAM 역할을 사용하는 동안 VPC 구성과의 연결을 생성하려면 다음 VPC 액세스 작업이 있어야 합니다.

- secretsmanager:GetSecretValue
- secretsmanager:PutSecretValue
- secretsmanager:DescribeSecret
- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSubnets

AWS 관리형 정책에 대한 AWS Glue 업데이트

이 서비스가 이러한 변경 내용을 추적하기 시작한 이후부터 AWS Glue의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 AWS Glue 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AwsGlueSessionUserRestrictedNotebookPolicy – 기존 정책에 대한 간단한 업데이트입니다.	소유자 태그 키에 `glue:TagResource` 작업 허용을 추가합니다. 소유자 태그 키가 있는 세션의 생성 시 태깅을 지원하는 데 필요합니다.	2024년 8월 30일
AwsGlueSessionUserRestrictedNotebookServiceRole - 기존 정책에 대한 마이너 업데이트	소유자 태그 키에 `glue:TagResource` 작업 허용을 추가합니다. 소유자 태그 키가 있는 세션의 생성 시 태깅을 지원하는 데 필요합니다.	2024년 8월 30일
AWSGlueServiceServiceServiceRole — 기존 정책에 대한 마이너 업데이트	소유자 태그 키에 `glue:TagResource` 작업 허용을 추가합니다. 소유자 태그 키가 있는 세션의 생성 시 태깅을 지원하는 데 필요합니다.	2024년 8월 5일
AwsGlueSessionUserRestrictedServiceRole — 기존 정책에 대한 마이너 업데이트	소유자 태그 키에 `glue:TagResource` 작업 허용을 추가합니다. 소유자 태그 키가 있는 세션의 생성 시 태깅을 지원하는 데 필요합니다.	2024년 8월 5일
AWSGlueServiceServiceServiceRole — 기존 정책에 대한 마이너 업데이트	`glue:StartCompletion` 및 `glue:GetCompletion` 을 정책에 추가합니다. AWS Glue에서의 Amazon Q 데이터 통합에 필요합니다.	2024년 4월 30일

변경 사항	설명	날짜
AwsGlueSessionUserRestrictedNotebookServiceRole - 기존 정책에 대한 마이너 업데이트	`glue:StartCompletion` 및 `glue:GetCompletion` 을 정책에 추가합니다. AWS Glue에서의 Amazon Q 데이터 통합에 필요합니다.	2024년 4월 30일
AwsGlueSessionUserRestrictedServiceRole — 기존 정책에 대한 마이너 업데이트	`glue:StartCompletion` 및 `glue:GetCompletion` 을 정책에 추가합니다. AWS Glue에서의 Amazon Q 데이터 통합에 필요합니다.	2024년 4월 30일
AWSGlueServiceNotebookRole - 기존 정책에 대한 마이너 업데이트.	`glue:StartCompletion` 및 `glue:GetCompletion` 을 정책에 추가합니다. AWS Glue에서의 Amazon Q 데이터 통합에 필요합니다.	2024년 1월 30일
AwsGlueSessionUserRestrictedNotebookPolicy – 기존 정책에 대한 간단한 업데이트입니다.	`glue:StartCompletion` 및 `glue:GetCompletion` 을 정책에 추가합니다. AWS Glue에서의 Amazon Q 데이터 통합에 필요합니다.	2023년 11월 29일
AWSGlueServiceNotebookRole - 기존 정책에 대한 마이너 업데이트.	`codewhisperer:GenerateRecommendations` 를 정책에 추가합니다. AWS Glue가 CodeWhisperer 권장 사항을 생성하는 새 기능에 필요합니다.	2023년 10월 9일
AWSGlueServiceRole – 기존 정책에 대한 마이너 업데이트.	AWS Glue 로깅을 더 잘 반영하도록 CloudWatch 권한 범위를 좁힙니다.	2023년 8월 4일

변경 사항	설명	날짜
AWSGlueConsoleFullAccess – 기존 정책에 대한 마이너 업데이트.	`databrew` 레시피 나열 및 설명 권한을 정책에 추가합니다. AWS Glue에서 레시피에 액세스할 수 있는 경우 새 기능에 대한 전체 관리 액세스 권한을 제공하는 데 필요합니다.	2023년 5월 9일
AWSGlueConsoleFullAccess – 기존 정책에 대한 마이너 업데이트.	`cloudformation:ListStacks` 를 정책에 추가합니다. AWS CloudFormation 인증 요구 사항이 변경된 후에도 기존 기능을 유지합니다.	2023년 3월 28일
대화형 세션 기능에 대해 추가된 새로운 관리형 정책: • AwsGlueSessionUserRestrictedServiceRole • AwsGlueSessionUserRestrictedPolicy • AwsGlueSessionUserRestrictedNotebookServiceRole • AwsGlueSessionUserRestrictedNotebookPolicy	이러한 정책은 AWS Glue Studio에서 대화형 세션과 노트북에 대한 추가 보안을 제공하도록 설계되었습니다. 소유자만 액세스할 수 있도록 정책에서 `CreateSession` API 작업에 대한 액세스를 제한합니다.	2021년 11월 30일

변경 사항	설명	날짜
AWSGlueConsoleSageMakerNotebookFullAccess – 기존 정책 업데이트	AWS Glue가 스크립트 및 임시 파일을 저장하는 데 사용하는 Amazon S3 버킷에 대한 읽기/쓰기 권한을 부여하는 작업에 대한 중복 리소스 ARN(`arn:aws:s3:::aws-glue-*/*`)을 제거했습니다. `"StringEquals"` 를 `"ForAnyValue:StringLike"` 로 변경하여 구문 문제를 수정하고, 순서가 어긋난 각 위치에서 `"Effect": "Allow"` 줄을 `"Action":` 줄 앞으로 이동했습니다.	2021년 7월 15일
AWSGlueConsoleFullAccess – 기존 정책 업데이트	AWS Glue가 스크립트 및 임시 파일을 저장하는 데 사용하는 Amazon S3 버킷에 대한 읽기/쓰기 권한을 부여하는 작업에 대한 중복 리소스 ARN(`arn:aws:s3:::aws-glue-*/*`)을 제거했습니다.	2021년 7월 15일
AWS Glue에서 변경 사항 추적 시작	AWS Glue에서 AWS 관리형 정책에 대한 변경 내용 추적을 시작했습니다.	2021년 6월 10일

AWS Glue 리소스 ARN 지정

AWS Glue에서 AWS Identity and Access Management(IAM) 정책을 사용하여 리소스에 대한 액세스를 제어할 수 있습니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다. AWS Glue의 일부 리소스는 ARN을 지원하지 않습니다.

주제

- Data Catalog ARN
- AWS Glue에서 카탈로그 이외 객체의 ARN
- AWS Glue 카탈로그 이외 단수 API 작업에 대한 액세스 제어
- 여러 항목을 검색하는 AWS Glue 카탈로그 이외 API 직업에 대한 액세스 제어
- AWS Glue 카탈로그 이외 BatchGet API 작업에 대한 액세스 제어

Data Catalog ARN

Data Catalog 리소스에는 `catalog`가 루트인 계층 구조가 있습니다.

```
arn:aws:glue:region:account-id:catalog
```

각 AWS 계정에는 12자리 계정 ID가 카탈로그 ID인 Data Catalog가 AWS 리전에 하나씩 있습니다. 다음 표와 같이 리소스에는 고유한 ARN이 연결되어 있습니다.

리소스 유형	ARN 형식
카탈로그	`arn:aws:glue:` *`region`*`:`*`account-id`* `:catalog` 예: `arn:aws:glue:us-east-1:123456789012:catalog`
데이터베이스	`arn:aws:glue:` *`region`*`:`*`account-id`* `:database/` *`database name`* 예: `arn:aws:glue:us-east-1:123456789012:database/db1`
표	`arn:aws:glue:` *`region`*`:`*`account-id`* `:table/`*`database name`*`/`*`table name`* 예: `arn:aws:glue:us-east-1:123456789012:table/db1/tbl1`
페더레이션 S3 테이블 카탈로그(모든 테이블 버킷)	`arn:aws:glue:` *`region`*`:`*`account-id`* `:catalog/s3tablescatalog` 예: `arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog`

리소스 유형	ARN 형식
페더레이션 S3 테이블 버킷 카탈로그(하위 카탈로그)	arn:aws:glue: *region*:*account-id* :catalog/s3tablescatalog/ *bucket name* 예: arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog/amzn-s3-demo-bucket1
페더레이션 S3 테이블 데이터베이스	arn:aws:glue: *region*:*account-id* :database/s3tablescatalog/ *child catalog name*/*database name* 예: arn:aws:glue:us-east-1:123456789012:database/s3tablescatalog/amzn-s3-demo-bucket1/nsdb1
페더레이션 S3 테이블	arn:aws:glue: *region*:*account-id* :table/s3tablescatalog/ *child catalog name*/*database name*/*table name* 예: arn:aws:glue:us-east-1:123456789012:table/s3tablescatalog/amzn-s3-demo-bucket1/nsdb1/s3tbl1
페더레이션 S3 테이블 카탈로그(Lake Formation에 등록된 단일 테이블 버킷)	arn:aws:glue: *region*:*account-id* :catalog/#### ## 예: arn:aws:glue:us-east-1:123456789012:catalog/amzn-s3-demo-bucket1
페더레이션 S3 테이블 데이터베이스	arn:aws:glue: *region*:*account-id* :catalog/#### ##/###### ## 예: arn:aws:glue:us-east-1:123456789012:database/amzn-s3-demo-bucket1/nsdb1
페더레이션 S3 테이블	arn:aws:glue: *region*:*account-id* :catalog/#### ##/###### ##/### ## 예: arn:aws:glue:us-east-1:123456789012:table/amzn-s3-demo-bucket1/ *nsdb1*/s3tbl1

리소스 유형	ARN 형식
페더레이션 또는 관리형 카탈로그(다중 카탈로그의 최상위 카탈로그)	arn:aws:glue: *region*:*account-id* :catalog/ *top-level catalog name* 예: arn:aws:glue:us-east-1:123456789012:catalog/nscatalog 관리형 카탈로그의 ARN 형식은 동일한 구조를 따릅니다.
페더레이션 다중 레벨 카탈로그(다중 레벨 카탈로그의 하위 카탈로그)	arn:aws:glue: *region*:*account-id* :catalog/ *top-level catalog name*/*child catalog name* 예: arn:aws:glue:us-east-1:123456789012:catalog/nscatalog/dbcatalog
페더레이션 데이터베이스	arn:aws:glue: *region*:*account-id* :database/ *name space catalog name*/*child catalog name*/*database name* 예: arn:aws:glue:us-east-1:123456789012:database/nscatalog/dbcatalog/schemadb
페더레이션 테이블	arn:aws:glue: *region*:*account-id* :table/*name space catalog name*/*child catalog name*/*database name*/*table name* 예: arn:aws:glue:us-east-1:123456789012:table/nscatalog/dbcatalog/schemadb/rstbl1
카탈로그 링크 컨테이너	arn:aws:glue: *region*:*account-id* :catalog/*## #### ##* 예: arn:aws:glue:glue:us-east-1:123456789012:catalog /linkcontainer-example
데이터베이스	arn:aws:glue: *region*:*account-id* :catalog/*## #### ##*/*### ### ##* 예: arn:aws:glue:glue:us-east-1:123456789012:database /linkcontainer-example/link-db

리소스 유형	ARN 형식
사용자 정의 함수	arn:aws:glue: *region*:*account-id* :userDefinedFunction/ *database name*/*user-defined function name* 예: arn:aws:glue:us-east-1:123456789012:userDefinedFunction/db1/func1
연결	arn:aws:glue: *region*:*account-id* :connection/ *connection name* 예: arn:aws:glue:us-east-1:123456789012:connection/connection1
대화형 세션	arn:aws:glue: *region*:*account-id* :session/ *interactive session id* 예: arn:aws:glue:us-east-1:123456789012:session/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

세분화된 액세스 제어를 사용하려면 IAM 정책 및 리소스 정책에서 이러한 ARN을 사용하여 특정 리소스에 대한 액세스 권한을 부여 및 거부할 수 있습니다. 와일드카드가 정책에서 허용됩니다. 예를 들어, 다음 ARN은 데이터베이스 default의 모든 테이블과 일치합니다.

```
arn:aws:glue:us-east-1:123456789012:table/default/*
```

Important

Data Catalog 리소스에 대해 수행된 모든 작업에는 리소스와 해당 리소스의 모든 상위 항목에 대한 권한이 필요합니다. 예를 들어, 테이블에 대한 파티션을 생성하려면 해당 테이블, 데이터베이스 및 이 테이블이 있는 카탈로그에 대한 권한이 필요합니다. 다음 예에서는 Data Catalog에서 데이터베이스 PrivateDatabase의 테이블 PrivateTable에 대한 파티션을 생성하는 데 필요한 권한을 보여줍니다.

```
{
   "Sid": "GrantCreatePartitions",
   "Effect": "Allow",
   "Action": [
```

```
        "glue:BatchCreatePartitions"
    ],
    "Resource": [
        "arn:aws:glue:us-east-1:123456789012:table/PrivateDatabase/PrivateTable",
        "arn:aws:glue:us-east-1:123456789012:database/PrivateDatabase",
        "arn:aws:glue:us-east-1:123456789012:catalog"
    ]
}
```

리소스와 해당 리소스의 모든 상위 항목에 대한 권한 이외에도 모든 삭제 작업에는 해당 리소스의 모든 하위 항목에 대한 권한이 필요합니다. 예를 들어, 데이터베이스를 삭제하려면 데이터베이스 및 데이터베이스가 위치하고 있는 카탈로그 외에도 데이터베이스에 있는 모든 테이블과 사용자 정의 함수에 대한 권한이 필요합니다. 다음 예에서는 Data Catalog에서 `PrivateDatabase` 데이터베이스를 삭제하는 데 필요한 권한을 보여줍니다.

```
{
    "Sid": "GrantDeleteDatabase",
    "Effect": "Allow",
    "Action": [
        "glue:DeleteDatabase"
    ],
    "Resource": [
        "arn:aws:glue:us-east-1:123456789012:table/PrivateDatabase/*",
        "arn:aws:glue:us-east-1:123456789012:userDefinedFunction/PrivateDatabase/
*",
        "arn:aws:glue:us-east-1:123456789012:database/PrivateDatabase",
        "arn:aws:glue:us-east-1:123456789012:catalog"
    ]
}
```

즉, Data Catalog 리소스에 대한 작업은 다음 권한 규칙을 따릅니다.

- 카탈로그에 대한 작업에는 카탈로그에 대한 권한만 필요합니다.
- 데이터베이스에 대한 작업에는 데이터베이스와 카탈로그에 대한 권한이 필요합니다.
- 데이터베이스에 대한 삭제 작업에는 데이터베이스 및 카탈로그와 해당 데이터베이스 내 모든 테이블과 사용자 정의 함수에 대한 권한이 필요합니다.
- 테이블, 파티션 또는 테이블 버전에 대한 작업에는 테이블, 데이터베이스 및 카탈로그에 대한 권한이 필요합니다.
- 사용자 정의 함수에 대한 작업에는 사용자 정의 함수, 데이터베이스 및 카탈로그에 대한 권한이 필요합니다.

- 연결에 대한 작업에는 연결과 카탈로그에 대한 권한이 필요합니다.

AWS Glue에서 카탈로그 이외 객체의 ARN

일부 AWS Glue 리소스에서는 ARN을 사용하여 액세스를 제어할 수 있도록 리소스 수준 권한을 허용합니다. IAM 정책에서 이러한 ARN을 사용하여 세분화된 액세스 제어를 활성화할 수 있습니다. 다음 표에는 리소스 ARN을 포함할 수 있는 리소스가 나와 있습니다.

리소스 유형	ARN 형식
크롤러	arn:aws:glue: *region*:*account-id* :crawler/ *crawler-name* 예: arn:aws:glue:us-east-1:123456789012:crawler/mycrawler
작업	arn:aws:glue: *region*:*account-id* :job/*job-name* 예: arn:aws:glue:us-east-1:123456789012:job/testjob
트리거	arn:aws:glue: *region*:*account-id* :trigger/ *trigger-name* 예: arn:aws:glue:us-east-1:123456789012:trigger/sampletrigger
개발 엔드포인트	arn:aws:glue: *region*:*account-id* :devEndpoint/ *development-endpoint-name* 예: arn:aws:glue:us-east-1:123456789012:devEndpoint/temporarydevendpoint
기계 학습 변환	arn:aws:glue: *region*:*account-id* :mlTransform/ *transform-id* 예: arn:aws:glue:us-east-1:123456789012:mlTransform/tfm-1234567890

AWS Glue 카탈로그 이외 단수 API 작업에 대한 액세스 제어

AWS Glue 카탈로그 이외 단수 API 작업은 단일 항목(개발 엔드포인트)에 대해 수행됩니다. `GetDevEndpoint`, `CreateUpdateDevEndpoint` 및 `UpdateDevEndpoint`를 예로 들 수 있습니다. 이러한 작업의 경우 정책은 `"action"` 블록에 API 이름을, `"resource"` 블록에 리소스 ARN을 배치해야 합니다.

사용자가 `GetDevEndpoint` 작업을 호출하도록 허용하려 한다고 가정해 보겠습니다. 다음 정책은 `myDevEndpoint-1`이라는 엔드포인트에 필요한 최소 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MinimumPermissions",
            "Effect": "Allow",
            "Action": "glue:GetDevEndpoint",
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/
myDevEndpoint-1"
        }
    ]
}
```

다음 정책은 와일드카드(*)를 사용하여 `myDevEndpoint-`와 일치하는 리소스에 대한 `UpdateDevEndpoint` 액세스를 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionWithWildcard",
            "Effect": "Allow",
            "Action": "glue:UpdateDevEndpoint",
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/myDevEndpoint-
*"
        }
    ]
}
```

다음 예에서처럼 정책 2개를 결합할 수 있습니다. 이름이 A로 시작하는 개발 엔드포인트에 대한 EntityNotFoundException을 볼 수 있습니다. 하지만 다른 개발 엔드포인트에 액세스하려고 시도하면 액세스 거부 오류가 반환됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CombinedPermissions",
            "Effect": "Allow",
            "Action": [
                "glue:UpdateDevEndpoint",
                "glue:GetDevEndpoint"
            ],
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/A*"
        }
    ]
}
```

여러 항목을 검색하는 AWS Glue 카탈로그 이외 API 직업에 대한 액세스 제어

일부 AWS Glue API 작업은 여러 항목(예: 여러 개발 엔드포인트)을 검색합니다(예: GetDevEndpoints). 이 작업의 경우 와일드카드(*) 리소스만 지정할 수 있으며 특정 ARN을 지정할 수 없습니다.

예를 들어 정책에 GetDevEndpoints를 포함하려면 리소스의 범위를 와일드카드(*)로 지정해야 합니다. 이 예에서는 단수 작업(GetDevEndpoint, CreateDevEndpoint 및 DeleteDevendpoint)의 범위 또한 모든(*) 리소스로 지정합니다.

```
{
            "Sid": "PluralAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:GetDevEndpoints",
                "glue:GetDevEndpoint",
                "glue:CreateDevEndpoint",
                "glue:UpdateDevEndpoint"
            ],
            "Resource": [
                "*"
            ]
```

```
}
```

AWS Glue 카탈로그 이외 BatchGet API 작업에 대한 액세스 제어

일부 AWS Glue API 작업은 여러 항목(예: 여러 개발 엔드포인트)을 검색합니다(예: `BatchGetDevEndpoints`). 이 작업에서 액세스할 수 있는 리소스의 범위를 제한하도록 ARN을 지정할 수 있습니다.

예를 들어 특정 개발 엔드포인트에 대한 액세스를 허용하려면 정책에 리소스 ARN과 함께 `BatchGetDevEndpoints`를 포함시킵니다.

```
{
            "Sid": "BatchGetAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:BatchGetDevEndpoints"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:123456789012:devEndpoint/de1"
            ]
}
```

이 정책을 통해 `de1`이라는 개발 엔드포인트에 성공적으로 액세스할 수 있습니다. 그러나 `de2`라는 개발 엔드포인트에 액세스를 시도하면 오류가 반환됩니다.

```
An error occurred (AccessDeniedException) when calling the BatchGetDevEndpoints
 operation: No access to any requested resource.
```

> **Important**
>
> `List` 및 `BatchGet` API 작업 사용과 같이 IAM 정책을 설정하기 위한 대체 접근 방식은 AWS Glue 자격 증명 기반 정책 예제 단원을 참조하십시오.

교차 계정 액세스 권한 부여

계정 간에 데이터 카탈로그 리소스에 대한 액세스 권한을 부여하면 추출, 변환, 로드(ETL) 작업에서 다른 계정의 데이터를 쿼리하고 결합할 수 있습니다.

주제

- AWS Glue에서 교차 계정 액세스 권한을 부여하는 방법
- 데이터 카탈로그 리소스 정책 추가 또는 업데이트
- 교차 계정 API 호출 수행
- 교차 계정 ETL 호출 수행
- 교차 계정 CloudTrail 로깅
- 교차 계정 리소스 소유권 및 결제
- 교차 계정 액세스 제한 사항

AWS Glue에서 교차 계정 액세스 권한을 부여하는 방법

AWS Glue 메서드를 사용하거나 AWS Lake Formation 교차 계정 권한 부여를 사용하여 외부 AWS 계정에 데이터에 대한 액세스 권한을 부여할 수 있습니다. AWS Glue 매서드는 AWS Identity and Access Management(IAM) 정책을 사용하여 세분화된 액세스 제어를 수행합니다. Lake Formation은 관계형 데이터베이스 시스템의 GRANT/REVOKE 명령과 유사한 더 간단한 GRANT/REVOKE 권한 모델을 사용합니다.

이 섹션에서는 AWS Glue 메서드 사용에 대해 설명합니다. Lake Formation 교차 계정 권한 부여 사용에 대한 자세한 내용은 AWS Lake Formation Developer Guide의 Granting Lake Formation Permissions를 참조하세요.

리소스에 대한 교차 계정 액세스 권한을 부여하기 위한 두 가지 AWS Glue 방법이 있습니다.

- 데이터 카탈로그 리소스 정책 사용
- IAM 역할 사용

리소스 정책을 사용하여 교차 계정 액세스 권한 부여

다음은 데이터 카탈로그 리소스 정책을 사용하여 교차 계정 액세스 권한을 부여하는 일반적인 단계입니다.

1. 계정 A의 관리자(또는 기타 권한이 있는 자격 증명)가 계정 A의 데이터 카탈로그에 리소스 정책을 연결합니다. 이 정책은 계정 B에게 계정 A의 카탈로그에 있는 리소스에 대한 작업을 수행할 수 있는 특정 교차 계정 권한을 부여합니다.
2. 계정 B의 관리자는 계정 A에게서 받은 권한을 위임하는 계정 B의 IAM 자격 증명에 IAM 정책을 연결합니다.

계정 B의 자격 증명이 이제 계정 A의 지정된 리소스에 대한 액세스 권한을 갖습니다.

자격 증명은 리소스에 액세스하기 위해 리소스 소유자(계정 A)와 상위 계정(계정 B) 모두의 권한이 필요합니다.

IAM 역할을 사용하여 교차 계정 액세스 권한 부여

다음은 IAM 역할을 사용하여 교차 계정 액세스 권한을 부여하는 일반적인 단계입니다.

1. 리소스를 소유한 계정(계정 A)의 관리자(또는 권한이 있는 다른 자격 증명)가 IAM 역할을 생성합니다.
2. 계정 A의 관리자는 문제의 리소스에 대한 교차 계정 액세스 권한을 부여하는 역할에 정책을 연결합니다.
3. 계정 A의 관리자는 다른 계정(계정 B)의 IAM 자격 증명을 해당 역할을 맡을 수 있는 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.

   AWS 서비스에 역할을 맡을 수 있는 권한을 부여하고 싶다면, 신뢰 정책의 보안 주체는 AWS 서비스 보안 주체가 됩니다.
4. 이제, 계정 B의 관리자가 계정 B에 있는 하나 이상의 IAM 자격 증명에 권한을 위임해 해당 자격 증명이 역할을 맡을 수 있습니다. 그러면 계정 B의 해당 자격 증명에 계정 A의 리소스에 대한 액세스 권한이 부여됩니다.

IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리 단원을 참조하십시오. 사용자, 그룹, 역할 및 권한에 대한 자세한 내용은 IAM User Guide의 Identities (users, groups, and roles)를 참조하세요.

두 가지 방법을 비교하려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이점을 참조하세요. AWS Glue에서는 두 옵션을 모두 지원하지만, 리소스 정책은 데이터 카탈로그 리소스에 대한 액세스 권한만 부여할 수 있다는 제한 사항이 있습니다.

예를 들어, 계정 B의 Dev 역할에게 계정 A의 데이터베이스 db1에 대한 액세스 권한을 부여하려면 다음 리소스 정책을 계정 A의 카탈로그에 연결합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase"
      ],
      "Principal": {"AWS": [
        "arn:aws:iam::account-B-id:role/Dev"
      ]},
      "Resource": [
        "arn:aws:glue:us-east-1:account-A-id:catalog",
        "arn:aws:glue:us-east-1:account-A-id:database/db1"
      ]
    }
  ]
}
```

또한 계정 B는 계정 A의 db1에 액세스하기 전에 다음 IAM 정책을 Dev 역할에 연결해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:account-A-id:catalog",
        "arn:aws:glue:us-east-1:account-A-id:database/db1"
      ]
    }
  ]
}
```

데이터 카탈로그 리소스 정책 추가 또는 업데이트

콘솔, API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 AWS Glue 데이터 카탈로그 리소스 정책을 추가하거나 업데이트할 수 있습니다.

> **Important**
>
> AWS Lake Formation이 있는 계정에서 이미 교차 계정 권한을 부여한 경우 데이터 카탈로그 리소스 정책을 추가하거나 업데이트하려면 추가 단계가 필요합니다. 자세한 내용은 AWS Lake

Formation 개발자 가이드의 AWS Glue 및 Lake Formation을 모두 사용하여 계정 권한 관리를 참조하세요.
Lake Formation 크로스 계정 권한 부여가 있는지 확인하려면 `glue:GetResourcePolicies` API 작업 또는 AWS CLI를 사용합니다. `glue:GetResourcePolicies`이(가) 이미 존재하는 데이터 카탈로그 정책 이외의 정책을 반환하는 경우 Lake Formation 권한 부여가 존재합니다. 자세한 내용은 AWS Lake Formation 개발자 가이드의 GetResourcePolicies API 작업을 사용하여 크로스 계정 권한 부여 보기를 참조하세요.

데이터 카탈로그 리소스 정책을 추가하거나 업데이트하려면(콘솔)

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

   `glue:PutResourcePolicy` 권한이 있는 AWS Identity and Access Management(IAM) 관리 사용자로 로그인합니다.
2. 탐색 창에서 설정을 선택합니다.
3. [데이터 카탈로그 설정(Data catalog settings)] 페이지의 [권한(Permissions)]에서 리소스 정책을 텍스트 영역에 붙여넣습니다. 그런 다음 저장을 선택합니다.

   정책의 권한이 Lake Formation을 사용하여 부여된 권한에 추가된다는 알림이 콘솔에 표시되면 [계속(Proceed)]을 선택합니다.

데이터 카탈로그 리소스 정책을 추가하거나 업데이트하려면(AWS CLI)

- `aws glue put-resource-policy` 명령을 제출합니다. Lake Formation 권한 부여가 이미 있는 경우 `--enable-hybrid` 옵션을 `'TRUE'` 값과 함께 포함해야 합니다.

  이 명령의 사용 예는 AWS Glue용 리소스 기반 정책 예제 섹션을 참조하세요.

교차 계정 API 호출 수행

모든 AWS Glue Data Catalog 작업에는 `CatalogId` 필드가 있습니다. 교차 계정 액세스를 활성화하기 위한 필수 권한이 부여된 경우 호출자는 계정을 교차하여 데이터 카탈로그 API 호출을 수행할 수 있습니다. 호출자는 해당 대상 계정의 리소스에 액세스하기 위해 `CatalogId`의 대상 AWS 계정 ID를 전달하여 이렇게 합니다.

`CatalogId` 값을 입력하지 않으면 AWS Glue에서는 기본적으로 호출자의 계정 ID를 사용하고 교차 계정 호출이 수행되지 않습니다.

교차 계정 ETL 호출 수행

일부 AWS Glue PySpark 및 Scala API에는 카탈로그 ID 필드가 있습니다. 교차 계정 액세스가 가능하도록 모든 필수 권한이 부여된 경우 ETL 작업은 대상 계정의 데이터 카탈로그 리소스에 액세스할 수 있도록 카탈로그 ID 필드에 대상 AWS 계정 ID를 전달하여 계정 간에 API 작업에 대해 PySpark 및 Scala를 호출할 수 있습니다.

카탈로그 ID 값을 입력하지 않으면 AWS Glue에서는 기본적으로 호출자의 계정 ID를 사용하고 교차 계정 호출이 수행되지 않습니다.

`catalog_id`를 지원하는 PySpark API는 GlueContext 클래스 단원을 참조하십시오. `catalogId`를 지원하는 Scala API는 AWS Glue Scala GlueContext API 단원을 참조하십시오.

다음 예는 ETL 작업을 실행하기 위해 피부여자에게 필요한 권한을 보여줍니다. 이 예에서 *`grantee-account-id`*는 작업을 실행하는 클라이언트의 `catalog-id`이고, *`grantor-account-id`*는 리소스의 소유자입니다. 이 예에서는 부여자의 계정에 있는 모든 카탈로그 리소스에 대한 권한을 부여합니다. 부여되는 리소스의 범위를 제한하기 위해 카탈로그, 데이터베이스, 테이블 및 연결에 해당하는 특정 ARN을 제공할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetConnection",
                "glue:GetDatabase",
                "glue:GetTable",
                "glue:GetPartition"
            ],
            "Principal": {"AWS": ["arn:aws:iam::grantee-account-id:root"]},
            "Resource": [
                "arn:aws:glue:us-east-1:grantor-account-id:*"
            ]
        }
    ]
}
```

> **Note**
>
> 부여자의 계정에 있는 테이블이 부여자의 계정에 있는 Amazon S3 위치를 가리키는 경우 피부여자의 계정에서 ETL 작업을 실행하는 데 사용하는 IAM 역할은 부여자의 계정에서 객체를 나열하고 가져올 수 있는 권한이 있어야 합니다.

계정 A의 클라이언트가 ETL 작업을 생성하고 실행할 수 있는 권한을 이미 가지고 있는 경우 교차 계정 액세스를 위해 ETL 작업을 설정하는 기본 단계는 다음과 같습니다.

1. 교차 계정 데이터 액세스를 허용합니다(Amazon S3 교차 계정 액세스가 이미 설정된 경우에는 이 단계를 건너뜀).
   a. 계정 A에서 교차 계정 액세스를 허용하도록 계정 B에서 Amazon S3 버킷 정책을 업데이트합니다.
   b. 계정 B에서 버킷에 대한 액세스를 허용하도록 계정 A에서 IAM 정책을 업데이트합니다.
2. 교차 계정 데이터 카탈로그 액세스를 허용합니다.
   a. 계정 A에서 액세스를 허용하도록 계정 B에서 데이터 카탈로그에 연결된 리소스 정책을 생성하거나 업데이트합니다.
   b. 계정 B에서 데이터 카탈로그에 대한 액세스를 허용하도록 계정 A에서 IAM 정책을 업데이트합니다.

교차 계정 CloudTrail 로깅

AWS Glue 추출, 변환, 로드 작업이 AWS Lake Formation 교차 계정 권한 부여를 통해 공유되는 데이터 카탈로그 테이블의 기본 데이터에 액세스하면 추가 AWS CloudTrail 로깅 동작이 있습니다.

이 설명을 위해 테이블을 공유한 AWS 계정은 소유자 계정이고 테이블을 공유한 계정은 수신자 계정입니다. 수신자 계정의 ETL 작업이 소유자 계정의 테이블에 있는 데이터에 액세스하면 수신자 계정의 로그에 추가된 데이터 액세스 CloudTrail 이벤트가 소유자 계정의 CloudTrail 로그에 복사됩니다. 이는 소유자 계정이 다양한 수신자 계정의 데이터 액세스를 추적할 수 있도록 하기 위한 것입니다. 기본적으로 CloudTrail 이벤트에는 사람이 읽을 수 있는 보안 주체 식별자(기본 ARN)가 포함되어 있지 않습니다. 수신자 계정의 관리자는 로그에 보안 주체 ARN을 포함하도록 선택할 수 있습니다.

자세한 내용은 AWS Lake Formation 개발자 가이드의 크로스 계정 CloudTrail 로깅을 참조하세요.

참고

- the section called “로깅 및 모니터링”

교차 계정 리소스 소유권 및 결제

AWS 계정 하나(계정 A)의 사용자가 다른 계정(계정 B)에서 새 리소스(예: 데이터베이스)를 생성하는 경우 리소스의 소유자는 해당 리소스가 생성된 계정인 계정 B입니다. 계정 B의 관리자는 읽기 및 쓰기와 세 번째 계정에게 액세스 권한 부여 등을 비롯해 새 리소스에 액세스하기 위한 전체 권한을 자동으로 가져옵니다. 계정 A의 사용자는 계정 B가 부여한 적절한 권한이 있는 경우에만 방금 생성한 리소스에 액세스할 수 있습니다.

스토리지 비용과 새 리소스와 직접 연관된 기타 비용이 리소스 소유자인 계정 B에게 청구됩니다. 리소스를 생성한 사용자의 요청 비용은 요청자의 계정인 계정 A로 청구됩니다.

AWS Glue 결제 및 요금에 대한 자세한 내용은 AWS 요금 책정 방식을 참조하세요.

교차 계정 액세스 제한 사항

AWS Glue 교차 계정 액세스에는 다음과 같은 제한 사항이 있습니다.

- 리전의 AWS Glue 지원 이전에 Amazon Athena 또는 Amazon Redshift Spectrum을 사용하여 데이터베이스와 테이블을 생성했으며 리소스 소유자 계정이 Amazon Athena 데이터 카탈로그를 AWS Glue로 마이그레이션하지 않은 경우에는 AWS Glue에 대한 교차 계정 액세스가 허용되지 않습니다. GetCatalogImportStatus(get_catalog_import_status)를 사용하여 현재 마이그레이션 상태를 찾을 수 있습니다. Athena 카탈로그를 AWS Glue로 마이그레이션하는 방법에 대한 자세한 내용은 Amazon Athena 사용 설명서의 AWS Glue Data Catalog로 단계별 업그레이드를 참조하세요.
- 교차 계정 액세스는 데이터베이스, 테이블, 사용자 정의 함수 및 연결을 포함한 데이터 카탈로그 리소스에 대해서만 지원됩니다.
- Athena에서 데이터 카탈로그에 대한 교차 계정 액세스를 사용하려면 카탈로그를 Athena `DataCatalog` 리소스로 등록해야 합니다. 지침은 Amazon Athena 사용 설명서의 다른 계정에서 AWS Glue Data Catalog 등록을 참조하세요.

AWS Glue 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 AWS Glue 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- AWS Glue에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 AWS 계정 외부의 사람이 내 AWS Glue 리소스에 액세스할 수 있게 허용하기를 원합니다.

AWS Glue에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojacksonIAM 사용자가 콘솔을 사용하여 가상 *my-example-widget*리소스에 대한 세부 정보를 보려고 하지만 가상 glue:*GetWidget*권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 glue:GetWidget on resource: my-example-widget
```

이 경우 glue:*GetWidget*작업을 사용하여 *my-example-widget*리소스에 액세스할 수 있도록 mateojackson사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하십시오. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS Glue에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 AWS Glue에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하십시오. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 내 AWS Glue 리소스에 액세스할 수 있게 허용하기를 원합니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- AWSGlue 에서 이러한 기능을 지원하는지 여부를 알아보려면 AWS Glue의 작동 방식 IAM 섹션을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

AWS Lake Formation 액세스 제어 모델

AWS Glue 5.0은 AWS Lake Formation을 통해 데이터에 액세스하는 두 가지 모델을 지원합니다.

주제

- 전체 테이블 액세스를 위해 AWS Lake Formation에서 AWS Glue 사용
- 세분화된 액세스 제어를 위해 AWS Lake Formation과 함께 AWS Glue 사용

전체 테이블 액세스를 위해 AWS Lake Formation에서 AWS Glue 사용

전체 테이블 액세스 소개

AWS Glue 5.0은 AWS Lake Formation에 정의된 정책에 따라 Apache Spark에서 전체 테이블 액세스(FTA) 제어를 지원합니다. 이 기능을 사용하면 작업 역할에 전체 테이블 액세스 권한이 있을 때 AWS Lake Formation 등록 테이블의 AWS Glue Spark 작업에서 읽기 및 쓰기 작업을 수행할 수 있습니다.

FTA는 테이블 수준에서 보안 규정을 준수해야 하는 사용 사례에 이상적이고, AWS Lake Formation 테이블이 있는 탄력적 분산형 데이터세트(RDD), 사용자 지정 라이브러리 및 사용자 정의 함수(UDF)를 비롯한 Spark 기능을 지원합니다.

AWS Glue Spark 작업이 전체 테이블 액세스(FTA)로 구성된 경우 AWS Lake Formation 자격 증명은 AWS Lake Formation 등록 테이블에 대한 Amazon S3 데이터를 읽거나 쓰는 데 사용되는 반면, 작업의 런타임 역할 자격 증명은 AWS Lake Formation에 등록되지 않은 테이블을 읽거나 쓰는 데 사용됩니다. 이 기능을 사용하면 Apache Hive 및 Iceberg 테이블에서 CREATE, ALTER, DELETE, UPDATE 및 MERGE INTO 문을 포함한 데이터 조작 언어(DML) 작업을 수행할 수 있습니다.

> **Note**
>
> 요구 사항을 검토하고 세분화된 액세스 제어(FGAC) 또는 전체 테이블 액세스(FTA)가 요구 사항에 적합한지 확인합니다. 지정된 AWS Glue 작업에 대해 단 하나의 AWS Lake Formation 권한 방법이 활성화될 수 있습니다. 작업은 전체 테이블 액세스(FTA)와 세분화된 액세스 제어(FGAC)를 동시에 실행할 수 없습니다.

AWS Glue에서의 전체 테이블 액세스(FTA) 작동 방식

AWS Lake Formation에서는 데이터 액세스 제어에 있어 세분화된 액세스 제어(FGAC)와 전체 테이블 액세스(FTA)라는 두 가지 접근 방식을 제공합니다. FGAC는 열, 행 및 셀 수준 필터링을 통해 보안이 향상되므로 세분화된 권한이 필요한 시나리오에 적합합니다. FTA는 테이블 수준 권한이 필요한 간단한 액세스 제어에 적합합니다. 세분화된 액세스 모드를 활성화할 필요가 없어 구현이 간소화되고, 시스템 드라이버와 시스템 실행기를 방지하여 성능을 개선하고 비용을 절감하며, 읽기 및 쓰기 작업(예: CREATE, ALTER, DELETE, UPDATE, MERGE INTO 명령)을 모두 지원합니다.

AWS Glue 4.0에서 AWS Lake Formation 기반 데이터 액세스는 AWS Glue에서 제공하는 유틸리티 클래스인 GlueContext 클래스를 통해 작동했습니다. AWS Glue 5.0에서는 네이티브 Spark SQL, Spark DataFrames를 통해 AWS Lake Formation 기반 데이터 액세스가 제공되고, 계속해서 GlueContext 클래스를 통해 지원됩니다.

전체 테이블 액세스 구현

1단계: AWS Lake Formation에서 전체 테이블 액세스 활성화

전체 테이블 액세스(FTA) 모드를 사용하려면 AWS Lake Formation에서 IAM 세션 태그 검증 없이 타사 쿼리 엔진이 데이터에 액세스하도록 허용해야 합니다. 활성화하려면 전체 테이블 액세스를 위한 애플리케이션 통합의 단계를 따릅니다.

2단계: 작업 런타임 역할에 대한 IAM 권한 설정

기본 데이터에 대한 읽기 또는 쓰기 액세스의 경우 작업 런타임 역할에는 AWS Lake Formation 권한 외에도 `lakeformation:GetDataAccess` IAM 권한이 필요합니다. 이 권한을 통해 AWS Lake Formation은 데이터에 액세스하기 위한 임시 자격 증명 요청을 승인합니다.

다음은 Amazon S3의 스크립트에 액세스할 수 있는 IAM 권한을 제공하는 방법, S3에 로그 업로드, AWS Glue Glue API 권한 및 AWS Lake Formation에 액세스할 수 있는 권한에 대한 정책 예제입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/scripts/*"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:GetDatabases",
        "glue:GetTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreateTable",
```

```
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/default",
        "arn:aws:glue:us-east-1:123456789012:table/default/*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    }
  ]
}
```

2.1단계: AWS Lake Formation 권한 구성

Amazon S3에서 데이터를 읽는 AWS Glue Spark 작업에는 AWS Lake Formation SELECT 권한이 필요합니다.

Amazon S3에서 데이터를 쓰거나 삭제하는 AWS Glue Spark 작업에는 AWS Lake Formation ALL 권한이 필요합니다.

AWS Glue Data Catalog와 상호 작용하는 AWS Glue Spark 작업에는 DESCRIBE, ALTER, DROP 권한이 필요합니다.

3단계: AWS Lake Formation을 사용하여 전체 테이블 액세스를 위한 Spark 세션 초기화

AWS Lake Formation에 등록된 테이블에 액세스하려면 Spark 초기화 중에 다음 구성을 설정하여 AWS Lake Formation 자격 증명을 사용하도록 Spark를 구성해야 합니다.

AWS Lake Formation에 등록된 테이블에 액세스하려면 AWS Lake Formation 자격 증명을 사용하도록 Spark 세션을 명시적으로 구성해야 합니다. Spark 세션을 초기화할 때 다음 구성 추가:

```
from pyspark.sql import SparkSession

# Initialize Spark session with Lake Formation configurations
spark = SparkSession.builder \
```

```
    .appName("Lake Formation Full Table Access") \
    .config("spark.sql.catalog.glue_catalog",
 "org.apache.spark.sql.catalog.hive.GlueCatalog") \
    .config("spark.sql.catalog.glue_catalog.glue.lakeformation-enabled", "true") \
    .config("spark.sql.defaultCatalog", "glue_catalog") \
    .getOrCreate()
```

키 구성:

- `spark.sql.catalog.glue_catalog`: GlueCatalog 구현을 사용하는 'glue_catalog' 카탈로그 등록
- `spark.sql.catalog.glue_catalog.glue.lakeformation-enabled`: 이 카탈로그에 대한 AWS Lake Formation 통합을 명시적으로 활성화
- 카탈로그 이름(이 예제에서는 'glue_catalog')을 사용자 지정할 수 있지만 두 구성 설정 모두에서 일관적이어야 합니다.

Hive

```
--conf spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.AWS
 Glue.accesscontrol.AWSLakeFormationCredentialResolver
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg

```
--conf spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.AWS
 Glue.accesscontrol.AWSLakeFormationCredentialResolver
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
--conf spark.sql.catalog.<catalog>.AWS Glue.lakeformation-enabled=true
```

- `spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.AWS Glue.accesscontrol.AWSLakeFormationCredentialResolver`: AWS Lake Formation 등록 테이블에 AWS Lake Formation S3 자격 증명을 사용하도록 EMR 파일 시스템(EMRFS) 구성 테이블이 등록되지 않은 경우 작업의 런타임 역할 자격 증명을 사용합니다.
- `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` 및 `spark.hadoop.fs.s3.folderObject.autoAction.disabled=true`: S3 폴더를 생성할 때 $folder$ 접미사 대신 콘텐츠 유형 헤더 application/x-directory를 사용하도록 EMRFS를 구성합니다. AWS Lake Formation 자격 증명은 $folder$ 접미사가 있는 테이블 폴더 읽기를 허용하지 않으므로 AWS Lake Formation 테이블을 읽을 때 필요합니다.
- `spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true`: 생성 전에 테이블 위치가 비어 있는지 확인하는 작업을 건너뛰도록 Spark를 구성합니다. AWS Lake Formation 등록 테이블에서는 필수입니다. 위치가 비어 있는지 확인하기 위한 AWS Lake Formation 자격 증명은 AWS Glue Data Catalog 테이블을 생성한 후에만 사용할 수 있기 때문입니다. 이 구성이 없으면 작업의 런타임 역할 자격 증명이 빈 테이블 위치를 검증합니다.
- `spark.sql.catalog.createDirectoryAfterTable.enabled=true`: Hive 메타스토어에서 테이블 생성 후 Amazon S3 폴더를 생성하도록 Spark를 구성합니다. AWS Lake Formation 등록 테이블에서는 필수입니다. Amazon S3 폴더를 생성하기 위한 AWS Lake Formation 자격 증명은 AWS Glue Data Catalog 테이블을 생성한 후에만 사용할 수 있기 때문입니다.
- `spark.sql.catalog.dropDirectoryBeforeTable.enabled=true`: Hive 메타스토어에서 테이블 삭제 전 Amazon S3 폴더를 제거하도록 Spark를 구성합니다. AWS Lake Formation 등록 테이블에서는 필수입니다. AWS Glue Data Catalog에서 테이블을 삭제한 후에는 S3 폴더를 삭제하기 위한 AWS Lake Formation 자격 증명을 사용할 수 없기 때문입니다.
- `spark.sql.catalog.<catalog>.AWS Glue.lakeformation-enabled=true`: AWS Lake Formation 등록 테이블에 AWS Lake Formation Amazon S3 자격 증명을 사용하도록 Iceberg 카탈로그를 구성합니다. 테이블이 등록되지 않은 경우 기본 환경 자격 증명을 사용합니다.

사용 패턴

DataFrames에서 FTA 사용

Spark에 익숙한 사용자의 경우 DataFrames를 AWS Lake Formation 전체 테이블 액세스에서 사용할 수 있습니다.

AWS Glue 5.0은 Lake Formation 전체 테이블 액세스에 대한 네이티브 Spark 지원을 추가하여 보호되는 테이블 사용 방식을 간소화합니다. 이 기능을 사용하면 전체 테이블 액세스 권한이 부여될 때 AWS Glue 5.0 AWS Glue Spark 작업에서 데이터를 직접 읽고 쓸 수 있으므로 이전에 특정 추출, 전환, 적재

(ETL) 작업의 제한 사항이 제거됩니다. 이제 탄력적 분산형 데이터세트(RDD), 사용자 지정 라이브러리, 사용자 정의 함수(UDF) 등의 고급 Spark 기능을 AWS Lake Formation 테이블에서 활용할 수 있습니다.

AWS Glue 5.0의 네이티브 Spark FTA

AWS Glue 5.0은 AWS Lake Formation에 정의된 정책에 따라 Apache Spark에서 전체 테이블 액세스(FTA) 제어를 지원합니다. 이 제어 수준은 테이블 수준에서 보안 규정을 준수해야 하는 사용 사례에 적합합니다.

Apache Iceberg 테이블 예제

```
from pyspark.sql import SparkSession

catalog_name = "spark_catalog"
aws_region = "us-east-1"
aws_account_id = "123456789012"
warehouse_path = "s3://amzn-s3-demo-bucket/warehouse/"

spark = SparkSession.builder \

 .config("spark.sql.extensions","org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensi
 \
    .config(f"spark.sql.catalog.{catalog_name}",
 "org.apache.iceberg.spark.SparkSessionCatalog") \
    .config(f"spark.sql.catalog.{catalog_name}.warehouse", f"{warehouse_path}") \
    .config(f"spark.sql.catalog.{catalog_name}.client.region",f"{aws_region}") \
    .config(f"spark.sql.catalog.{catalog_name}.glue.account-id",f"{aws_account_id}") \
    .config(f"spark.sql.catalog.{catalog_name}.glue.lakeformation-enabled","true") \
    .config(f"spark.sql.catalog.dropDirectoryBeforeTable.enabled", "true") \
    .config(f"spark.sql.catalog.{catalog_name}.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog") \
    .config(f"spark.sql.catalog.{catalog_name}.io-impl",
 "org.apache.iceberg.aws.s3.S3FileIO") \
    .config("spark.sql.defaultCatalog", catalog_name) \  # Add this line
    .getOrCreate()

database_name = "your_database"
table_name = "your_table"

df = spark.sql(f"select * from {database_name}.{table_name}")
df.show()
```

필수 IAM 권한

AWS Glue 작업 실행 역할에 필요한 사항:

```
{
    "Action": "lakeformation:GetDataAccess",
    "Resource": "*",
    "Effect": "Allow"
}
```

추가로 데이터 위치에 대해 적절한 S3 액세스 권한.

Lake Formation 구성

5.0에서 AWS Glue 네이티브 Spark FTA를 사용하기 전에:

1. AWS Lake Formation에서 IAM 세션 태그 검증 없이 타사 쿼리 엔진이 데이터에 액세스하도록 허용
2. AWS Lake Formation 콘솔을 통해 AWS Glue 작업 실행 역할에 적절한 테이블 권한 부여
3. 위 예제에 표시된 필수 파라미터로 Spark 세션 구성

DynamicFrames에서 FTA 사용

AWS Glue의 네이티브 DynamicFrames는 최적화된 ETL 작업을 위해 AWS Lake Formation 전체 테이블 액세스에서 사용할 수 있습니다. 전체 테이블 액세스(FTA)는 테이블 수준에서 권한을 부여하는 보안 모델을 제공하고, 행 및 열 수준 권한 검사의 오버헤드를 우회하므로 세분화된 액세스 제어(FGAC)에 비해 데이터 처리 속도가 빨라집니다. 이 접근 방식은 전체 테이블을 처리하고 테이블 수준 권한이 보안 요구 사항을 충족해야 하는 경우에 유용합니다.

AWS Glue 4.0에서는 FTA를 사용하는 DynamicFrames에 특정 GlueContext 구성이 필요했습니다. FTA를 사용하는 기존 AWS Glue 4.0 DynamicFrame 코드는 AWS Glue 5.0에서 계속 작동하지만 최신 버전이 더욱 유연하게 네이티브 Spark FTA 지원도 제공합니다. 새로운 개발의 경우 DataFrames 섹션에서 설명하는 네이티브 Spark 접근 방식을 사용하는 것이 좋습니다. 특히 AWS Lake Formation 테이블이 있는 탄력적 분산형 데이터세트(RDD), 사용자 지정 라이브러리 및 사용자 정의 함수(UDF)와 같은 추가 기능이 필요한 경우 더욱 바람직합니다.

필수 권한

Glue 작업을 실행하는 IAM 역할에 필요한 사항:

- `lakeformation:GetDataAccess` 권한

- Lake Formation 콘솔을 통해 부여된 적절한 Lake Formation 테이블 권한

AWS Glue 5.0에서의 DynamicFrame 구현 예제

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

# Initialize Glue context
sc = SparkContext()
glueContext = GlueContext(sc)

# Configure catalog for Iceberg tables
catalog_name = "glue_catalog"
aws_region = "us-east-1"
aws_account_id = "123456789012"
warehouse_path = "s3://amzn-s3-demo-bucket/warehouse/"

spark = glueContext.spark_session
spark.conf.set(f"spark.sql.catalog.{catalog_name}",
 "org.apache.iceberg.spark.SparkCatalog")
spark.conf.set(f"spark.sql.catalog.{catalog_name}.warehouse", f"{warehouse_path}")
spark.conf.set(f"spark.sql.catalog.{catalog_name}.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog")
spark.conf.set(f"spark.sql.catalog.{catalog_name}.io-impl",
 "org.apache.iceberg.aws.s3.S3FileIO")
spark.conf.set(f"spark.sql.catalog.{catalog_name}.glue.lakeformation-enabled","true")
spark.conf.set(f"spark.sql.catalog.{catalog_name}.client.region",f"{aws_region}")
spark.conf.set(f"spark.sql.catalog.{catalog_name}.glue.id", f"{aws_account_id}")

# Read Lake Formation-protected table with DynamicFrame
df = glueContext.create_data_frame.from_catalog(
    database="your_database",
    table_name="your_table"
)
```

추가 구성

AWS Glue Studio 노트북에서 전체 테이블 액세스 모드 구성

AWS Glue Studio 노트북의 대화형 Spark 세션에서 AWS Lake Formation 등록 테이블에 액세스하려면 호환성 권한 모드를 사용해야 합니다. 대화형 세션을 시작하기 전에 `%%configure` 매직 명령을 사

용하여 Spark 구성을 설정합니다. 세션이 시작된 후에는 적용할 수 없으므로 이 구성은 노트북의 첫 번째 명령이어야 합니다. 테이블 유형을 기반으로 구성 선택:

Hive 테이블

```
%%configure
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationC
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg 테이블

```
%%configure
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationC
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
--conf spark.sql.catalog.glue_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.glue_catalog.warehouse=s3://example-s3-bucket_DATA_LOCATION
--conf spark.sql.catalog.glue_catalog.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog
--conf spark.sql.catalog.glue_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO
--conf spark.sql.catalog.glue_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.glue_catalog.glue.region=REGION
```

자리 표시자 바꾸기:

- S3_DATA_LOCATION: *s3://amzn-s3-demo-bucket*
- REGION: *AWS ##(#: us-east-1)*
- ACCOUNT_ID: *AWS ## ID*

> **Note**
>
> 노트북에서 Spark 작업을 실행하기 전에 이러한 구성을 설정해야 합니다.

지원되는 작업

이러한 작업은 AWS Lake Formation 자격 증명을 사용하여 테이블 데이터에 액세스합니다.

> **Note**
>
> AWS Lake Formation 활성화:
>
> - FTA: Spark 구성 `spark.sql.catalog.{catalog_name}.glue.lakeformation-enabled` 활성화

- CREATE TABLE
- ALTER TABLE
- INSERT INTO
- INSERT OVERWRITE
- SELECT
- UPDATE
- MERGE INTO
- DELETE FROM
- ANALYZE TABLE
- REPAIR TABLE
- DROP TABLE
- Spark 데이터 소스 쿼리
- Spark 데이터 소스 쓰기

> **Note**
>
> 위에 나열되지 않은 작업은 계속해서 IAM 권한을 사용하여 테이블 데이터에 액세스합니다.

AWS Glue 4.0에서 AWS Glue 5.0 FTA로 마이그레이션

4.0 GlueContext FTA에서 AWS Glue 5.0 AWS Glue 네이티브 Spark FTA로 마이그레이션하는 경우:

1. AWS Lake Formation에서 IAM 세션 태그 검증 없이 타사 쿼리 엔진이 데이터에 액세스하도록 허용합니다. 1단계: AWS Lake Formation에서 전체 테이블 액세스 활성화의 내용을 따릅니다.
2. 작업 런타임 역할을 변경할 필요가 없습니다. 하지만 AWS Glue 작업 실행 역할에 lakeformation:GetDataAccess IAM 권한이 있는지 확인합니다.
3. 스크립트에서 Spark 세션 구성을 수정합니다. 다음 Spark 구성이 있는지 확인:

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=s3://<bucket-name>/warehouse/
--conf spark.sql.catalog.spark_catalog.client.region=<REGION>
--conf spark.sql.catalog.spark_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.spark_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

4. GlueContext DataFrames가 네이티브 Spark DataFrames로 변경되도록 스크립트를 업데이트합니다.
5. AWS Glue 5.0을 사용하도록 AWS Glue 작업 업데이트

고려 사항 및 제한

- 전체 테이블 액세스가 활성화되지 않은 작업을 사용하여 Hive 테이블이 생성되고, 삽입된 레코드가 없는 경우 전체 테이블 액세스를 사용하는 작업의 후속 읽기 또는 쓰기는 실패합니다. 전체 테이블 액세스 권한이 없는 AWS Glue Spark가 $folder$ 접미사를 테이블 폴더 이름에 추가하기 때문입니다. 이 문제는 다음과 같은 방법으로 해결할 수 있습니다.
  - FTA가 활성화되지 않은 작업의 테이블에 1개 이상의 행을 삽입합니다.
  - S3의 폴더 이름에 $folder$ 접미사를 사용하지 않도록 FTA가 활성화되지 않은 작업을 구성합니다. Spark 구성 `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true`를 설정하면 가능합니다.
  - Amazon S3 콘솔 또는 Amazon S3 CLI를 사용하여 테이블 위치 `s3://path/to/table/table_name`에 Amazon S3 폴더를 생성합니다.
- 전체 테이블 액세스는 EMR 파일 시스템(EMRFS)에서만 작동합니다. S3A 파일 시스템은 호환되지 않습니다.
- Hive 및 Iceberg 테이블의 경우 전체 테이블 액세스가 지원됩니다. Hudi 및 Delta 테이블에 대한 지원은 아직 추가되지 않았습니다.

- AWS Lake Formation 세분 영역 액세스 제어(FGAC) 규칙 또는 AWS Glue Data Catalog 뷰가 있는 테이블을 참조하는 작업은 실패합니다. FGAC 규칙 또는 AWS Glue Data Catalog 뷰로 테이블을 쿼리하려면 FGAC 모드를 사용해야 합니다. AWS 설명서: 세분화된 액세스 제어를 위해 AWS Lake Formation에서 AWS Glue 사용에서 설명하는 단계에 따라 FGAC 모드를 활성화할 수 있습니다.
- 전체 테이블 액세스는 Spark 스트리밍을 지원하지 않습니다.
- FGAC와 동시에 사용할 수 없습니다.

세분화된 액세스 제어를 위해 AWS Lake Formation과 함께 AWS Glue 사용

개요

AWS Glue 버전 5.0 이상을 사용하면 AWS Lake Formation을 사용하여 S3에서 지원하는 Data Catalog 테이블에 세분화된 액세스 제어를 적용할 수 있습니다. 이 기능을 사용하면 Apache Spark용 AWS Glue 작업 내에서 read 쿼리에 대한 테이블, 행, 열 및 셀 수준 액세스 제어를 구성할 수 있습니다. Lake Formation 및 이를 AWS Glue와 함께 사용하는 방법에 대해 자세히 알아보려면 다음 섹션을 참조하세요.

Glue 4.0 이하에서 지원되는 AWS Lake Formation 권한을 가진 `GlueContext` 기반 테이블 수준 액세스 제어는 Glue 5.0에서 지원되지 않습니다. Glue 5.0에서는 새로운 Spark 네이티브 세분화된 액세스 제어(FGAC)를 사용합니다. 다음의 세부 정보를 적어 둡니다.

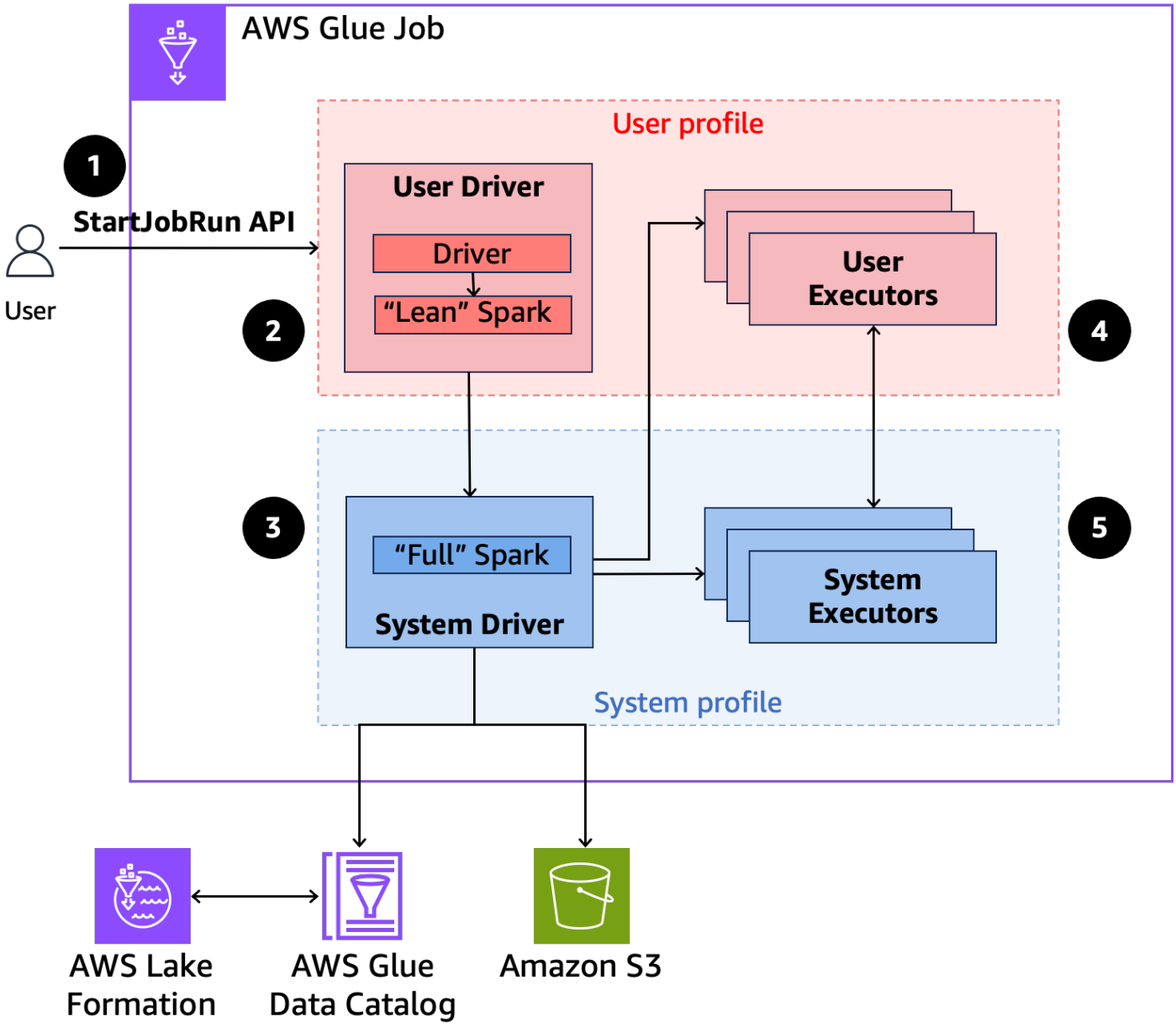
- row/column/cell 액세스 제어를 위한 세분화된 액세스 제어(FGAC)가 필요한 경우 Glue 4.0의 `GlueContext`/Glue DynamicFrame 및 이전 버전에서 Glue 5.0의 Spark 데이터프레임으로 마이그레이션해야 합니다. 예제는 GlueContext/Glue DynamicFrame에서 Spark DataFrame으로 마이그레이션 단원을 참조하십시오.
- 전체 테이블 액세스(FTA) 제어가 필요한 경우 Glue 5.0에서 DynamicFrames와 함께 FTA를 활용할 수 있습니다. 또한 AWS Lake Formation 테이블을 사용하여 탄력적 분산형 데이터세트(RDD), 사용자 지정 라이브러리 및 사용자 정의 함수(UDF)와 같은 추가 기능을 위해 네이티브 Spark 접근 방식으로 마이그레이션할 수 있습니다. 예제는 AWS Glue 4.0에서 AWS Glue 5.0 FTA로 마이그레이션을 참조하세요.
- FGAC가 필요하지 않은 경우, Spark 데이터프레임으로의 마이그레이션이 필요하지 않으며 작업 북마크, 푸시다운 조건자와 같은 `GlueContext` 기능은 계속 작동합니다.
- FGAC를 사용하는 작업에는 사용자 드라이버 1, 시스템 드라이버 1, 시스템 실행기 1, 대기 사용자 실행기 1, 이렇게 작업자가 최소 4명 필요합니다.

AWS Glue를 AWS Lake Formation과 함께 사용하면 추가 요금이 발생합니다.

AWS Glue에서 AWS Lake Formation을 사용하는 방식

AWS Glue를 Lake Formation과 함께 사용하면 AWS Glue에서 작업을 실행하는 경우 Lake Formation 권한 제어를 적용하기 위해 각 Spark 작업에 권한 계층을 적용할 수 있습니다. AWS Glue는 Spark 리소스 프로파일을 사용하여 작업을 효과적으로 실행하도록 두 개의 프로파일을 생성합니다. 사용자 프로파일은 사용자 제공 코드를 실행하는 반면, 시스템 프로필은 Lake Formation 정책을 적용합니다. 자세한 내용은 What is AWS Lake Formation 및 고려 사항 및 제한 사항을 참조하세요.

다음 개요에서는 AWS Glue가 Lake Formation 보안 정책에 따라 보호되는 데이터에 액세스하는 방법을 설명합니다.



1. 사용자가 AWS Lake Formation이 활성화된 AWS Glue 작업에서 `StartJobRun` API를 호출합니다.
2. AWS Glue는 사용자 드라이버로 작업을 전송하고 사용자 프로파일에서 작업을 실행합니다. 사용자 드라이버는 태스크를 시작하고, 실행기를 요청하며, S3 또는 Glue Catalog에 액세스할 수 없는 린 버전의 Spark를 실행합니다. 작업 계획을 빌드합니다.
3. AWS Glue는 시스템 드라이버라는 두 번째 드라이버를 설정하고 시스템 프로파일(권한 있는 자격 증명 포함)에서 실행합니다. AWS Glue는 통신을 위해 두 드라이버 사이에서 암호화된 TLS 채널을 설정합니다. 사용자 드라이버는 채널을 사용하여 작업 계획을 시스템 드라이버로 전송합니다. 시스템 드라이버는 사용자가 제출한 코드를 실행하지 않습니다. 전체 Spark를 실행하고 S3 및 데이터 액세스를 위해 Data Catalog와 통신합니다. 실행기를 요청하고 작업 계획을 일련의 실행 단계로 컴파일합니다.
4. 그런 다음, AWS Glue는 사용자 드라이버 또는 시스템 드라이버를 사용하여 실행기에서 단계를 실행합니다. 모든 단계의 사용자 코드는 사용자 프로파일 실행기에서만 실행됩니다.
5. AWS Lake Formation에서 보호하는 Data Catalog 테이블에서 데이터를 읽는 단계 또는 보안 필터를 적용하는 단계는 시스템 실행기로 위임됩니다.

최소 작업자 요구 사항

AWS Glue의 Lake Formation 지원 작업에는 사용자 드라이버 1, 시스템 드라이버 1, 시스템 실행기 1, 대기 사용자 실행기 1, 이렇게 작업자가 최소 4명 필요합니다. 이는 표준 AWS Glue 작업에 필요한 최소 작업자 2명보다 많습니다.

AWS Glue의 Lake Formation 지원 작업에는 시스템 프로필과 사용자 프로필에 대해 하나씩, 두 개의 Spark 드라이버가 사용됩니다. 마찬가지로 실행기도 두 프로필로 나뉨:

- 시스템 실행기: Lake Formation 데이터 필터가 적용되는 작업을 처리합니다.
- 사용자 실행기: 필요에 따라 시스템 드라이버가 요청합니다.

Spark 작업은 본질적으로 지연되므로 AWS Glue는 사용자 실행기에 대해 두 드라이버를 뺀 후 총 작업자의 10%(최소 1개)를 예약합니다.

모든 Lake Formation 지원 작업은 오토 스케일링이 활성화되어 있습니다. 즉, 사용자 실행기는 필요한 경우에만 시작됩니다.

예제 구성은 고려 사항 및 제한 사항을 참조하세요.

작업 런타임 역할 IAM 권한

Lake Formation 권한은 AWS Glue 데이터 카탈로그 리소스, Amazon S3 위치 및 해당 위치의 기본 데이터에 대한 액세스를 제어합니다. IAM 권한은 Lake Formation 및 AWS Glue API와 리소스에 대한 액세스를 제어합니다. Data Catalog의 테이블에 액세스할 수 있는 Lake Formation 권한이 있더라도 (SELECT) `glue:Get*` API 작업에 IAM 권한이 없으면 작업이 실패합니다.

다음은 S3의 스크립트에 액세스할 수 있는 IAM 권한을 제공하는 방법, S3에 로그 업로드, AWS Glue API 권한 및 Lake Formation에 액세스할 수 있는 권한에 대한 정책 예제입니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScriptAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
            "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts",
            "arn:aws:s3:::*.amzn-s3-demo-bucket/*" ]
        },
        {
         "Sid": "LoggingAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
               "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
            ]
        },
        {
            "Sid": "GlueCatalogAccess",
            "Effect": "Allow",
            "Action": [
                 "glue:Get*",
                 "glue:Create*",
                 "glue:Update*"
            ],
            "Resource": ["*"]
```

```
        },
        {
            "Sid": "LakeFormationAccess",
            "Effect": "Allow",
            "Action": [
             "lakeformation:GetDataAccess"
             ],
            "Resource": ["*"]
        }
    ]
}
```

작업 런타임 역할에 대한 Lake Formation 권한 설정

먼저 Lake Formation에 Hive 테이블의 위치를 등록합니다. 그런 다음, 원하는 테이블에서 작업 런타임 역할에 대한 권한을 생성합니다. Lake Formation에 대한 자세한 내용은 AWS Lake Formation 개발자 안내서의 What is AWS Lake Formation?을 참조하세요.

Lake Formation 권한을 설정한 후 AWS Glue에서 Spark 작업을 제출할 수 있습니다.

작업 실행 제출

Lake Formation 권한 부여 설정을 완료한 후 AWS Glue에서 Spark 작업을 제출할 수 있습니다. Iceberg 작업을 실행하려면 다음과 같은 Spark 구성을 제공해야 합니다. Glue 작업 파라미터를 통해 구성하려면 다음 파라미터를 입력합니다.

- 키:

```
--conf
```

- 값:

```
spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
       --conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION>
       --conf spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID>
       --conf spark.sql.catalog.spark_catalog.client.region=<REGION>
       --conf spark.sql.catalog.spark_catalog.glue.endpoint=https://
glue.<REGION>.amazonaws.com
```

대화형 세션 사용

AWS Lake Formation 권한 부여 설정을 완료한 후 AWS Glue에서 대화형 세션을 사용할 수 있습니다. 코드를 실행하기 전에 %%configure 매직을 통해 다음 Spark 구성을 제공해야 합니다.

```
%%configure
{
    "--enable-lakeformation-fine-grained-access": "true",
    "--conf":
 "spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
 --conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION> --conf
 spark.sql.catalog.spark_catalog.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
 --conf spark.sql.catalog.spark_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO --
conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 --conf spark.sql.catalog.spark_catalog.client.region=<REGION> --conf
 spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID> --conf
 spark.sql.catalog.spark_catalog.glue.endpoint=https://glue.<REGION>.amazonaws.com"
}
```

Glue AWS 5.0 노트북용 FGAC 또는 대화형 세션

AWS Glue에서 FGAC(세분화된 액세스 제어)를 활성화하려면 첫 번째 셀을 생성하기 전에 Lake Formation에 필요한 Spark conf를 %%configure 매직의 일부로 지정해야 합니다.

나중에 `SparkSession.builder().conf("").get()` 또는 `SparkSession.builder().conf("").create()` 직접 호출을 사용하여 지정하는 것으로는 충분하지 않습니다. 이는 AWS Glue 4.0 동작에서 변경된 것입니다.

오픈 테이블 형식 지원

AWS Glue 버전 5.0 이상에는 Lake Formation을 기반으로 하는 세분화된 액세스 제어에 대한 지원이 포함되어 있습니다. AWS Glue는 Hive 및 Iceberg 테이블 유형을 지원합니다. 다음 표에서는 지원되는 모든 옵션을 설명합니다.

운영	Hive	Iceberg
DDL 명령	IAM 역할 권한만 있음	IAM 역할 권한만 있음
증분 쿼리	해당 사항 없음	완전 지원

운영	Hive	Iceberg
시간 이동 쿼리	이 테이블 형식에 해당되지 않음	완전 지원
메타데이터 테이블	이 테이블 형식에 해당되지 않음	지원되지만 특정 테이블은 숨겨집니다. 자세한 내용은 고려 사항 및 제한 사항을 참조하세요.
DML INSERT	IAM 권한만 있음	IAM 권한만 있음
DML UPDATE	이 테이블 형식에 해당되지 않음	IAM 권한만 있음
DML DELETE	이 테이블 형식에 해당되지 않음	IAM 권한만 있음
읽기 작업	완전 지원	완전 지원
저장 프로시저	해당 사항 없음	register_table 및 migrate를 제외하고 지원됩니다. 자세한 내용은 고려 사항 및 제한 사항을 참조하세요.

GlueContext/Glue DynamicFrame에서 Spark DataFrame으로 마이그레이션

다음은 Glue 4.0의 GlueContext/Glue DynamicFrame에서 Glue 5.0의 Spark DataFrame으로 마이그레이션하는 Python 및 Scala 예시입니다.

Python

이전:

```
escaped_table_name= '`<dbname>`.`<table_name>`'

additional_options = {
  "query": f'select * from {escaped_table_name} WHERE column1 = 1 AND column7 = 7'
}
```

```
# DynamicFrame example
dataset = glueContext.create_data_frame_from_catalog(
    database="<dbname>",
    table_name=escaped_table_name,
    additional_options=additional_options)
```

이후:

```
table_identifier= '`<catalogname>`.`<dbname>`.`<table_name>`"' #catalogname is optional

# DataFrame example
dataset = spark.sql(f'select * from {table_identifier} WHERE column1 = 1 AND column7 =
 7')
```

Scala

이전:

```
val escapedTableName = "`<dbname>`.`<table_name>`"

val additionalOptions = JsonOptions(Map(
    "query" -> s"select * from $escapedTableName WHERE column1 = 1 AND column7 = 7"
    )
)

# DynamicFrame example
val datasource0 = glueContext.getCatalogSource(
    database="<dbname>",
    tableName=escapedTableName,
    additionalOptions=additionalOptions).getDataFrame()
```

이후:

```
val tableIdentifier = "`<catalogname>`.`<dbname>`.`<table_name>`" //catalogname is
 optional

# DataFrame example
val datasource0 = spark.sql(s"select * from $tableIdentifier WHERE column1 = 1 AND
 column7 = 7")
```

고려 사항 및 제한 사항

AWS Glue와 함께 Lake Formation을 사용하는 경우 다음 고려 사항 및 제한 사항을 고려합니다.

Lake Formation을 사용하는 AWS Glue는 AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부)를 제외한 지원되는 모든 리전에서 사용할 수 있습니다.

- AWS Glue는 Apache Hive 및 Apache Iceberg 테이블에 대해서만 Lake Formation을 통한 세분화된 액세스 제어를 지원합니다. Apache Hive 형식으로는, Parquet, ORC 및 CSV가 포함됩니다.
- Spark 작업에서 Lake Formation만 사용할 수 있습니다.
- Lake Formation을 사용하는 AWS Glue는 작업 전체에서 단일 Spark 세션만 지원합니다.
- Lake Formation이 활성화된 경우, AWS Glue에는 하나의 시스템 드라이버, 시스템 실행기, 하나의 사용자 드라이버 및 선택적으로 사용자 실행기(작업에 UDFs 또는 `spark.createDataFrame`가 있는 경우에 필요)가 필요하므로 더 많은 수의 작업자가 필요합니다.
- Lake Formation을 사용하는 AWS Glue는 리소스 링크를 통해 공유되는 교차 계정 테이블 쿼리만 지원합니다. 리소스-링크에는 소스 계정의 리소스와 동일한 이름이 지정되어야 합니다.
- AWS Glue 작업에 대해 세분화된 액세스 제어를 활성화하려면 `--enable-lakeformation-fine-grained-access` 작업 파라미터를 전달합니다.
- AWS Glue 다중 카탈로그 계층 구조로 작동하도록 AWS Glue 작업을 구성할 수 있습니다. AWS Glue `StartJobRun` API와 함께 사용할 구성 파라미터에 대한 자세한 내용은 EMR Serverless에서 AWS Glue 다중 카탈로그 계층 구조 작업을 참조하세요.
- 다음은 지원되지 않습니다.
  - 복원력 있는 분산 데이터세트(RDD)
  - Spark 스트리밍
  - Lake Formation에 부여된 권한으로 쓰기
  - 중첩된 열에 대한 액세스 제어
- AWS Glue는 다음을 포함하여 시스템 드라이버의 완전한 격리를 저해할 수 있는 기능을 차단합니다.
  - UDT, HiveUDF 및 사용자 지정 클래스가 포함된 사용자 정의 함수
  - 사용자 지정 데이터 소스
  - Spark 확장, 커넥터 또는 메타스토어에 대한 추가 jar 제공
  - `ANALYZE TABLE` 명령
- 액세스 제어, `EXPLAIN PLAN` 및 DDL 작업(예: `DESCRIBE TABLE`)을 적용하려면 제한된 정보를 노출하지 않습니다.

- AWS Glue는 Lake Formation 지원 애플리케이션의 시스템 드라이버 Spark 로그에 대한 액세스를 제한합니다. 시스템 드라이버는 더 많은 액세스 권한으로 실행되므로 시스템 드라이버가 생성하는 이벤트 및 로그에는 민감한 정보가 포함될 수 있습니다. 권한이 없는 사용자 또는 코드가 이 민감한 데이터에 액세스하지 못하도록 AWS Glue는 시스템 드라이버 로그에 대한 액세스를 비활성화했습니다. 문제 해결은 AWS Support에 문의하세요.
- Lake Formation에 테이블 위치를 등록한 경우 AWS Glue 작업 런타임 역할에 대한 IAM 권한과 관계없이 데이터 액세스 경로는 Lake Formation에 저장된 자격 증명을 통과합니다. 테이블 위치에 등록된 역할을 잘못 구성하면 테이블 위치에 대한 S3 IAM 권한이 있는 역할을 사용하는 제출된 작업이 실패합니다.
- Lake Formation 테이블에 쓰는 경우 Lake Formation에 부여된 권한이 아닌 IAM 권한을 사용합니다. 작업 런타임 역할에 필요한 S3 권한이 있는 경우 이를 사용하여 쓰기 작업을 실행할 수 있습니다.

다음은 Apache Iceberg를 사용하는 경우 고려 사항 및 제한 사항입니다.

- Apache Iceberg는 세션 카탈로그에서만 사용할 수 있으며, 임의로 이름이 지정된 카탈로그에서는 사용할 수 없습니다.
- Lake Formation에 등록된 Iceberg 테이블은 메타데이터 테이블 `history`, `metadata_log_entries`, `snapshots`, `files`, `manifests`, `refs`만 지원합니다. AWS Glue는 `partitions`, `path`, `summaries`와 같이 민감한 데이터를 포함할 수 있는 열을 숨깁니다. 이 제한 사항은 Lake Formation에 등록되지 않은 Iceberg 테이블에 적용되지 않습니다.
- Lake Formation에 등록하지 않은 테이블은 모든 Iceberg 저장 프로시저를 지원합니다. `register_table` 및 `migrate` 절차는 어떤 테이블에서도 지원되지 않습니다.
- V1 대신 Iceberg DataFrameWriterV2를 사용하는 것이 좋습니다.

작업자 할당 예

다음 파라미터로 구성한 작업의 경우입니다.

```
--enable-lakeformation-fine-grained-access=true
--number-of-workers=20
```

작업자 할당은 다음과 같습니다.

- 사용자 드라이버에 대한 작업자 한 명.
- 시스템 드라이버에 대한 작업자 한 명.
- 사용자 실행기용으로 예약된 나머지 작업자 18명의 10%(2명)입니다.

- 시스템 실행기에 작업자 16명이 할당됩니다.

오토 스케일링을 활성화하면 필요한 경우 사용자 실행기가 시스템 실행기의 할당되지 않은 용량을 활용할 수 있습니다.

사용자 실행기 할당 제어

다음 구성을 사용하여 사용자 실행기의 예약 비율을 조정할 수 있습니다.

```
--conf spark.dynamicAllocation.maxExecutorsRatio=<value between 0 and 1>
```

이 구성을 사용하면 사용 가능한 총 용량을 기준으로 예약된 사용자 실행기 수를 미세 조정할 수 있습니다.

문제 해결

문제 해결 방법은 다음 섹션을 참조하세요.

로깅

AWS Glue는 Spark 리소스 프로파일을 사용하여 작업 실행을 분할합니다. AWS Glue는 사용자 프로파일을 사용하여 사용자가 제공한 코드를 실행하는 반면, 시스템 프로필은 Lake Formation 정책을 적용합니다. 사용자 프로파일로 실행된 태스크의 로그에 액세스할 수 있습니다.

Live UI 및 Spark 기록 서버

Live UI 및 Spark 기록 서버에는 사용자 프로파일에서 생성된 모든 Spark 이벤트 및 시스템 드라이버에서 생성된 수정된 이벤트가 있습니다.

실행기 탭에서 사용자 및 시스템 드라이버의 모든 태스크를 볼 수 있습니다. 그러나 로그 링크는 사용자 프로파일에서만 사용할 수 있습니다. 또한 출력 레코드 수와 같은 일부 정보는 Live UI에서 수정됩니다.

Lake Formation 권한이 부족하여 작업 실패

작업 런타임 역할에 액세스 중인 테이블에서 SELECT 및 DESCRIBE를 실행할 권한이 있는지 확인합니다.

RDD 실행이 실패한 작업

AWS Glue는 현재 Lake Formation 지원 작업에서 탄력적 분산 데이터세트(RDD) 작업을 지원하지 않습니다.

Amazon S3의 데이터 파일에 액세스할 수 없음

Lake Formation에서 데이터 레이크의 위치를 등록했는지 확인합니다.

보안 검증 예외

AWS Glue에서 보안 검증 오류를 탐지했습니다. AWS Support에 문의하여 지원을 받으세요.

여러 계정에서 AWS Glue Data Catalog 및 테이블 공유

여러 계정에서 데이터베이스와 테이블을 공유하고 Lake Formation을 계속 사용할 수 있습니다. 자세한 내용은 Lake Formation에서의 교차 계정 데이터 공유 및 AWS Glue Data Catalog 및 테이블 교차 계정을 공유하려면 어떻게 해야 합니까?를 참조하세요.

다음 표에서는 워크로드에 대한 세분화된 액세스 제어(FGAC)와 전체 테이블 액세스(FTA) 중에서 선택하는 방법을 요약합니다.

Feature	세분화된 액세스 제어(FGAC)	전체 테이블 액세스(FTA)
Access Level	Column/row level	Full table
Use Case	Queries and ETL with limited permissions	ETL
Performance Impact	Requires system/user space transitions for access control evaluation, adding latency	Optimized performance

AWS Glue를 통해 Amazon S3 Access Grants 사용

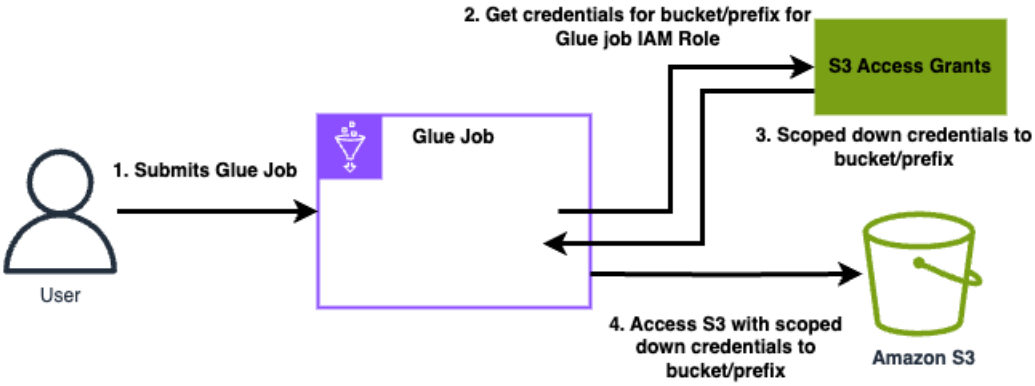
Glue 버전 5.0을 사용하면 Amazon S3 Access Grants에서 AWS Glue의 Amazon S3 데이터에 대한 액세스를 강화하기 위해 사용 가능한 확장 가능한 액세스 제어 솔루션이 제공됩니다. S3 데이터에 대한 권한 구성이 복잡하거나 대규모인 경우 S3 Access Grants를 사용하여 사용자 및 역할을 위한 S3 데이터 권한을 확장할 수 있습니다.

S3 Access Grants를 사용하여 Amazon S3 데이터에 대한 액세스를 AWS Glue 작업에 액세스할 수 있는 자격 증명에 연결된 런타임 역할 또는 IAM 역할에 의해 부여된 권한 이상으로 강화할 수 있습니다. 자세한 내용은 Amazon S3 사용 설명서에서 Amazon S3 Access Grants로 액세스 관리를 참조하세요.

AWS Glue가 S3 Access Grants에서 작동하는 방법

AWS Glue 버전 5.0 이상은 S3 Access Grants와의 네이티브 통합을 제공합니다. AWS Glue에서 S3 Access Grants를 활성화하고 Spark 작업을 실행할 수 있습니다. Spark 작업에서 S3 데이터에 대한 요청이 발생할 경우 Amazon S3에서는 특정 버킷, 접두사 또는 객체로 범위가 지정된 임시 보안 인증을 제공합니다.

다음은 AWS Glue가 S3 Access Grants가 액세스를 관리하는 데이터에 액세스하는 방법에 대한 개략적인 개요입니다.



1. 사용자는 Amazon S3에 저장된 데이터를 사용하는 AWS Glue Spark 작업을 제출합니다.
2. AWS Glue는 버킷, 접두사 또는 객체에 대한 액세스 권한을 부여하는 사용자에 대한 임시 자격 증명을 벤딩하도록 S3 Access Grants에 요청합니다.
3. AWS Glue는 사용자를 위한 AWS Security Token Service(STS) 토큰 형태로 임시 자격 증명을 반환합니다. 토큰의 범위는 S3 버킷, 접두사 또는 객체에 액세스할 수 있도록 지정됩니다.
4. AWS Glue는 STS 토큰을 사용하여 S3에서 데이터를 검색합니다.
5. AWS Glue는 S3로부터 데이터를 수신하고 사용자에게 결과를 반환합니다.

AWS Glue에서의 S3 Access Grants 고려 사항

AWS Glue와 함께 S3 Access Grants를 사용하는 경우에는 다음 동작 및 제한 사항을 참고하세요.

기능 지원

- S3 Access Grants는 AWS Glue 버전 5.0 이상에서 지원됩니다.
- Spark는 AWS Glue와 함께 S3 Access Grants를 사용하는 경우에 지원되는 유일한 작업 유형입니다.

- Delta Lake 및 Hudi는 AWS Glue에서 S3 Access Grants를 사용하는 경우에 지원되는 유일한 오픈 테이블 형식입니다.
- 다음 기능은 S3 Access Grants와 함께 사용할 경우 지원되지 않습니다.
  - Apache Iceberg 테이블
  - IAM 역할을 사용하는 Amazon S3에 대한 AWS CLI 요청
  - 오픈 소스 S3A 프로토콜을 통한 S3 액세스

동작 고려 사항

- AWS Glue는 자격 증명 캐시를 제공하여 사용자가 Spark 작업 내에서 동일한 자격 증명을 반복적으로 요청할 필요가 없도록 합니다. 따라서 AWS Glue는 자격 증명을 요청할 때 항상 기본 수준 권한을 요청합니다. 자세한 정보는 Amazon S3 사용 설명서의 S3 데이터에 대한 액세스 요청을 참조하세요.

AWS Glue에서 S3 Access Grants 설정

사전 조건

호출자 또는 관리자가 S3 Access Grants 인스턴스를 생성했습니다.

AWS Glue 정책 및 작업 구성 설정

AWS Glue에서 S3 Access Grants를 설정하려면 신뢰 및 IAM 정책을 구성하고 작업 파라미터를 통해 구성을 전달해야 합니다.

1. 권한 부여에 사용되는 역할(세션 또는 작업을 실행하는 AWS Glue 역할)에 대해 다음과 같은 최소 신뢰 및 IAM 정책을 구성합니다.

   신뢰 정책:

```
{
            "Sid": "Stmt1234567891011",
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity",
                "sts:SetContext"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "access-grants.s3.amazonaws.com"
```

```
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012",
                    "aws:SourceArn": "arn:aws:s3:<region>:123456789012:access-grants/
default"
                }
            }
        }
```

IAM 정책:

```
{
            "Sid": "S3Grants",
            "Effect": "Allow",
            "Action": [
                "s3:GetDataAccess",
                "s3:GetAccessGrantsInstanceForPrefix"
            ],
            "Resource": "arn:aws:s3:<region>:123456789012:access-grants/default"
        },
        {
            "Sid": "BucketLevelReadPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "s3:AccessGrantsInstanceArn": [
                        "arn:aws:s3:<region>:123456789012:access-grants/default"
                    ]
                }
            }
        },
        {
            "Sid": "ObjectLevelReadPermissions",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetObjectAcl",
                "s3:GetObjectVersionAcl",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "s3:AccessGrantsInstanceArn": [
                        "arn:aws:s3:<region>:123456789012:access-grants/default"
                    ]
                }
            }
        }
```

2. AWS Glue 작업에서 AWS Glue 작업 파라미터 또는 SparkConf를 통해 다음 Spark 구성을 전달합니다.

```
--conf spark.hadoop.fs.s3.s3AccessGrants.enabled=true \
--conf spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM=false
```

AWS Glue의 로깅 및 모니터링

ETL(추출, 변환 및 로드) 작업 실행을 자동화할 수 있습니다. AWS Glue는 모니터링할 수 있는 크롤러 및 작업에 대한 지표를 제공합니다. 필요한 메타데이터로 AWS Glue Data Catalog을 설치한 후, AWS Glue는 환경 상태에 대한 지표를 제공합니다. Cron을 기반으로 시간 기반 일정으로 작업 및 크롤러의 호출을 자동화할 수 있습니다. 이벤트 기반 트리거를 시작할 경우 작업을 시작할 수 있습니다.

AWS Glue는 AWS Glue에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. 추적을 생성하는 경우 CloudTrail 이벤트를 Amazon Simple Storage Service(Amazon S3) 버킷, Amazon CloudWatch Logs 및 Amazon CloudWatch Events로 지

속적으로 전송할 수 있습니다. 모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다.

Amazon CloudWatch Events를 사용하여 AWS 서비스를 자동화하고 애플리케이션 가용성 문제나 리소스 변경 같은 시스템 이벤트에 자동으로 응답합니다. AWS 서비스의 이벤트는 거의 실시간으로 CloudWatch Events로 전송됩니다. 어떤 이벤트에 관심이 있으며 이벤트가 규칙과 일치할 때 어떤 자동화된 작업을 수행할지를 표시하는 간단한 규칙을 작성할 수 있습니다.

다음 사항도 참조하세요.

- EventBridge를 사용한 AWS Glue 자동화
- 교차 계정 CloudTrail 로깅

클라우드에서 중요한 보안 요소 중 하나는 로깅입니다. 클라우드 인프라를 디버깅하고 보호하는 데 필요한 정보를 캡처하면서 보안 암호 및 기밀 자료를 캡처하지 않는 방식으로 로깅을 구성해야 합니다. 어떤 정보와 데이터가 로깅되는지 숙지해야 합니다.

AWS Glue의 규정 준수 확인

AWS 서비스(이)가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact(을)를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하세요.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다.AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 규정 준수 및 거버넌스 - 이러한 솔루션 구현 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수 기능을 배포하는 단계를 제공합니다.
- HIPAA 적격 서비스 참조-HIPAA 적격 서비스가 나열되어 있습니다. 모든 AWS 서비스에 HIPAA 자격이 있는 것은 아닙니다.
- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 컬렉션입니다.
- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에서는 AWS 서비스를 보호하기 위한 모범 사례를 요약하고 여러 프레임워크(미국 표준 기술 연구소(NIST),

결제 카드 산업 보안 표준 위원회(PCI), 국제 표준화기구(ISO) 등)에서 보안 컨트롤에 대한 지침을 매핑합니다.

- AWS Config 개발자 가이드의 규칙을 사용하여 리소스 평가 – AWS Config 서비스는 내부 사례, 산업 지침 및 규제에 대한 리소스 구성의 준수 상태를 평가합니다.
- AWS Security Hub – 이 AWS 서비스(은)는 AWS 내 보안 상태에 대한 포괄적인 보기를 제공합니다. Security Hub는 보안 컨트롤을 사용하여 AWS리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.
- Amazon GuardDuty - 이 AWS 서비스는 의심스럽고 악의적인 활동이 있는지 환경을 모니터링하여 AWS 계정, 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 탐지합니다. GuardDuty는 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 따르는 데 도움을 줄 수 있습니다.
- AWS Audit Manager - 이 AWS 서비스은(는) AWS 사용을 지속해서 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 간소화할 수 있도록 지원합니다.

AWS Glue의 복원성

AWS 글로벌 인프라는 AWS 지역 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 리전을 제공하며 이러한 가용 리전은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하십시오.

AWS Glue 작업 복원력에 대한 자세한 내용은 오류: AWS Glue의 VPC 간 장애 조치 동작을 참조하세요.

AWS Glue에서 인프라 보안

관리형 서비스인 AWS Glue은 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

AWS에서 게시한 API 직접 호출을 사용하여 네트워크를 통해 AWS Glue에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

주제

- AWS Glue에 대한 인터페이스 VPC 엔드포인트(AWS PrivateLink) 구성(AWS PrivateLink)
- 공유 Amazon VPC 구성

AWS Glue에 대한 인터페이스 VPC 엔드포인트(AWS PrivateLink) 구성 (AWS PrivateLink)

인터페이스 VPC 엔드포인트를 생성하여 VPC와 AWS Glue 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 비공개로 AWS Glue API에 액세스할 수 있도록 지원하는 AWS PrivateLink 기술로 구동됩니다. VPC의 인스턴스는 AWS Glue API와 통신하는 데 퍼블릭 IP 주소를 필요로 하지 않습니다. VPC와 AWS Glue 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다.

자세한 내용은 Amazon VPC 사용 설명서에서 인터페이스 VPC 종단점(AWS PrivateLink)을 참조하십시오.

AWS Glue VPC 엔드포인트 고려 사항

AWS Glue에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서에서 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다.

AWS Glue은 VPC에서 모든 API 작업에 대한 직접 호출 수행을 지원합니다.

AWS Glue에 대한 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔이나 AWS Command Line Interface(AWS CLI)을 사용하여 AWS Glue 서비스에 대한 VPC 엔드포인트를 생성할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 사용하여 AWS Glue용 VPC 종단점을 생성합니다.

- com.amazonaws.*region*.glue

엔드포인트에 프라이빗 DNS를 사용하도록 설정하는 경우, 리전에 대한 기본 DNS 이름(예: `glue.us-east-1.amazonaws.com`)을 사용하여 AWS Glue에 API 요청을 할 수 있습니다.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

AWS Glue에 대한 VPC 엔드포인트 정책 생성

AWS Glue에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하십시오.

예제: AWS Glue에서 작업 생성 및 업데이트를 허용하도록 하는 VPC 엔드포인트 정책

다음은 AWS Glue에 대한 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 AWS Glue 작업에 부여합니다.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
             "glue:CreateJob",
             "glue:UpdateJob",
             "iam:PassRole"
         ],
         "Resource":"*"
      }
   ]
}
```

예제: 읽기 전용 Data Catalog 액세스를 허용하는 VPC 엔드포인트 정책

다음은 AWS Glue에 대한 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 AWS Glue 작업에 부여합니다.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetTableVersion",
                "glue:GetTableVersions",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:SearchTables"
            ],
            "Resource": "*"
        }
    ]
}
```

공유 Amazon VPC 구성

AWS Glue는 Amazon Virtual Private Cloud에서 공유 Virtual Private Cloud(VPC)를 지원합니다. Amazon VPC 공유를 통해 여러 AWS 계정이 Amazon EC2 인스턴스 및 Amazon Relational Database Service(Amazon RDS) 데이터베이스와 같은 애플리케이션 리소스를 공유 중앙 관리형 Amazon VPC에 생성할 수 있습니다. 이 모델에서 VPC(소유자)를 소유하는 계정은 AWS Organizations의 동일한 조직에 속한 다른 계정(참여자)과 한 개 또는 여러 개의 서브넷을 공유합니다. 서브넷을 공유한 후 참여자는 공유된 서브넷의 해당 애플리케이션 리소스를 보고, 생성하고, 수정하고, 삭제할 수 있습니다.

AWS Glue에서 공유된 서브넷과의 연결을 생성하려면 계정 내에 보안 그룹을 생성하고 해당 그룹을 공유된 서브넷에 연결해야 합니다.

자세한 내용은 다음 주제를 참조하십시오.

- Amazon VPC 사용 설명서의 공유 VPC 작업
- AWS Organizations User Guide의 What Is AWS Organizations?

AWS Glue 문제 해결

AWS Glue 작업 시 문제가 발생한다면 이 섹션의 주제를 참조하세요.

주제

- AWS Glue 문제 해결 정보 모으기
- Spark 오류 문제 해결
- 크롤러가 Lake Formation 권한을 사용하는 경우 발생하는 크롤러 오류
- 로그에서 AWS Glue for Ray 오류 해결
- AWS Glue 기계 학습 예외 사항
- AWS Glue 할당량

AWS Glue 문제 해결 정보 모으기

AWS Glue의 오류나 예상 밖의 행동이 나타나고 AWS Support에 연결할 필요가 있다면 먼저 실패한 작업과 관련된 이름, ID 및 로그에 대한 정보를 모아야 합니다. 이 정보가 있으면 지원가 해당 문제의 해결을 지원할 수 있습니다.

계정 ID와 함께 각 실패 유형에 따라 다음 정보를 모으십시오.

크롤러가 실패하면 다음 정보를 모으십시오.

- 크롤러 이름

  크롤러 작업의 로그는 `/aws-glue/crawlers` 하에 CloudWatch Logs에 위치합니다.

테스트 연결이 되지 않으면 다음 정보를 모으십시오.

- 연결 이름
- 연결 ID
- `jdbc:protocol://host:port/database-name` 형식의 JDBC 연결 문자열.

  테스트 연결의 로그는 `/aws-glue/testconnection` 하에 CloudWatch Logs에 위치합니다.

작업이 실패하면 다음 정보를 모으십시오.

- 작업 이름
- `jr_xxxxx` 형식의 작업 실행 ID.

  작업 실행의 로그는 `/aws-glue/jobs` 하에 CloudWatch Logs에 위치합니다.

Spark 오류 문제 해결

AWS Glue에서 오류가 발생하면 다음 정보를 사용하여 문제의 원인을 찾아 수정할 수 있습니다.

> **Note**
>
> AWS Glue GitHub 리포지토리는 AWS Glue FAQ에 추가적인 문제 해결 지침을 포함합니다.

주제

- 오류: 사용 가능하지 않는 리소스
- 오류: VPC에서 서브넷 ID용 S3 엔드포인트 또는 NAT 게이트웨이를 찾을 수 없습니다.
- 오류: 보안 그룹의 인바운드 규칙이 필요합니다.
- 오류: 보안 그룹의 아웃바운드 규칙이 필요합니다.
- 오류: 작업 실행에 실패했습니다. 그 이유는 전달된 역할에는 AWS Glue 서비스에 대한 역할 수임 권한이 주어져야 하기 때문입니다.
- 오류: DescribeVpcEndpoints 작업이 승인되지 않았습니다. VPC ID vpc-id를 검증할 수 없습니다.
- 오류: DescribeRouteTables 작업이 승인되지 않았습니다. VPC ID: vpc-id에서 서브넷 ID: 서브넷-id를 검증할 수 없습니다.
- 오류: ec2:DescribeSubnets 호출이 실패했습니다.
- 오류: ec2:DescribeSecurityGroups 호출이 실패했습니다.
- 오류: AZ용 서브넷을 찾을 수 없습니다.
- 오류: JDBC 대상으로 작성할 경우의 작업 실행 예외
- 오류: Amazon S3: 객체의 스토리지 클래스에 대해 작업이 유효하지 않습니다.
- 오류: Amazon S3 시간 제한
- 오류: Amazon S3 액세스가 거부되었습니다.
- 오류: Amazon S3 액세스 키 ID가 없습니다.
- 오류: s3a:// URI를 사용해 Amazon S3에 액세스할 때 작업 실행이 실패합니다.
- 오류: Amazon S3 서비스 토큰이 만료되었습니다.
- 오류: 네트워크 인터페이스용 프라이빗 DNS를 찾을 수 없습니다.
- 오류: 개발 엔드포인트 프로비저닝이 실패했습니다.

- 오류: 노트북 서버 생성이 실패했습니다.
- 오류: 로컬 노트북 시작이 실패했습니다.
- 오류: 크롤러 실행이 실패했습니다.
- 오류: 파티션이 업데이트되지 않았습니다.
- 오류: 버전 불일치로 인해 작업 북마크 업데이트 실패
- 오류: 작업 북마크가 사용 설정된 경우 작업이 데이터를 재처리합니다.
- 오류: AWS Glue에서 VPC 간 장애 조치 동작

오류: 사용 가능하지 않는 리소스

AWS Glue가 리소스가 사용 가능하지 않다는 메시지를 반환하면 오류 메시지 또는 로그를 보고 해당 문제에 대한 자세한 내용을 확인할 수 있습니다. 다음 작업은 문제 해결을 위한 일반적인 방법을 설명합니다.

- 사용할 연결과 개발 엔드포인트의 경우, 탄력적 네트워크 인터페이스를 벗어나지 않은 클러스터를 확인합니다.

오류: VPC에서 서브넷 ID용 S3 엔드포인트 또는 NAT 게이트웨이를 찾을 수 없습니다.

메시지에서 서브넷 ID 및 VPC ID를 선택하여 문제를 진단하십시오.

- AWS Glue에 필요한 Amazon S3 VPC 엔드포인트 설정이 있는지 확인합니다. 또한, NAT 게이트웨이가 구성 중 일부라면 확인하시오. 자세한 내용은 Amazon S3용 Amazon VPC 엔드포인트 단원을 참조하십시오.

오류: 보안 그룹의 인바운드 규칙이 필요합니다.

적어도 하나 이상의 보안 그룹은 모든 내보내기 포트를 열어야 합니다. 트래픽을 제한하기 위해서 인바운드 규칙의 소스 보안 그룹은 동일한 보안 그룹으로 제한될 수 있습니다.

- 사용하는 연결의 경우, 자기 참조적 인바운드 규칙을 위한 보안 그룹을 확인하십시오. 자세한 내용은 데이터 스토어에 대한 네트워크 액세스 설정 단원을 참조하십시오.
- 개발 엔드포인트를 사용 중이면 자기 참조적 인바운드 규칙을 위한 보안 그룹을 확인하십시오. 자세한 내용은 데이터 스토어에 대한 네트워크 액세스 설정 단원을 참조하십시오.

오류: 보안 그룹의 아웃바운드 규칙이 필요합니다.

적어도 하나 이상의 보안 그룹은 모든 들여오기 포트를 열어야 합니다. 트래픽을 제한하기 위해서 아웃바운드 규칙의 소스 보안 그룹은 동일한 보안 그룹으로 제한될 수 있습니다.

- 사용하는 연결의 경우, 자기 참조적 아웃바운드 규칙을 위한 보안 그룹을 확인하십시오. 자세한 내용은 데이터 스토어에 대한 네트워크 액세스 설정 단원을 참조하십시오.
- 개발 엔드포인트를 사용 중이면 자기 참조적 아웃바운드 규칙을 위한 보안 그룹을 확인하십시오. 자세한 내용은 데이터 스토어에 대한 네트워크 액세스 설정 단원을 참조하십시오.

오류: 작업 실행에 실패했습니다. 그 이유는 전달된 역할에는 AWS Glue 서비스에 대한 역할 수임 권한이 주어져야 하기 때문입니다.

작업을 정의하는 사용자는 AWS Glue에 대한 `iam:PassRole` 권한이 있어야 합니다.

- 사용자가 AWS Glue 작업을 만들면, 해당 사용자 역할에 AWS Glue에 대한 `iam:PassRole`이 포함된 정책이 있는지 확인하십시오. 자세한 내용은 3단계: AWS Glue에 액세스하는 사용자 또는 그룹에 정책 연결 단원을 참조하십시오.

오류: DescribeVpcEndpoints 작업이 승인되지 않았습니다. VPC ID vpc-id를 검증할 수 없습니다.

- `ec2:DescribeVpcEndpoints` 권한에 대한 정책이 AWS Glue에 전달되었는지 확인합니다.

오류: DescribeRouteTables 작업이 승인되지 않았습니다. VPC ID: vpc-id에서 서브넷 ID: 서브넷-id를 검증할 수 없습니다.

- `ec2:DescribeRouteTables` 권한에 대한 정책이 AWS Glue에 전달되었는지 확인합니다.

오류: ec2:DescribeSubnets 호출이 실패했습니다.

- `ec2:DescribeSubnets` 권한에 대한 정책이 AWS Glue에 전달되었는지 확인합니다.

오류: ec2:DescribeSecurityGroups 호출이 실패했습니다.

- `ec2:DescribeSecurityGroups` 권한에 대한 정책이 AWS Glue에 전달되었는지 확인합니다.

오류: AZ용 서브넷을 찾을 수 없습니다.

- 이 가용 영역에서는 AWS Glue를 사용할 수 없습니다. 메시지에 있는 지정된 영역 중 새로운 서브넷을 다른 가용 영역에서 만들고 사용합니다.

오류: JDBC 대상으로 작성할 경우의 작업 실행 예외

JDBC 대상으로 작성하는 작업을 실행하고 있다면 작업은 다음과 같은 상황에서 오류가 발생합니다.

- 작업이 Microsoft SQL Server 테이블에 쓴다면 테이블은 `Boolean` 유형으로 정의된 열을 가지고 있고 SQL 서버 데이터베이스에서 미리 정의되어야 합니다. [데이터 대상에 테이블 생성(Create tables in your data target)] 옵션이 있는 SQL Server 대상을 사용하여 AWS Glue 콘솔에서 작업을 정의하는 경우 소스 열을 데이터 유형이 `Boolean`인 대상 열에 매핑하지 마세요. 작업이 실행되면 오류가 발생할 수 있습니다.

  다음을 수행하여 오류를 피합니다.

  - [Boolean(부울)]을 사용하여 기존 테이블을 선택합니다.
  - `ApplyMapping` 변환을 편집하고 소스의 [Boolean(부울)] 열을 대상의 숫자 또는 문자열로 매핑합니다.
  - `ApplyMapping` 변환을 편집하여 소스의 [Boolean(부울)] 열을 제거합니다.
- 작업이 Oracle 테이블에 쓴다면 Oracle 객체 이름 길이를 조정해야 합니다. Oracle의 몇 가지 버전에서는 최대 분류자 길이는 30바이트 또는 128바이트로 제한됩니다. 이 제한은 Oracle 대상 데이터 스토어의 테이블 이름과 열 이름에 영향을 줍니다.

  다음을 수행하여 오류를 피합니다.

  - 버전의 제한 범위 안에서 Oracle 대상 테이블 이름을 짓습니다.
  - 기본 열 이름은 데이터의 필드 이름에서 생성됩니다. 열 이름이 한도보다 긴 경우, `ApplyMapping` 또는 `RenameField` 변환을 사용하여 열 이름이 한도를 넘지 않도록 변경합니다.

오류: Amazon S3: 객체의 스토리지 클래스에 대해 작업이 유효하지 않습니다.

AWS Glue에서 이 오류가 반환되면 AWS Glue 작업이 Amazon S3 스토리지 클래스 계층에 걸쳐 파티션이 있는 테이블에서 데이터를 읽고 있는 것일 수 있습니다.

- 스토리지 클래스 제외를 사용하면 해당 스토리지 클래스 계층에서 파티션이 포함된 테이블에 대해 AWS Glue 작업을 실행할 수 있습니다. 제외하지 않으면 이러한 계층에서 데이터를 읽는 작업은 `AmazonS3Exception: The operation is not valid for the object's storage class` 오류와 함께 실패합니다.

  자세한 내용은 Amazon S3 스토리지 클래스 제외 단원을 참조하십시오.

오류: Amazon S3 시간 제한

AWS Glue가 연결 제한 시간 오류를 반환한다는 의미는 다른 AWS 리전에서 Amazon S3 버킷으로 액세스를 시도한다는 것입니다.

- Amazon S3 VPC 엔드포인트는 AWS 리전 내의 버킷으로만 트래픽을 라우팅할 수 있습니다. 버킷을 다른 지역으로 연결할 필요가 있다면 차선책으로 NAT 게이트웨이를 사용합니다. 자세한 내용은 NAT 게이트웨이 단원을 참조하세요.

오류: Amazon S3 액세스가 거부되었습니다.

AWS Glue가 액세스 거부 오류를 Amazon S3 버킷 또는 객체로 반환한다면 제공된 IAM 역할에 데이터 스토어에 대한 권한이 있는 정책이 없다는 뜻입니다.

- ETL 작업은 소스 또는 대상으로 사용되는 Amazon S3 데이터 스토어로 액세스할 수 있어야 합니다. 크롤러는 크롤하는 Amazon S3 데이터 스토어로 액세스할 수 있어야 합니다. 자세한 내용은 2단계: AWS Glue에 대한 IAM 역할 생성 단원을 참조하십시오.

오류: Amazon S3 액세스 키 ID가 없습니다.

AWS Glue가 작업을 실행하고 있을 때 액세스 키 ID가 존재하지 않는다는 오류를 반환한다면 다음 중 하나가 원인일 수 있습니다.

- ETL 작업은 IAM 역할을 사용하여 데이터 스토어로 액세스합니다. 작업 시작 전에 작업용 IAM 역할이 삭제된 것은 아닌지 확인합니다.
- IAM 역할은 데이터 스토어에 대한 액세스 권한을 가지고 있습니다. `s3:ListBucket`이 포함된 모든 연결된 Amazon S3 정책이 올바른지 확인합니다.

오류: **s3a://** URI를 사용해 Amazon S3에 액세스할 때 작업 실행이 실패합니다.

작업이 핸들러 클래스로 XML 문서 파싱에 실패 같은 오류를 반환한다면 `s3a://` URI를 사용하여 수백 개의 파일을 기록하는 데 실패했기 때문입니다. 대신에 `s3://` URI를 사용하여 데이터 스토어에 액세스합니다. 다음 예외 흔적은 다음과 같은 오류를 찾아봐야 합니다.

```
1. com.amazonaws.SdkClientException: Failed to parse XML document with handler class
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser$ListBucketHandler
2. at
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser.parseXmlInputStream(XmlRespons
3. at
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser.parseListBucketObjectsResponse
4. at com.amazonaws.services.s3.model.transform.Unmarshallers
$ListObjectsUnmarshaller.unmarshall(Unmarshallers.java:70)
5. at com.amazonaws.services.s3.model.transform.Unmarshallers
$ListObjectsUnmarshaller.unmarshall(Unmarshallers.java:59)
6. at
 com.amazonaws.services.s3.internal.S3XmlResponseHandler.handle(S3XmlResponseHandler.java:62)
7. at
 com.amazonaws.services.s3.internal.S3XmlResponseHandler.handle(S3XmlResponseHandler.java:31)
8. at
 com.amazonaws.http.response.AwsResponseHandlerAdapter.handle(AwsResponseHandlerAdapter.java:70
9. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.handleResponse(AmazonHttpClient.java:1554)
10. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeOneRequest(AmazonHttpClient.java:1272)
11. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeHelper(AmazonHttpClient.java:1056)
12. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.doExecute(AmazonHttpClient.java:743)
13. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeWithTimer(AmazonHttpClient.java:717)
14. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.execute(AmazonHttpClient.java:699)
```

```
15. at com.amazonaws.http.AmazonHttpClient$RequestExecutor.access
$500(AmazonHttpClient.java:667)
16. at com.amazonaws.http.AmazonHttpClient
$RequestExecutionBuilderImpl.execute(AmazonHttpClient.java:649)
17. at com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:513)
18. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4325)
19. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4272)
20. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4266)
21. at com.amazonaws.services.s3.AmazonS3Client.listObjects(AmazonS3Client.java:834)
22. at org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:971)
23. at
 org.apache.hadoop.fs.s3a.S3AFileSystem.deleteUnnecessaryFakeDirectories(S3AFileSystem.java:115
24. at org.apache.hadoop.fs.s3a.S3AFileSystem.finishedWrite(S3AFileSystem.java:1144)
25. at org.apache.hadoop.fs.s3a.S3AOutputStream.close(S3AOutputStream.java:142)
26. at org.apache.hadoop.fs.FSDataOutputStream
$PositionCache.close(FSDataOutputStream.java:74)
27. at org.apache.hadoop.fs.FSDataOutputStream.close(FSDataOutputStream.java:108)
28. at org.apache.parquet.hadoop.ParquetFileWriter.end(ParquetFileWriter.java:467)
29. at
 org.apache.parquet.hadoop.InternalParquetRecordWriter.close(InternalParquetRecordWriter.java:1
30. at
 org.apache.parquet.hadoop.ParquetRecordWriter.close(ParquetRecordWriter.java:112)
31. at
 org.apache.spark.sql.execution.datasources.parquet.ParquetOutputWriter.close(ParquetOutputWrit
32. at org.apache.spark.sql.execution.datasources.FileFormatWriter
$SingleDirectoryWriteTask.releaseResources(FileFormatWriter.scala:252)
33. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun
$org$apache$spark$sql$execution$datasources$FileFormatWriter$$executeTask
$3.apply(FileFormatWriter.scala:191)
34. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun
$org$apache$spark$sql$execution$datasources$FileFormatWriter$$executeTask
$3.apply(FileFormatWriter.scala:188)
35. at org.apache.spark.util.Utils
$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1341)
36. at org.apache.spark.sql.execution.datasources.FileFormatWriter$.org$apache$spark
$sql$execution$datasources$FileFormatWriter$$executeTask(FileFormatWriter.scala:193)
37. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun$write$1$
$anonfun$3.apply(FileFormatWriter.scala:129)
38. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun$write$1$
$anonfun$3.apply(FileFormatWriter.scala:128)
39. at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:87)
40. at org.apache.spark.scheduler.Task.run(Task.scala:99)
41. at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:282)
42. at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
```

```
43. at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
44. at java.lang.Thread.run(Thread.java:748)
```

오류: Amazon S3 서비스 토큰이 만료되었습니다.

Amazon Redshift 간에 데이터를 이동할 때에는 1시간 후에 만료되는 임시 Amazon S3 자격 증명이 사용됩니다. 따라서 실행 시간이 긴 작업은 실패할 수 있습니다. Amazon Redshift 간에 데이터를 이동하기 위해 오래 실행되는 작업을 설정하는 방법은 aws-glue-programming-etl-connect-redshift-home 섹션을 참조하세요.

오류: 네트워크 인터페이스용 프라이빗 DNS를 찾을 수 없습니다.

작업 또는 개발 엔드포인트가 할당하는 작업을 실패하면 네트워크 설정 문제일 것입니다.

- Amazon이 제공한 DNS를 사용 중이면 `enableDnsHostnames` 값은 true로 설정해야 합니다. 자세한 내용은 DNS를 참조하십시오.

오류: 개발 엔드포인트 프로비저닝이 실패했습니다.

AWS Glue가 성공적으로 개발 엔드포인트를 성공적으로 할당할 수 없다면 네트워크 설정 문제일 것입니다.

- 개발 엔드포인트를 정의하면 VPC, 서브넷 및 보안 그룹이 특정 요구 사항에 부합한다는 것을 확인하도록 검증할 수 있습니다.
- 조건부 SSH 퍼블릭 키를 제공했다면 유효한 SSH 퍼블릭 키인지 확인합니다.
- VPC가 유효한 [DHCP option set(DHCP 옵션 세트)]를 사용하도록 VPC 콘솔을 확인합니다. 자세한 내용은 DHCP 옵션 세트를 참조하십시오.
- 여전히 클러스터가 프로비저닝 상태로 유지된다면 AWS Support에 문의합니다.

오류: 노트북 서버 생성이 실패했습니다.

AWS Glue가 개발 엔드포인트의 노트북 서버 생성에 실패했다면 다음과 같은 문제 중 하나일 수 있습니다.

- AWS Glue는 노트북 서버를 설치할 때 IAM 역할을 Amazon EC2로 전달합니다. IAM 역할은 Amazon EC2의 신뢰 관계를 맺고 있어야 합니다.

- IAM 역할은 동일한 이름의 인스턴스 프로파일을 가지고 있어야 합니다. IAM 콘솔에서 Amazon EC2를 위한 역할을 생성할 때 동일한 이름의 인스턴스 프로파일이 자동으로 생성됩니다. 유효하지 않은 인스턴스 프로파일 이름 `iamInstanceProfile.name`에 따라 로그 오류를 확인합니다. 자세한 내용은 인스턴스 프로파일 사용 섹션을 참조하세요.
- 역할이 노트북 서버를 생성하기 위해 통과한 정책에서 `aws-glue*` 버킷에 액세스할 수 있는 권한이 있는지 확인합니다.

오류: 로컬 노트북 시작이 실패했습니다.

로컬 노트북이 시작을 못하고 디렉토리 혹은 폴더를 찾을 수 없다는 오류를 보고 하면 다음과 같은 문제 중 하나일 수 있습니다.

- Microsoft Windows에서 실행하고 있다면 `JAVA_HOME` 환경 변수가 올바른 자바 디텍터리를 가리키는지 확인하십시오. 이 변수를 업데이트하지 않고서 Java를 업데이트할 수 있습니다. 그리고 존재하지 않는 폴더를 가리키면 Jupyter 노트북은 시작하지 못합니다.

오류: 크롤러 실행이 실패했습니다.

AWS Glue가 크롤러를 성공적으로 카탈로그 데이터에서 실행하지 못하면 다음과 같은 원인 중 하나일 수 있습니다. 먼저 오류가 AWS Glue 콘솔 크롤러 목록에 있는지 확인합니다. 크롤러 이름 옆에 중요 표시 아이콘이 있는지 확인하고 관련 메시지가 아이콘 옆에 뜨는지 확인합니다.

- `/aws-glue/crawlers`의 CloudWatch Logs에서 크롤러 실행 로그를 확인합니다.

오류: 파티션이 업데이트되지 않았습니다.

ETL 작업을 실행했을 때 Data Catalog에서 파티션이 업데이트되지 않은 경우 CloudWatch Logs의 `DataSink` 클래스에 있는 다음 로그 문이 유용할 수 있습니다.

- `"Attempting to fast-forward updates to the Catalog - nameSpace:"` - 이 작업에서 수정을 시도한 데이터베이스, 테이블 및 catalogId를 표시합니다. 이 문이 여기에 없는 경우 `enableUpdateCatalog`가 true로 설정되어 있으며 `getSink()` 파라미터로 또는 `additional_options`로 제대로 전달되는지 확인하십시오.
- `"Schema change policy behavior:"` - 전달한 스키마 `updateBehavior` 값을 표시합니다.
- `"Schemas qualify (schema compare):"` - true 또는 false입니다.
- `"Schemas qualify (case-insensitive compare):"` - true 또는 false입니다.

- 둘 다 false이고 updateBehavior가 UPDATE_IN_DATABASE로 설정되어 있지 않은 경우 DynamicFrame 스키마는 Data Catalog 테이블 스키마에 있는 열의 하위 집합과 동일하거나 이를 포함해야 합니다.

파티션 업데이트에 대한 자세한 내용은 AWS Glue ETL 작업을 사용하여 데이터 카탈로그에서 스키마 업데이트 및 새 파티션 추가 섹션을 참조하세요.

오류: 버전 불일치로 인해 작업 북마크 업데이트 실패

Amazon S3의 다른 데이터 세트에 동일한 변환/로직을 적용하기 위해 AWS Glue 작업을 매개변수화하려고 할 수 있습니다. 제공된 위치에서 처리된 파일을 추적하고자 합니다. 동일한 소스 버킷에서 동일한 작업을 실행하고 동일한/다른 대상에 동시에 쓰는 경우(동시성 >1) 다음 오류와 함께 작업이 실패합니다.

```
py4j.protocol.Py4JJavaError: An error occurred while
 callingz:com.amazonaws.services.glue.util.Job.commit.:com.amazonaws.services.gluejobexecutor.m
 Continuation update failed due to version mismatch. Expected version 2 but found
 version 3
```

솔루션: 동시성을 1로 설정하거나 작업을 동시에 실행하지 마십시오.

현재 AWS Glue 북마크는 동시 작업 실행을 지원하지 않으며 커밋이 실패합니다.

오류: 작업 북마크가 사용 설정된 경우 작업이 데이터를 재처리합니다.

AWS Glue 작업 북마크를 활성화했는데도 ETL 작업이 앞선 실행에서 이미 처리된 데이터를 재처리하는 경우가 있을 수 있습니다. 이 오류의 일반적인 원인을 확인해 보십시오.

최대 동시성

작업의 최대 동시 실행 횟수를 기본값인 1보다 크게 설정하면 작업 북마크를 방해할 수 있습니다. 이는 작업 북마크가 객체의 마지막 수정 시간을 확인하여 재처리가 필요한 객체를 확인할 때 발생할 수 있습니다. 자세한 내용은 AWS Glue에서 Spark 작업에 대한 작업 속성 구성에서 최대 동시성 관련 내용을 참조하십시오.

작업 객체 누락

작업 실행 스크립트는 다음 커밋으로 끝나야 합니다.

```
job.commit()
```

이 객체를 포함한 경우 AWS Glue에서는 작업 실행의 타임스탬프 및 경로를 기록합니다. 동일한 경로로 작업을 다시 실행하면 AWS Glue에서는 새 파일만 처리합니다. 이 객체를 포함하지 않고 작업 북마크를 활성화한 경우, 이 작업은 새 파일과 함께 이미 처리된 파일도 재처리하고 작업의 대상 데이터 스토어에 중복 항목을 생성합니다.

변환 컨텍스트 파라미터 누락

변환 컨텍스트가 `GlueContext` 클래스의 선택적 파라미터이기는 하지만, 이 파라미터를 넣지 않으면 작업 북마크가 제대로 작동하지 않습니다. 이 오류를 해결하려면 DynamicFrame을 생성할 때 다음과 같이 변환 컨텍스트 파라미터를 추가합니다.

```
sample_dynF=create_dynamic_frame_from_catalog(database,
 table_name,transformation_ctx="sample_dynF")
```

입력 소스

입력 소스에 대해 관계형 데이터베이스(JDBC 연결)를 사용하는 경우 테이블의 기본 키가 순차적인 경우에만 작업 북마크가 작동합니다. 작업 북마크는 업데이트된 행이 아니라 새 행에 대해 작동합니다. 이는 작업 북마크가 이미 있는 기본 키를 검색하기 때문입니다. 입력 소스가 Amazon Simple Storage Service(Amazon S3)인 경우에는 적용되지 않습니다.

최근 수정 시간

Amazon S3 입력 소스의 경우 작업 북마크는 파일 이름이 아니라 객체의 마지막으로 수정된 시간을 검사해 어떤 객체를 재처리해야 하는지 확인합니다. 마지막 작업 실행 이후로 입력 소스 데이터가 수정된 경우, 작업을 다시 실행하면 그 파일이 재처리됩니다.

오류: AWS Glue에서 VPC 간 장애 조치 동작

AWS Glue 4.0 및 이전 버전에서 작업의 장애 조치에 대해 다음 프로세스가 사용됩니다.

요약: AWS Glue 연결은 작업 실행이 제출될 때 선택됩니다. 작업 실행 시 일부 문제(IP 주소 부족, 소스 연결, 라우팅 문제)가 발생하면 작업 실행이 실패합니다. 재시도가 구성된 경우 AWS Glue는 동일한 연결로 재시도합니다.

1. 각 실행 시도에 대해 AWS Glue는 사용할 수 있는 연결 상태를 찾을 때까지 작업 구성에 나열된 순서대로 연결 상태를 확인합니다. 가용 영역(AZ)에 장애가 발생하는 경우 해당 AZ의 연결에서 확인에 실패하고 건너뜁니다.
2. AWS Glue는 다음을 사용하여 연결을 검증합니다.
   - 유효한 Amazon VPC ID 및 서브넷을 확인합니다.

- NAT 게이트웨이 또는 Amazon VPC 엔드포인트가 존재하는지 확인합니다.
- 서브넷에 할당된 IP 주소가 1개 이상인지 확인합니다.
- AZ가 정상 상태인지 확인합니다.

AWS Glue는 작업 실행 제출 시에는 연결을 확인할 수 없습니다.

3. Amazon VPC를 사용하는 작업의 경우, 모든 드라이버와 실행기는 작업 실행 제출 시 선택한 연결을 사용하여 동일한 AZ에서 생성됩니다.
4. 재시도가 구성된 경우 AWS Glue는 동일한 연결로 재시도합니다. 이 연결과 관련된 문제가 오래 지속된다고 보장할 수 없기 때문입니다. AZ에 장애가 발생하면 해당 AZ의 기존 작업 실행(작업 실행 단계에 따라 다름)이 실패할 수 있습니다. 재시도를 통해 AZ 장애를 감지하고 새 실행을 위해 다른 AZ를 선택해야 합니다.

크롤러가 Lake Formation 권한을 사용하는 경우 발생하는 크롤러 오류

아래 정보를 사용하여 Lake Formation 보안 인증을 사용하여 크롤러를 구성하는 동안 다양한 문제를 진단하고 수정합니다.

오류: The S3 location: s3://examplepath is not registered

Lake Formation 보안 인증을 사용하여 크롤러를 실행하려면 먼저 Lake Formation 권한을 설정해야 합니다. 이 오류를 해결하려면 Lake Formation에 대상 Amazon S3 위치를 등록하세요. 자세한 내용을 알아보려면 Registering an Amazon S3 location(Amazon S3 위치 등록)을 참조하세요.

오류: User/Role is not authorized to perform: lakeformation:GetDataAccess on resource

IAM 콘솔 또는 AWS CLI을(를) 사용하여 크롤러 역할에 대해 `lakeformation:GetDataAccess` 권한을 추가하세요. 이 권한을 통해 Lake Formation은 데이터에 액세스하기 위한 임시 보안 인증 요청을 승인합니다. 아래 정책을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
```

```
      "lakeformation:GetDataAccess"
    ],
    "Resource": "*"
  }
}
```

오류: Insufficient Lake Formation permission(s) on (Database name: exampleDatabase, Table Name: exampleTable)

Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)에서 데이터베이스에 대한 크롤러 역할 액세스 권한(`Create`, `Describe`, `Alter`)을 부여합니다. 이는 출력 데이터베이스로 지정됩니다. 테이블에 대한 권한도 부여할 수 있습니다. 자세한 내용을 알아보려면 명명된 리소스 메서드를 사용하여 데이터베이스 권한 부여를 참조하세요.

오류: Insufficient Lake Formation permission(s) on s3://examplepath

1. 교차 계정 크롤링
   a. Amazon S3 버킷이 등록된 계정(계정 B)을 사용하여 Lake Formation 콘솔(https://console.aws.amazon.com/lakeformation/)에 로그인합니다. 크롤러가 실행될 계정에 데이터 위치 권한을 부여합니다. 이렇게 하면 크롤러가 대상 Amazon S3 위치에서 데이터를 읽을 수 있습니다.
   b. 크롤러가 생성된 계정(계정 A)에서 크롤러 실행에 사용되는 IAM 역할에 대상 Amazon S3 위치에 대한 데이터 위치 권한을 부여하여 크롤러가 Lake Formation의 대상에서 데이터를 읽을 수 있도록 합니다. 자세한 내용을 알아보려면 데이터 위치 권한 부여(외부 계정)를 참조하세요.
2. 계정 내(크롤러 및 등록된 Amazon S3 위치가 동일한 계정에 있음) 크롤링 - Amazon S3 위치에서 크롤러 실행에 사용되는 IAM 역할에 데이터 위치 권한을 부여하여 크롤러가 Lake Formation의 대상에서 데이터를 읽을 수 있도록 합니다. 자세한 내용을 알아보려면 Granting data location permissions (same account)(데이터 위치 권한 부여(동일한 계정))를 참조하세요.

Lake Formation 보안 인증을 사용한 크롤러 구성에 대해 자주 묻는 질문

1. AWS 콘솔을 사용하여 Lake Formation 보안 인증을 사용하여 실행되도록 크롤러를 구성하려면 어떻게 해야 합니까?

   AWS Glue 콘솔(https://console.aws.amazon.com/glue/)에서 크롤러를 구성하는 동안 Use Lake Formation credentials for crawling Amazon S3 data source(Amazon S3 데이터 소스 크롤링에 Lake Formation 자격 증명 사용) 옵션을 선택합니다. 크로스 계정 크롤링의 경우 대상 Amazon S3 위치가

Lake Formation에 등록된 AWS 계정 ID를 지정합니다. 계정 내 크롤링의 경우 accountId 필드는 선택 사항입니다.

2. AWS CLI을(를) 사용하여 Lake Formation 보안 인증을 사용하여 실행되도록 크롤러를 구성하려면 어떻게 해야 합니까?

   `CreateCrawler` API 직접 호출 중에 `LakeFormationConfiguration` 추가:

   ```
   "LakeFormationConfiguration": {
       "UseLakeFormationCredentials": true,
       "AccountId": "111111111111" (AWS account ID where the target Amazon S3 location
    is registered with Lake Formation)
     }
   ```

3. Lake Formation 보안 인증을 사용하는 크롤러에 대해 지원되는 대상은 무엇입니까?

   Lake Formation 보안 인증을 사용하는 크롤러는 Amazon S3(계정 내 및 크로스 계정 크롤링), 계정 내 데이터 카탈로그 대상(기본 위치가 Amazon S3인 경우), Apache Iceberg 대상에 대해서만 지원됩니다.

4. Lake Formation 보안 인증을 사용하여 단일 크롤러의 일부로 여러 Amazon S3 버킷을 크롤링할 수 있습니까?

   아니요, Lake Formation 보안 인증 벤딩을 사용하는 크롤링 대상의 경우 기본 Amazon S3 위치가 동일한 버킷에 속해야 합니다. 예를 들어, 고객은 동일한 버킷(버킷 1) 아래에 있는 경우 여러 대상 위치 (`s3://bucket1/folder1, s3://bucket1/folder2`)을(를) 사용할 수 있습니다. 다른 버킷 지정(s3://bucket1/folder1, s3://bucket2/folder2)은 지원되지 않습니다.

로그에서 AWS Glue for Ray 오류 해결

AWS Glue를 사용하면 작업 실행 중에 Ray 프로세스에서 생성된 로그에 액세스할 수 있습니다. Ray 작업에서 오류나 예상치 못한 동작이 발생하는 경우 먼저 로그에서 정보를 수집하여 오류의 원인을 파악합니다. 대화형 세션에 대해서도 유사한 로그를 제공합니다. 세션 로그는 `/aws-glue/ray/sessions` 접두사와 함께 제공됩니다.

작업이 실행되면 로그 라인이 CloudWatch에 실시간으로 전송됩니다. 실행을 완료한 후 인쇄 명령문이 CloudWatch 로그에 추가됩니다. 로그는 작업 실행 후 2주 동안 보관됩니다.

Ray 작업 로그 검사

작업이 실패하면 작업 이름과 작업 실행 ID를 수집합니다. 이러한 항목은 AWS Glue 콘솔에서 찾을 수 있습니다. 작업 페이지로 이동한 다음 Runs(실행) 탭으로 이동합니다. Ray 작업 로그는 다음과 같은 전용 CloudWatch 로그 그룹에 저장됩니다.

- `/aws-glue/ray/jobs/script-log/` - 기본 Ray 스크립트에서 내보낸 로그를 저장합니다.
- `/aws-glue/ray/jobs/ray-monitor-log/` - Ray autoscaler 프로세스에서 내보낸 로그를 저장합니다. 이러한 로그는 헤드 노드에 대해 생성되며 다른 워커 노드에 대해서는 생성되지 않습니다.
- `/aws-glue/ray/jobs/ray-gcs-logs/` - 글로벌 제어 스토어(GCS) 프로세스에서 생성된 로그를 저장합니다. 이러한 로그는 헤드 노드에 대해 생성되며 다른 워커 노드에 대해서는 생성되지 않습니다.
- `/aws-glue/ray/jobs/ray-process-logs/` - 헤드 노드에서 실행되는 다른 Ray 프로세스(주로 대시보드 에이전트)에서 생성된 로그를 저장합니다. 이러한 로그는 헤드 노드에 대해 생성되며 다른 워커 노드에 대해서는 생성되지 않습니다.
- `/aws-glue/ray/jobs/ray-raylet-logs/` - 각 raylet 프로세스에서 생성된 로그를 저장합니다. 이러한 로그는 헤드 노드를 포함하여 각 워커 노드에 대해 단일 스트림으로 수집됩니다.
- `/aws-glue/ray/jobs/ray-worker-out-logs/` - 클러스터의 각 작업자에 대한 `stdout` 로그를 저장합니다. 이러한 로그는 헤드 노드를 포함한 각 워커 노드에서 생성됩니다.
- `/aws-glue/ray/jobs/ray-worker-err-logs/` - 클러스터의 각 작업자에 대한 `stderr` 로그를 저장합니다. 이러한 로그는 헤드 노드를 포함한 각 워커 노드에서 생성됩니다.
- `/aws-glue/ray/jobs/ray-runtime-env-log/` - Ray 설정 프로세스에 대한 로그를 저장합니다. 이러한 로그는 헤드 노드를 포함한 각 워커 노드에서 생성됩니다.

Ray 작업 오류 해결

Ray 로그 그룹의 구성을 이해하고 오류 해결에 도움이 되는 로그 그룹을 찾으려면 Ray 아키텍처에 대한 배경 정보를 알고 있으면 도움이 됩니다.

AWS Glue ETL에서 작업자는 인스턴스에 해당합니다. AWS Glue 작업에 대한 작업자를 구성할 때 해당 작업 전용 인스턴스의 유형과 수량을 설정합니다. Ray에서는 작업자라는 용어를 다양한 방식으로 사용합니다.

Ray에서는 헤드 노드와 워커커 노드를 사용하여 Ray 클러스터 내에서 인스턴스의 책임을 구분합니다. Ray 워커 노드는 분산 계산 결과를 얻기 위해 계산을 수행하는 여러 액터 프로세스를 호스팅할 수 있습니다. 함수의 복제본을 실행하는 액터를 복제본이라고 합니다. 복제본 액터를 작업자 프로세스라고도

합니다. 복제본은 클러스터를 조정하기 위한 추가 프로세스를 실행하기 때문에 헤드라고 하는 헤드 노드에서도 실행될 수 있습니다.

계산에 기여하는 각 액터는 자체 로그 스트림을 생성합니다. 이를 통해 다음과 같은 몇 가지 인사이트를 얻을 수 있습니다.

- 로그를 내보내는 프로세스 수가 작업에 할당된 작업자 수보다 많을 수 있습니다. 각 인스턴스의 각 코어에는 주로 액터가 있습니다.
- Ray 헤드 노드는 클러스터 관리 및 시작 로그를 내보냅니다. 반면 Ray 워커 노드는 수행된 작업에 대한 로그만 내보냅니다.

Ray 아키텍처에 대한 자세한 내용은 Ray 설명서의 아키텍처 백서를 참조하세요.

문제 영역: Amazon S3 액세스

작업 실행 실패 메시지를 확인합니다. 충분한 정보가 제공되지 않는 경우 `/aws-glue/ray/jobs/script-log/` 섹션을 확인하세요.

문제 영역: PIP 종속성 관리

`/aws-glue/ray/jobs/ray-runtime-env-log/`을 검토합니다.

문제 영역: 메인 프로세스의 중간 값 검사

기본 스크립트에서 `stderr` 또는 `stdout`에 쓰고 `/aws-glue/ray/jobs/script-log/`에서 로그를 검색합니다.

문제 영역: 하위 프로세스의 중간 값 검사

`remote` 함수에서 `stderr` 또는 `stdout`에 씁니다. 그런 다음 `/aws-glue/ray/jobs/ray-worker-out-logs/` 또는 `/aws-glue/ray/jobs/ray-worker-err-logs/`에서 로그를 검색합니다. 함수가 복제본에서 실행되었을 수 있으므로 원하는 출력을 찾기 위해 여러 로그를 검사해야 할 수 있습니다.

문제가 있는 영역: 오류 메시지에서 IP 주소 해석

특정 오류 상황에서는 작업에서 IP 주소가 포함된 오류 메시지를 생성할 수 있습니다. 이러한 IP 주소는 임시 주소이며 클러스터에서 노드를 식별하고 노드 사이에서 통신하는 데 사용됩니다. 노드의 로그는 IP 주소를 기반으로 고유한 접미사를 사용하는 로그 스트림에 게시됩니다.

CloudWatch에서는 이 접미사를 식별하여 로그를 필터링해 이 IP 주소에 특정한 로그를 검사할 수 있습니다. 예를 들어 주어진 *FAILED_IP* 및 *JOB_RUN_ID*에 대해 다음을 사용해 접미사를 식별할 수 있습니다.

```
filter @logStream like /JOB_RUN_ID/
| filter @message like /IP-/
| parse @message "IP-[*]" as ip
| filter ip like /FAILED_IP/
| fields replace(ip, ":", "_") as uIP
| stats count_distinct by uIP as logStreamSuffix
| display logStreamSuffix
```

AWS Glue 기계 학습 예외 사항

이 주제에서는 기계 학습과 관련된 AWS Glue 예외에 대한 HTTP 오류 코드 및 문자열에 대해 설명합니다. 작업을 수행할 때 발생할 수 있는 각 기계 학습 작업에 대해 오류 코드 및 오류 문자열이 제공됩니다. 또한 오류를 일으킨 작업을 다시 시도할 수 있는지 여부를 확인할 수 있습니다.

CancelMLTaskRunActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”
  - “[transformName] 변환을 위한 [accountId] 계정에서 [taskRunId]에 대한 ML 작업 실행을 찾을 수 없습니다.”

  재시도 가능: 아니요.

CreateMLTaskRunActivity

이 활동에는 다음과 같은 예외가 있습니다.

- InvalidInputException (400)
  - “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”
  - “변환에서 AWS Glue 테이블 입력 소스를 지정해야 합니다.”
  - “입력 소스 열 [columnName]에 카탈로그에 정의된 잘못된 데이터 형식이 있습니다.”

- “정확히 하나의 입력 레코드 테이블을 제공해야 합니다.”
- “데이터베이스 이름을 지정해야 합니다.”
- “테이블 이름을 지정해야 합니다.”
- “스키마가 변환에 정의되지 않았습니다.”
- “스키마는 주어진 기본 키 [primaryKey]를 포함해야 합니다.”
- “데이터 카탈로그 스키마를 가져오는 중 문제가 발생했습니다. [message].”
- “최대 용량과 작업자 수/유형을 동시에 설정할 수 없습니다.”
- “WorkerType과 NumberOfWorkers 둘 다 설정해야 합니다.”
- “MaxCapacity는 >=[maxCapacity]이어야 합니다.”
- “NumberOfWorkers는 >=[maxCapacity]이어야 합니다.”
- “최대 재시도는 음수가 아니어야 합니다.”
- “일치 항목 찾기 파라미터가 설정되지 않았습니다.”
- “일치 항목 찾기 파라미터에 기본 키를 지정해야 합니다.”

재시도 가능: 아니요.

- AlreadyExistsException (400)
  - “이름이 [transformName]인 변형이 이미 있습니다.”

  재시도 가능: 아니요.
- IdempotentParameterMismatchException (400)
  - “[transformName] 변환에 대한 멱등성 생성 요청에 일치하지 않는 파라미터가 있습니다.”

  재시도 가능: 아니요.
- InternalServiceException (500)
  - “종속성 오류가 발생했습니다.”

  재시도 가능: 예.
- ResourceNumberLimitExceededException (400)
  - “ML 변환 수([count])가 [limit] 변환의 제한을 초과했습니다.”

  재시도 가능: 예, 이 새 변환을 위한 공간을 만들기 위해 변환을 삭제한 후 재시도합니다.

DeleteMLTransformActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.

GetMLTaskRunActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”
  - “[transformName] 변환을 위한 [accountId] 계정에서 [taskRunId]에 대한 ML 작업 실행을 찾을 수 없습니다.”

  재시도 가능: 아니요.

GetMLTaskRunsActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”
  - “[transformName] 변환을 위한 [accountId] 계정에서 [taskRunId]에 대한 ML 작업 실행을 찾을 수 없습니다.”

  재시도 가능: 아니요.

GetMLTransformActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

재시도 가능: 아니요.

GetMLTransformsActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “계정 ID는 비워 둘 수 없습니다.”
  - “[column] 열에는 정렬이 지원되지 않습니다.”
  - “[column]은(는) 비워 둘 수 없습니다.”
  - “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”

  재시도 가능: 아니요.

GetSaveLocationForTransformArtifactActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “지원되지 않는 아티팩트 유형 [artifactType]입니다.”
  - “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”

  재시도 가능: 아니요.

GetTaskRunArtifactActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”
  - “[transformName] 변환을 위한 [accountId] 계정에서 [taskRunId]에 대한 ML 작업 실행을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “파일 이름 '[fileName]’이(가) 게시에 유효하지 않습니다.”
  - “[taskType] 작업 유형에 대한 아티팩트를 검색할 수 없습니다.”
  - “[artifactType]에 대한 아티팩트를 검색할 수 없습니다.”
  - “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”

  재시도 가능: 아니요.

PublishMLTransformModelActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”
  - “계정 ID - [accountId] - 및 변환 ID - [transformId]에서 버전 - [version] 버전이 있는 기존 모델을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “파일 이름 '[fileName]’이(가) 게시에 유효하지 않습니다.”
  - “부호 없는 문자열 [string] 앞에 잘못된 마이너스 부호가 있습니다.”
  - “[string]의 끝에 잘못된 숫자가 있습니다.”
  - “문자열 값 [string]은(는) 부호 없는 long 범위를 초과합니다.”
  - “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”

  재시도 가능: 아니요.

PullLatestMLTransformModelActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”

  재시도 가능: 아니요.
- ConcurrentModificationException (400)
  - “파라미터가 일치하지 않는 레이싱 입력으로 인해 훈련할 모델 버전을 만들 수 없습니다.”
  - “변환 ID [transformId]에 대한 ML 변환 모델이 오래되었거나 다른 프로세스에 의해 업데이트되고 있습니다. 다시 시도하십시오.”

  재시도 가능: 예.

PutJobMetadataForMLTransformActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”
  - “[transformName] 변환을 위한 [accountId] 계정에서 [taskRunId]에 대한 ML 작업 실행을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”
  - “알 수 없는 작업 메타데이터 유형 [jobType]입니다.”
  - “업데이트할 작업 실행 ID를 제공해야 합니다.”

  재시도 가능: 아니요.

StartExportLabelsTaskRunActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”
  - “계정 ID [accountId]에 변환 ID [transformId]에 대한 labelset가 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “[message].”
  - “제공된 S3 경로가 변환과 동일한 리전에 있지 않습니다. 예상한 리전은 [region]이지만 받은 리전은 [region]입니다.”

  재시도 가능: 아니요.

StartImportLabelsTaskRunActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “[message].”
  - “잘못된 레이블 파일 경로입니다.”
  - “[labelPath]에서 레이블 파일에 액세스할 수 없습니다. [message].”
  - “변환에 제공된 IAM 역할을 사용할 수 없습니다. 역할: [role]입니다.”
  - “크기가 0인 잘못된 레이블 파일입니다.”
  - “제공된 S3 경로가 변환과 동일한 리전에 있지 않습니다. 예상한 리전은 [region]이지만 받은 리전은 [region]입니다.”

  재시도 가능: 아니요.
- ResourceNumberLimitExceededException (400)

- “레이블 파일이 [limit]MB 제한을 초과했습니다.”

재시도 가능: 아니오. 레이블 파일을 여러 개의 작은 파일로 나누는 것이 좋습니다.

StartMLEvaluationTaskRunActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “정확히 하나의 입력 레코드 테이블을 제공해야 합니다.”
  - “데이터베이스 이름을 지정해야 합니다.”
  - “테이블 이름을 지정해야 합니다.”
  - “일치 항목 찾기 파라미터가 설정되지 않았습니다.”
  - “일치 항목 찾기 파라미터에 기본 키를 지정해야 합니다.”

  재시도 가능: 아니요.
- MLTransformNotReadyException (400)
  - “이 작업은 준비 상태에 있는 변환에만 적용할 수 있습니다.”

  재시도 가능: 아니요.
- InternalServiceException (500)
  - “종속성 오류가 발생했습니다.”

  재시도 가능: 예.
- ConcurrentRunsExceededException (400)
  - “ML 작업 실행 수 [count]이(가) 작업 실행 [limit]의 변환 제한을 초과했습니다.”
  - “ML 작업 실행 수 [count]이(가) 작업 실행 [limit]의 제한을 초과했습니다.”

  재시도 가능: 예, 작업 실행이 완료될 때까지 기다린 후 재시도합니다.

StartMLLabelingSetGenerationTaskRunActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “정확히 하나의 입력 레코드 테이블을 제공해야 합니다.”
  - “데이터베이스 이름을 지정해야 합니다.”
  - “테이블 이름을 지정해야 합니다.”
  - “일치 항목 찾기 파라미터가 설정되지 않았습니다.”
  - “일치 항목 찾기 파라미터에 기본 키를 지정해야 합니다.”

  재시도 가능: 아니요.
- InternalServiceException (500)
  - “종속성 오류가 발생했습니다.”

  재시도 가능: 예.
- ConcurrentRunsExceededException (400)
  - “ML 작업 실행 수 [count]이(가) 작업 실행 [limit]의 변환 제한을 초과했습니다.”

  재시도 가능: 예, 작업 실행이 완료된 후 재시도합니다.

UpdateMLTransformActivity

이 활동에는 다음과 같은 예외가 있습니다.

- EntityNotFoundException (400)
  - “[transformName] 핸들이 있는 [accountId] 계정에서 MLTransform을 찾을 수 없습니다.”

  재시도 가능: 아니요.
- InvalidInputException (400)
  - “이름이 [transformName]인 다른 변환이 이미 있습니다.”
  - “[message].”

- “변환 이름은 비워 둘 수 없습니다.”
- “최대 용량과 작업자 수/유형을 동시에 설정할 수 없습니다.”
- “WorkerType과 NumberOfWorkers 둘 다 설정해야 합니다.”
- “MaxCapacity는 >=[minMaxCapacity]이어야 합니다.”
- “NumberOfWorkers는 >=[minNumWorkers]이어야 합니다.”
- “최대 재시도는 음수가 아니어야 합니다.”
- “예기치 않은 입력으로 인해 내부 서비스 오류가 발생했습니다.”
- “일치 항목 찾기 파라미터가 설정되지 않았습니다.”
- “일치 항목 찾기 파라미터에 기본 키를 지정해야 합니다.”

재시도 가능: 아니요.

- AlreadyExistsException (400)
  - “이름이 [transformName]인 변형이 이미 있습니다.”

  재시도 가능: 아니요.
- IdempotentParameterMismatchException (400)
  - “[transformName] 변환에 대한 멱등성 생성 요청에 일치하지 않는 파라미터가 있습니다.”

  재시도 가능: 아니요.

AWS Glue 할당량

AWS Support에 연락하여 AWS 일반 참조에 나열된 서비스 할당량에 대해 할당량 증가를 요청할 수 있습니다. 다르게 표시되지 않는 한 리전별로 각 할당량이 적용됩니다. 자세한 내용은 AWS Glue 엔드포인트 및 할당량을 참조하세요.

AWS Glue 성능 개선

성능 조정을 위한 기준선 전략

AWS Glue 성능을 개선하려면 특정 성능 관련 AWS Glue 파라미터를 업데이트하는 것을 고려할 수 있습니다. 파라미터를 조정할 준비를 할 때는 다음과 같은 모범 사례를 고려하세요.

- 문제를 식별하기 전에 성능 목표를 결정합니다.
- 조정 파라미터를 변경하기 전에 지표를 사용하여 문제를 식별합니다.

작업을 조정할 때 가장 일관된 결과를 얻으려면 조정 작업에 대한 기준선 전략을 수립하세요.

일반적으로 성능 조정은 다음 워크플로우에서 수행됩니다.

1. 성능 목표를 결정합니다.
2. 지표를 측정합니다.
3. 병목 현상을 식별합니다.
4. 병목 현상이 미치는 영향을 줄입니다.
5. 의도한 목표를 달성할 때까지 2~4단계를 반복합니다.

작업 유형에 맞는 튜닝 전략

Spark 작업 - AWS 규범적 지침의 Apache Spark 작업을 위한 AWS Glue 성능 튜닝 모범 사례의 지침을 따르세요.

기타 작업 - 다른 런타임 환경에서 사용할 수 있는 전략을 적용하여 Ray를 위한 AWS Glue 및 AWS Glue Python 쉘 작업을 튜닝할 수 있습니다.

Apache Spark용 AWS Glue 작업의 성능 개선

Spark용 AWS Glue의 성능을 개선하려면 특정 성능 관련 AWS Glue 및 Spark 파라미터를 업데이트하는 것을 고려할 수 있습니다.

지표를 통해 병목 현상을 식별하고 그 영향을 줄이기 위한 구체적인 전략에 대한 자세한 내용은 AWS 권장 가이드의 Apache Spark용 AWS Glue 작업 성능 조정 모범 사례를 참조하세요. 이 가이드에서는 Spark 아키텍처, 탄력적 분산 데이터 세트 등 모든 런타임 환경에서 Apache Spark에 적용할 수 있는

주요 주제를 소개합니다. 설명서에서는 이러한 주제를 사용하여 셔플 최적화 및 작업 병렬화와 같은 특정 성능 튜닝 전략을 구현하도록 안내합니다.

Spark UI를 표시하도록 AWS Glue를 구성하여 병목 현상을 식별할 수 있습니다. 자세한 내용은 the section called “Spark UI로 모니터링” 단원을 참조하십시오.

또한 AWS Glue는 작업이 연결되는 특정 유형의 데이터 스토어에 적용할 수 있는 성능 특성을 제공합니다. 데이터 스토어의 성능 파라미터에 대한 참조 정보는 the section called “연결 파라미터”에서 찾을 수 있습니다.

AWS Glue ETL에서 푸시다운을 사용한 읽기 최적화

푸시다운은 데이터 검색에 대한 로직을 데이터 소스에 더 가깝게 푸시하는 최적화 기법입니다. 소스는 Amazon S3와 같은 파일 시스템 또는 데이터베이스일 수 있습니다. 소스에서 직접 특정 작업을 실행하는 경우 네트워크를 통해 AWS Glue에서 관리하는 Spark 엔진으로 모든 데이터를 가져오지 않아도 되므로 시간과 처리 용량을 절감할 수 있습니다.

이를 다른 방법으로 설명하자면 푸시다운이 데이터 스캔을 줄이는 것입니다. 이 기법이 적절한 시기를 식별하는 프로세스에 대한 자세한 내용은 AWS 권장 가이드의 Apache Spark용 AWS Glue 작업 성능 조정 모범 사례 가이드에 있는 데이터 스캔량 감소를 참조하세요.

Amazon S3에 저장된 파일에서 조건자 푸시다운

Amazon S3에서 접두사를 기준으로 구성된 파일을 사용하는 경우 푸시다운 조건자를 정의하여 대상 Amazon S3 경로를 필터링할 수 있습니다. 전체 데이터 세트를 읽고 `DynamicFrame` 내에서 필터를 적용하는 대신 AWS Glue 데이터 카탈로그에 저장된 파티션 메타데이터에 필터를 직접 적용할 수 있습니다. 이 방법을 사용하면 필요한 데이터만 선택적으로 나열하고 읽을 수 있습니다. 파티션별 버킷 쓰기를 포함하여 이 프로세스에 대한 자세한 내용은 the section called “파티션 관리” 섹션을 참조하세요.

Amazon S3에서 `push_down_predicate` 파라미터를 사용하여 조건부 푸시다운을 수행합니다. Amazon S3의 버킷을 연도, 월, 일을 기준으로 파티셔닝했다고 가정합니다. 2022년 6월의 고객 데이터를 검색하려는 경우 관련 Amazon S3 경로만 읽도록 AWS Glue에 지시할 수 있습니다. 이 경우 `push_down_predicate`는 `year='2022' and month='06'`입니다. 따라서 다음과 같이 읽기 작업을 수행할 수 있습니다.

Python

```
customer_records = glueContext.create_dynamic_frame.from_catalog(
    database = "customer_db",
    table_name = "customer_tbl",
```

```
    push_down_predicate = "year='2022' and month='06'"
)
```

Scala

```
val customer_records = glueContext.getCatalogSource(
database="customer_db",
tableName="customer_tbl",
pushDownPredicate="year='2022' and month='06'"
).getDynamicFrame()
```

이전 시나리오에서 push_down_predicate는 AWS Glue 데이터 카탈로그의 모든 파티션 목록을 검색하고 기본 Amazon S3 파일을 읽기 전에 파티션을 필터링합니다. 대부분의 경우 이 방법이 유용하지만 수백만 개의 파티션이 있는 데이터 세트를 작업할 때는 파티션을 나열하는 데 시간이 많이 걸릴 수 있습니다. 이 문제를 해결하기 위해 서버 측 파티션 정리를 사용하여 성능을 개선할 수 있습니다. AWS Glue 데이터 카탈로그에서 데이터에 대한 파티션 인덱스를 구축하면 됩니다. 파티션 인덱스에 대한 자세한 내용은 the section called “파티션 인덱스 생성” 섹션을 참조하세요. 그런 다음 `catalogPartitionPredicate` 옵션을 사용하여 인덱스를 참조할 수 있습니다. `catalogPartitionPredicate`를 사용하여 파티션을 검색하는 예제는 the section called “카탈로그 파티션 조건자” 섹션을 참조하세요.

JDBC 소스를 사용할 때 푸시다운

`GlueContext`에서 사용되는 AWS Glue JDBC 리더는 소스에서 직접 실행할 수 있는 사용자 지정 SQL 쿼리를 제공하여 지원되는 데이터베이스에서의 푸시다운을 지원합니다. `sampleQuery` 파라미터를 설정하여 이 작업을 수행할 수 있습니다. 샘플 쿼리는 선택할 열을 지정할 수 있을 뿐만 아니라 Spark 엔진으로 전송되는 데이터를 제한하는 푸시다운 조건자를 제공할 수 있습니다.

기본적으로 샘플 쿼리는 단일 노드에서 작동하므로 대량의 데이터를 처리할 때 작업에 실패할 수 있습니다. 이 기능을 사용하여 대규모로 데이터를 쿼리하려면 `enablePartitioningForSampleQuery`를 true로 설정하여 쿼리 파티셔닝을 구성해야 합니다. 그러면 선택한 키의 여러 노드에 쿼리가 배포됩니다. 쿼리 파티셔닝에는 몇 가지 다른 필수 구성 파라미터도 필요합니다. 쿼리 파티셔닝에 대한 자세한 내용은 the section called “JDBC를 병렬로 읽기” 섹션을 참조하세요.

`enablePartitioningForSampleQuery`를 설정하면 AWS Glue는 데이터베이스를 쿼리할 때 푸시다운 조건자를 파티셔닝 조건자와 결합합니다. AWS Glue에서 파티션 조건을 추가하려면 `sampleQuery`는 `AND`로 끝나야 합니다. (푸시다운 조건자를 제공하지 않는 경우 `sampleQuery`는

WHERE로 끝나야 합니다.) id가 1,000보다 큰 행만 검색하도록 조건자를 푸시하는 아래 예제를 참조하세요. 이 sampleQuery에서는 id가 지정된 값보다 큰 행의 이름 및 위치 열만 반환합니다.

Python

```
sample_query = "select name, location from customer_tbl WHERE id>=1000 AND"
customer_records = glueContext.create_dynamic_frame.from_catalog(
    database="customer_db",
    table_name="customer_tbl",
    sample_query = "select name, location from customer_tbl WHERE id>=1000 AND",

    additional_options = {
                            "hashpartitions": 36 ,
                            "hashfield":"id",
                            "enablePartitioningForSampleQuery":True,
                            "sampleQuery":sample_query
                           }
)
```

Scala

```
val additionalOptions = Map(
        "hashpartitions" -> "36",
        "hashfield" -> "id",
        "enablePartitioningForSampleQuery" -> "true",
        "sampleQuery" -> "select name, location from customer_tbl WHERE id >= 1000
 AND"
        )

    val customer_records = glueContext.getCatalogSource(
        database="customer_db",
        tableName="customer_tbl").getDynamicFrame()
```

> **Note**
>
> customer_tbl이 데이터 카탈로그와 기초 데이터 스토어의 이름이 다른 경우 쿼리가 기초 데이터 스토어로 전달되므로 sample_query에 기초 테이블 이름을 제공해야 합니다.

AWS Glue 데이터 카탈로그와 통합하지 않고도 JDBC 테이블을 쿼리할 수도 있습니다. 사용자 이름과 암호를 메서드의 파라미터로 제공하는 대신 `useConnectionProperties` 및 `connectionName`을 제공하여 기존 연결의 보안 인증을 재사용할 수 있습니다. 이 예제에서는 `my_postgre_connection`이라는 연결에서 보안 인증을 검색합니다.

Python

```
connection_options_dict = {
    "useConnectionProperties": True,
    "connectionName": "my_postgre_connection",
    "dbtable":"customer_tbl",
    "sampleQuery":"select name, location from customer_tbl WHERE id>=1000 AND",
    "enablePartitioningForSampleQuery":True,
    "hashfield":"id",
    "hashpartitions":36
    }

customer_records = glueContext.create_dynamic_frame.from_options(
    connection_type="postgresql",
    connection_options=connection_options_dict
    )
```

Scala

```
val connectionOptionsJson = """
      {
        "useConnectionProperties": true,
        "connectionName": "my_postgre_connection",
        "dbtable": "customer_tbl",
        "sampleQuery": "select name, location from customer_tbl WHERE id>=1000 AND",
        "enablePartitioningForSampleQuery" : true,
        "hashfield" : "id",
        "hashpartitions" : 36
      }
    """

    val connectionOptions = new JsonOptions(connectionOptionsJson)

    val dyf = glueContext.getSource("postgresql",
 connectionOptions).getDynamicFrame()
```

AWS Glue에서 푸시다운에 대한 참고 및 제한 사항

푸시다운은 개념상 비스트리밍 소스(AWS)에서 읽을 때 적용됩니다. Glue는 다양한 소스를 지원하며, 푸시다운 기능은 소스와 커넥터에 따라 다릅니다.

- Snowflake에 연결할 때는 `query` 옵션을 사용할 수 있습니다. AWS Glue 4.0 이상 버전의 Redshift 커넥터에도 비슷한 기능이 있습니다. `query`를 사용하여 Snowflake에서 읽는 방법에 대한 자세한 내용은 the section called “Snowflake에서 읽기” 섹션을 참조하세요.
- DynamoDB ETL 리더는 필터 또는 푸시다운 조건자를 지원하지 않습니다. MongoDB 및 DocumentDB도 이러한 종류의 기능을 지원하지 않습니다.
- Amazon S3에서 오픈 테이블 형식으로 저장된 데이터를 읽을 때는 Amazon S3에 있는 파일을 파티셔닝하는 방법이 더 이상 적합하지 않습니다. 오픈 테이블 형식을 사용하여 파티션에서 읽고 쓰려면 해당 형식에 대한 설명서를 참조하세요.
- DynamicFrame 메서드는 Amazon S3 프로젝션 푸시다운을 수행하지 않습니다. 조건자 필터를 통과한 파일에서 모든 열을 읽습니다.
- AWS Glue에서 `custom.jdbc` 커넥터를 사용할 때 푸시다운 기능은 소스와 커넥터에 따라 달라집니다. 해당 커넥터 설명서를 검토하여 AWS Glue에서 푸시다운을 지원하는지 여부와 지원 방법을 확인하세요.

에 자동 조정 사용 AWS Glue

Auto Scaling은 에서 사용할 수 있습니다.AWS Glue ETL, 대화형 세션 및 스트리밍 작업 AWS Glue 버전 3.0 이상.

Auto Scaling을 사용하면 다음과 같은 이점을 얻을 수 있습니다.

- AWS Glue 는 작업 실행의 각 단계 또는 마이크로배치의 병렬 처리에 따라 클러스터에서 작업자를 자동으로 추가하고 제거합니다.
- 이를 통해 실험하고 에 할당할 작업자 수를 결정할 필요가 줄어듭니다.AWS Glue ETL 작업.
- 주어진 최대 작업자 수를 사용하면 AWS Glue 는 워크로드에 적합한 크기의 리소스를 선택합니다.
- 의 작업 실행 세부 정보 페이지에서 CloudWatch 지표를 확인하여 작업 실행 중에 클러스터 크기가 어떻게 변경되는지 확인할 수 있습니다.AWS Glue 스튜디오.

에 대한 Auto Scaling AWS Glue ETL 및 스트리밍 작업을 통해 의 컴퓨팅 리소스의 온디맨드 스케일 아웃 및 스케일 인 가능 AWS Glue 작업. 온디맨드 확장은 처음에 작업 실행 시작 시 필요한 컴퓨팅 리소스만 할당하고 작업 중 수요에 따라 필요한 리소스를 프로비저닝하는 데 도움이 됩니다.

Auto Scaling은 의 동적 스케일 인도 지원합니다.AWS Glue 작업 과정에서 작업 리소스. 작업 실행 중 Spark 애플리케이션에서 더 많은 실행기를 요청하면 클러스터에 더 많은 작업자가 추가됩니다. 실행기가 활성 계산 태스크 없이 유휴 상태인 경우 실행자와 해당 작업자가 제거됩니다.

오토 스케일링이 Spark 애플리케이션의 비용 및 사용률에 도움이 되는 일반적인 시나리오는 다음과 같습니다.

- Amazon S3에서 많은 수의 파일을 나열하거나 실행기가 비활성 상태일 때 로드를 수행하는 Spark 드라이버
- 과도한 프로비저닝으로 인해 소수의 실행기로만 실행되는 Spark 단계
- Spark 단계에서 데이터 스큐 또는 균일하지 않은 계산 수요

요구 사항

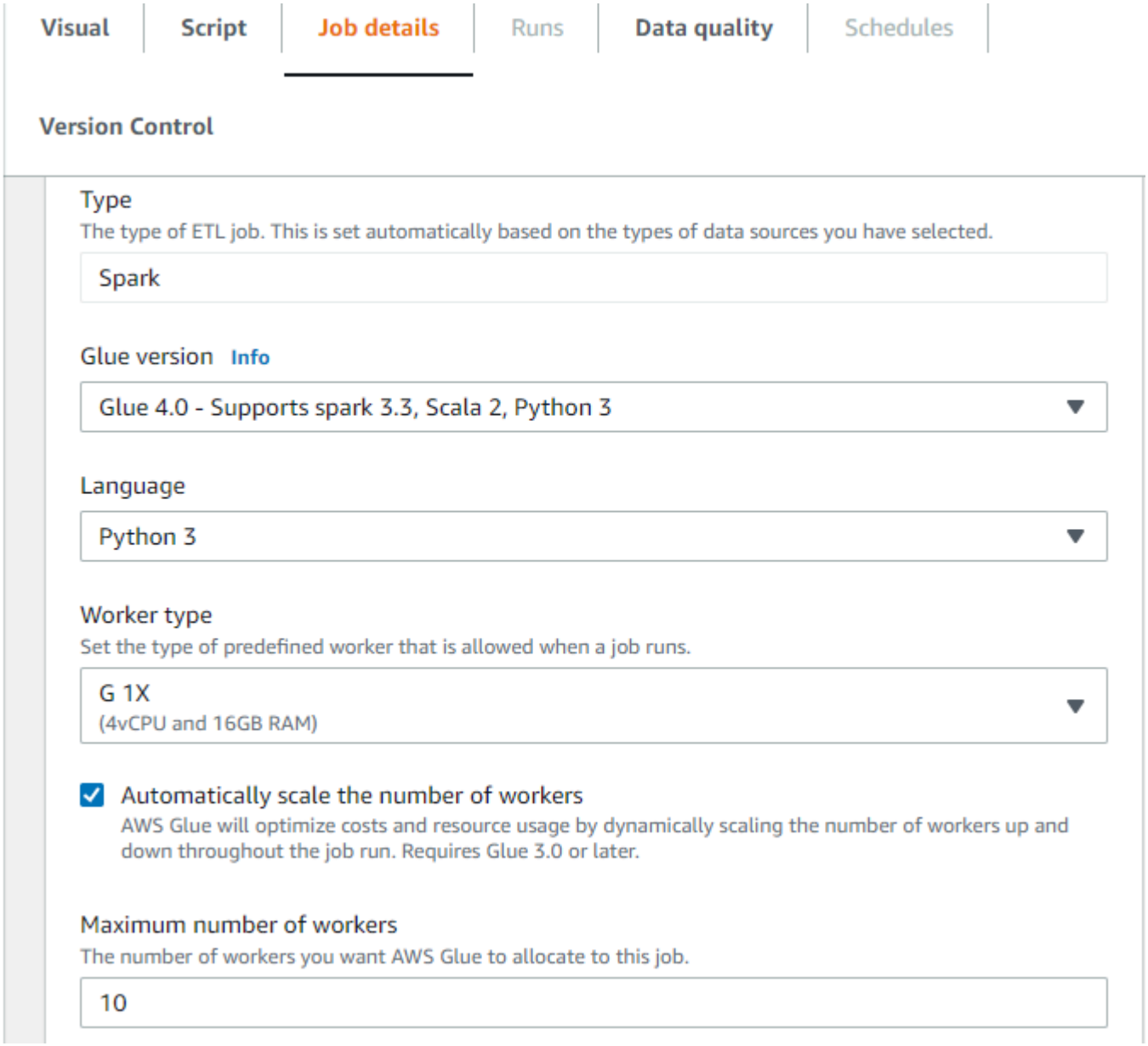
Auto Scaling은 에서만 사용할 수 있습니다.AWS Glue 버전 3.0 이상. Auto Scaling 사용하려면 마이그레이션 가이드에 따라 기존 작업을 로 마이그레이션할 수 있습니다.AWS Glue 버전 3.0 이상 또는 를 사용하여 새 작업 생성 AWS Glue 버전 3.0 이상.

Auto Scaling은 에 사용할 수 있습니다.AWS Glue `G.1X`, `G.2X`, `G.4X``G.8X`, 또는 `G.025X` (스트리밍 작업에만 해당) 작업자 유형이 있는 작업. 표준DPUs은 지원되지 않습니다.

에서 Auto Scaling 활성화 AWS Glue Studio

의 작업 세부 정보 탭에서 AWS Glue Studio에서 유형을 Spark 또는 Spark Streaming 으로 선택하고 Glue 버전 **`Glue 3.0`** 이상을 선택합니다. 그러면 작업자 유형 아래에 확인란이 표시됩니다.

- 작업자 수 자동 크기 조정(Automatically scale the number of workers) 옵션을 선택합니다.
- 최대 작업자 수(Maximum number of workers)를 설정하여 작업 실행에 판매할 수 있는 최대 작업자 수를 정의합니다.



또는 를 AWS CLI 사용하여 Auto Scaling 활성화 SDK

작업 실행을 위해 AWS CLI에서 Auto Scaling을 활성화하려면 다음 구성`start-job-run`으로 를 실행합니다.

```
{
    "JobName": "<your job name>",
    "Arguments": {
        "--enable-auto-scaling": "true"
    },
    "WorkerType": "G.2X", // G.1X and G.2X are allowed for Auto Scaling Jobs
```

```
    "NumberOfWorkers": 20, // represents Maximum number of workers
    ...other job run configurations...
}
```

ETL 작업 실행이 완료되면 를 호출get-job-run하여 DPU-초 이내에 작업 실행의 실제 리소스 사용량을 확인할 수도 있습니다. 참고: 새 필드는 Auto Scaling 으로 활성화된 AWS Glue 4.0 이상의 배치 작업에 DPUSeconds 대해서만 표시됩니다. 이 필드는 스트리밍 작업에는 지원되지 않습니다.

```
$ aws glue get-job-run --job-name your-job-name --run-id jr_xx --endpoint https://
glue.us-east-1.amazonaws.com --region us-east-1
{
    "JobRun": {
        ...
        "GlueVersion": "3.0",
        "DPUSeconds": 386.0
    }
}
```

동일한 구성의 를 사용하여 Auto ScalingAWS Glue SDK으로 작업 실행을 구성할 수도 있습니다.

대화형 세션을 사용하여 Auto Scaling 활성화

대화형 세션으로 AWS Glue 작업을 구축할 때 Auto Scaling을 활성화하려면 대화형 세션 구성을 AWS Glue참조하세요.

팁 및 고려 사항

AWS Glue Auto Scaling 미세 조정을 위한 팁 및 고려 사항:

- 최대 작업자 수의 초기 값에 대한 아이디어가 없는 경우 추정 AWS Glue DPU 에 설명된 대략적인 계산부터 시작할 수 있습니다. 볼륨이 매우 적은 데이터의 경우 최대 작업자 수로 너무 큰 값을 구성해서는 안 됩니다.
- AWS Glue Auto Scaling은 작업에 구성된 최대 수DPU(최대 작업자 수 및 작업자 유형으로 계산됨)를 spark.default.parallelism 기반으로 spark.sql.shuffle.partitions 및 를 구성합니다. 이러한 구성에서 고정 값을 선호하는 경우 다음 작업 파라미터로 이러한 파라미터를 덮어쓸 수 있습니다.
  - 키: --conf
  - 값: spark.sql.shuffle.partitions=200 --conf spark.default.parallelism=200

- 스트리밍 작업의 경우 기본적으로 AWS Glue 는 마이크로 배치 내에서 자동 조정을 수행하지 않으며 자동 조정을 시작하려면 여러 마이크로 배치가 필요합니다. 마이크로 배치 내에서 자동 오토 스케일링을 활성화하려면 `--auto-scale-within-microbatch`를 제공합니다. 자세한 내용은 작업 파라미터 참조를 참조하세요.

Amazon CloudWatch 지표를 사용한 Auto Scaling 모니터링

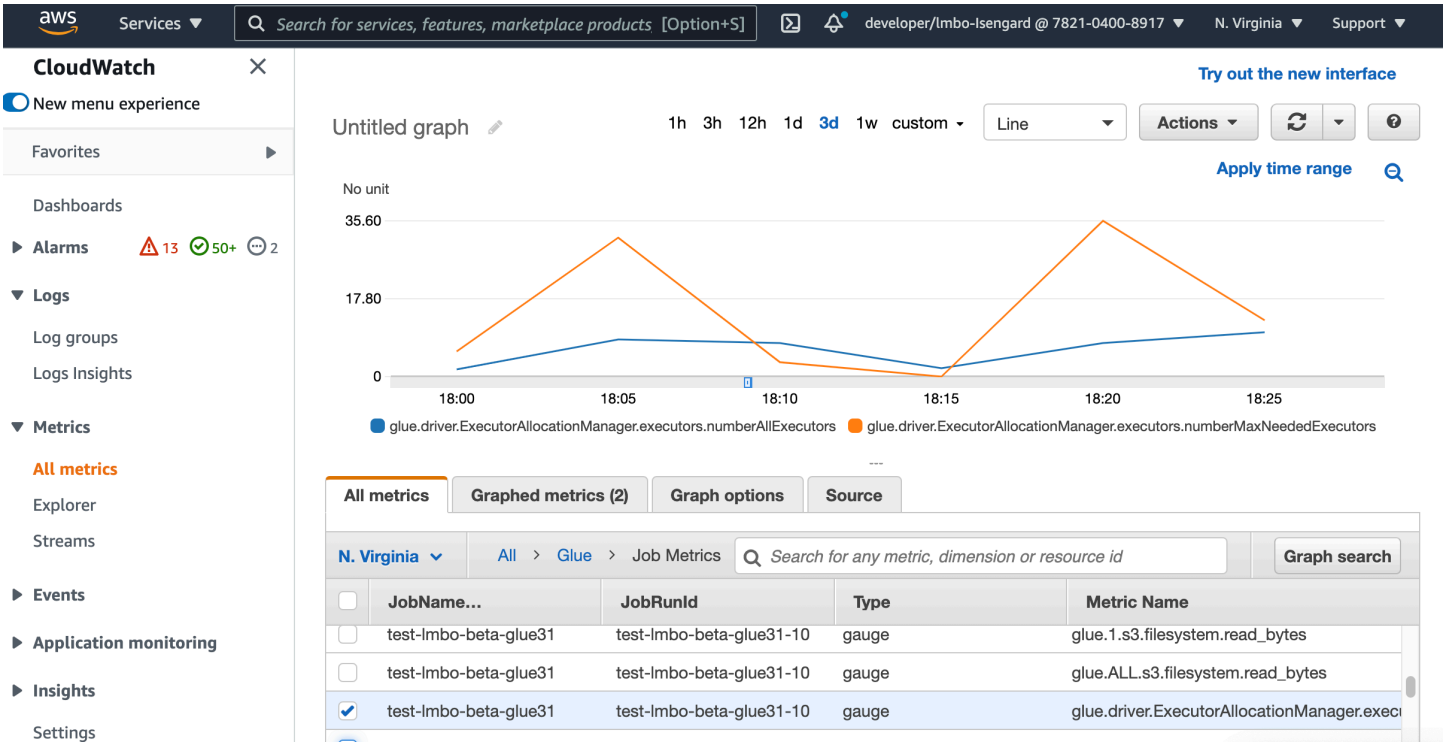
CloudWatch 실행기 지표는 AWS Glue Auto Scaling을 활성화한 경우 3.0 이상 작업. 지표를 사용하여 Auto Scaling으로 사용 가능한 Spark 애플리케이션의 실행기 수요와 최적 사용량을 모니터링할 수 있습니다. 자세한 내용은 Amazon CloudWatch 지표를 사용하여 AWS Glue 모니터링 단원을 참조하십시오.

AWS Glue 또한 관찰 가능성 지표를 활용하여 리소스 사용률에 대한 인사이트를 얻을 수 있습니다. 예를 들어 `glue.driver.workerUtilization`을 모니터링하여 오토 스케일링을 사용하거나 사용하지 않고 실제로 사용된 리소스의 양을 모니터링할 수 있습니다. 또 다른 예로 `glue.driver.skewness.job` 및 `glue.driver.skewness.stage`를 모니터링하여 데이터 스큐 상태를 확인할 수 있습니다. 이러한 인사이트는 오토 스케일링을 활성화하고 구성을 미세 조정하는 데 도움이 됩니다. 자세한 내용은 AWS Glue 관찰성 메트릭을 사용한 모니터링를 사용하여 모니터링을 참조하세요.

- glue.driver.ExecutorAllocationManager.executors.numberAllExecutors
- glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors

이러한 지표에 대한 자세한 내용은 DPU 용량 계획 모니터링 섹션을 참조하세요.

> **Note**
>
> CloudWatch 대화형 세션에는 실행기 지표를 사용할 수 없습니다.

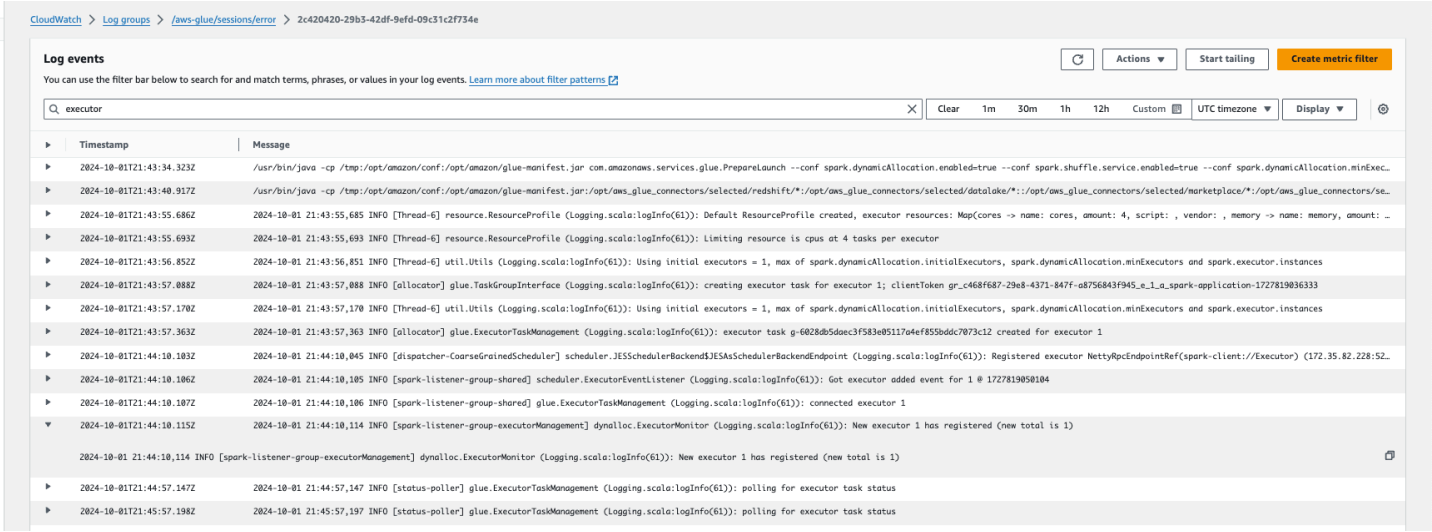


Amazon CloudWatch Logs를 사용한 Auto Scaling 모니터링

대화형 세션을 사용하는 경우 연속 Amazon CloudWatch Logs를 활성화하고 로그에서 “실행기”를 검색하거나 Spark UI를 사용하여 실행기 수를 모니터링할 수 있습니다. 이렇게 하려면 `%%configure` 매직을 사용하여 `enable auto scaling`을 통해 연속 로깅을 활성화합니다.

```
%%configure{
    "--enable-continuous-cloudwatch-log": "true",
    "--enable-auto-scaling": "true"
}
```

Amazon CloudWatch Logsevents에서 로그에서 'executor'를 검색합니다.

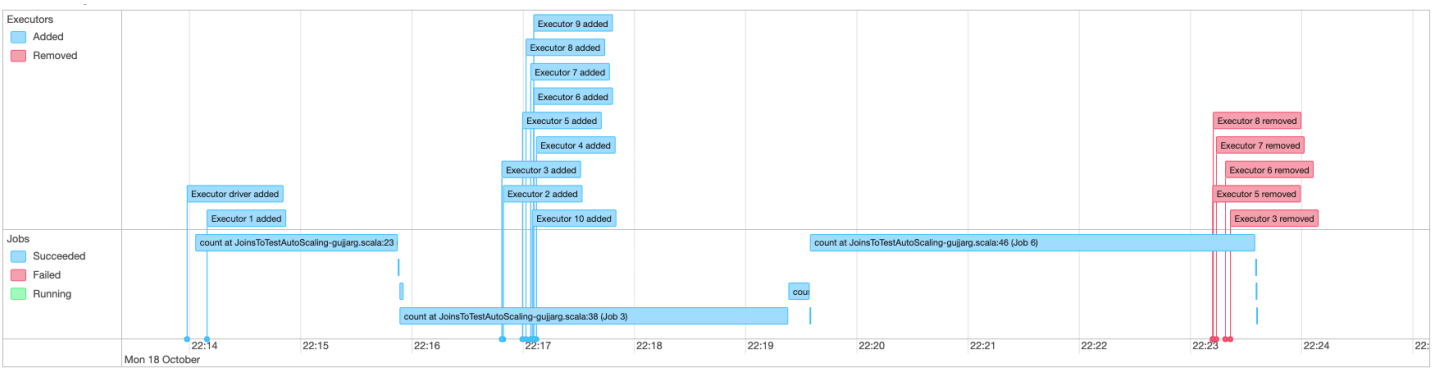


Spark UI로 Auto Scaling 모니터링

Auto Scaling을 활성화하면 의 수요에 따라 동적 스케일 업 및 스케일 다운을 사용하여 추가 및 제거되는 실행기를 모니터링할 수도 있습니다.AWS Glue Glue Spark UI를 사용하는 작업. 자세한 내용은 AWS Glue 작업을 위한 Apache Spark 웹 UI 사용 설정 단원을 참조하십시오.

Jupyter Notebook의 대화형 세션을 사용하는 경우 다음 매직을 실행하여 Spark UI와 함께 오토 스케일링을 활성화할 수 있습니다.

```
%%configure{
    "--enable-auto-scaling": "true",
    "--enable-continuous-cloudwatch-log": "true"
}
```



Auto Scaling 작업 실행 DPU 사용량 모니터링

AWS Glue Studio 작업 실행 보기를 사용하여 Auto Scaling 작업의 DPU 사용량을 확인할 수 있습니다.

1. AWS Glue Studio 탐색 창에서 모니터링을 선택합니다. 모니터링 페이지가 표시됩니다.
2. 아래로 스크롤하여 작업 실행 차트로 이동합니다.
3. 관심 있는 작업 실행으로 이동하여 DPU 시간 열로 스크롤하여 특정 작업 실행의 사용량을 확인합니다.

제한 사항

AWS Glue 스트리밍 Auto Scaling은 현재 외부에서 DataFrame 생성된 정적이 있는 스트리밍 DataFrame 조인을 지원하지 않습니다`ForEachBatch`. 내부에 DataFrame 생성된 정적`ForEachBatch`은 예상대로 작동합니다.

제한된 실행을 통한 워크로드 분할

Spark 애플리케이션의 오류는 일반적으로 비효율적인 Spark 스크립트, 대규모 트랜스포메이션의 분산된 인메모리 실행 및 데이터 집합 이상으로 인해 발생합니다. 드라이버 또는 실행기의 메모리 부족 문제를 일으킬 수 있는 많은 이유가 있습니다(예: 데이터 왜곡, 너무 많은 객체 나열 또는 큰 데이터 셔플). 이러한 문제는 Spark로 엄청난 양의 백로그 데이터를 처리할 때 자주 발생합니다.

AWS Glue를 사용하면 OOM 문제를 해결하고 워크로드 분할을 통해 ETL 처리를 더 쉽게 할 수 있습니다. 워크로드 분할이 활성화되면 각 ETL 작업 실행은 처리되지 않은 데이터만 선택하며, 이 작업 실행으로 처리할 파일 수 또는 데이터 집합 크기에 대한 상한이 있습니다. 향후 작업 실행은 나머지 데이터를 처리합니다. 예를 들어 1,000개의 파일을 처리해야 하는 경우 파일 수를 500으로 설정하고 2개의 작업 실행으로 분리할 수 있습니다.

워크로드 분할은 Amazon S3 데이터 원본에 대해서만 지원됩니다.

워크로드 분할 사용

스크립트에서 수동으로 옵션을 설정하거나 카탈로그 테이블 속성을 추가하여 제한된 실행을 사용할 수 있습니다.

스크립트에서 제한된 실행으로 워크로드 분할을 사용하려면

1. 데이터를 다시 처리하지 않으려면 새 작업 또는 기존 작업에서 작업 북마크를 사용합니다. 자세한 내용은 작업 북마크를 사용하여 처리된 데이터 추적을 참조하세요.
2. 스크립트를 수정하고 AWS Glue `getSource` API의 추가 옵션에서 경계 제한을 설정합니다. `state` 요소를 저장하기 위해 작업 북마크에 대한 트랜스포메이션 컨텍스트도 설정해야 합니다. 예:

Python

```
glueContext.create_dynamic_frame.from_catalog(
    database = "database",
    table_name = "table_name",
    redshift_tmp_dir = "",
    transformation_ctx = "datasource0",
    additional_options = {
        "boundedFiles" : "500", # need to be string
      # "boundedSize" : "1000000000" unit is byte
    }
)
```

Scala

```
val datasource0 = glueContext.getCatalogSource(
    database = "database", tableName = "table_name", redshiftTmpDir = "",
    transformationContext = "datasource0",
    additionalOptions = JsonOptions(
        Map("boundedFiles" -> "500") // need to be string
      //"boundedSize" -> "1000000000" unit is byte
    )
).getDynamicFrame()
```

```
val connectionOptions = JsonOptions(
    Map("paths" -> List(baseLocation), "boundedFiles" -> "30")
)
val source = glueContext.getSource("s3", connectionOptions, "datasource0", "")
```

Data Catalog 테이블에서 제한된 실행으로 워크로드 분할을 사용하려면

1. Data Catalog에서 테이블 구조의 `parameters` 필드에 키-값 페어를 설정합니다. 자세한 내용은 테이블 세부 정보 보기 및 편집을 참조하세요.
2. 데이터 집합 크기 또는 처리되는 파일 수의 상한을 설정합니다.

   - `boundedSize`를 데이터 집합의 대상 크기(바이트)로 설정합니다. 테이블에서 대상 크기에 도달하면 작업 실행이 중지됩니다.

- boundedFiles를 대상 파일 수로 설정합니다. 대상 파일 수를 처리한 후 작업 실행이 중지됩니다.

> **Note**
>
> 하나의 경계만 지원되므로 boundedSize 또는 boundedFiles 중 하나만 설정해야 합니다.

작업을 자동으로 실행하도록 AWS Glue 트리거 설정

제한된 실행을 사용하면 작업을 자동으로 실행하고 순차적 실행에서 데이터를 증분적으로 로드하도록 AWS Glue 트리거를 설정할 수 있습니다. AWS Glue 콘솔로 이동하여 트리거를 생성하고 일정 시간을 설정하고 작업에 연결합니다. 그런 다음 자동으로 다음 작업 실행을 트리거하고 새 데이터 배치를 처리합니다.

또한 AWS Glue 워크플로로 여러 작업을 오케스트레이션하여 서로 다른 파티션의 데이터를 병렬로 처리할 수 있습니다. 자세한 내용은 AWS Glue 트리거 및 AWS Glue 워크플로를 참조하세요.

사용 사례 및 옵션에 대한 자세한 내용은 AWS Glue에서 워크로드 분할로 Spark 애플리케이션 최적화를 참조하세요.

AWS Glue의 알려진 문제

다음은 AWS Glue에서 알려진 문제입니다.

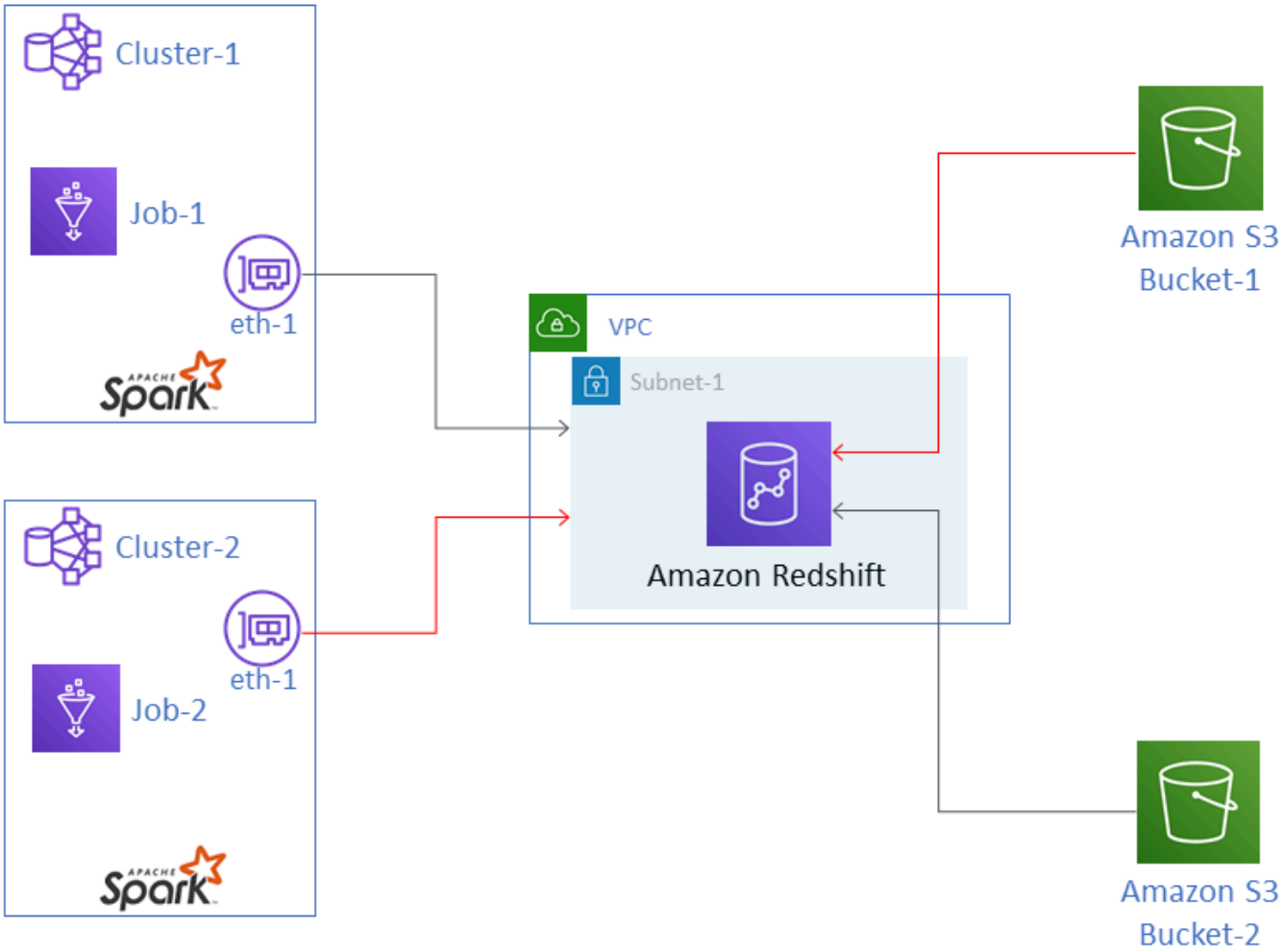
주제

- 작업 간 데이터 액세스 방지

작업 간 데이터 액세스 방지

각각 별도의 AWS Glue Spark 클러스터에서 실행되는 AWS Glue Spark 작업 2개가 단일 AWS 계정에 있다고 가정하겠습니다. 두 작업은 AWS Glue 연결을 사용해 동일한 가상 프라이빗 클라우드(VPC)의 리소스에 액세스합니다. 이때 한쪽 클러스터에서 실행되는 작업은 다른 클러스터에서 실행되는 작업의 데이터에 액세스할 수 있어야 합니다.

다음 다이어그램은 이러한 상황을 예로 든 그림입니다.



위 다이어그램을 보면 AWS Glue Job-1은 Cluster-1에서, 그리고 Job-2는 Cluster-2에서 실행 중입니다. 두 작업 모두 VPC의 Subnet-1에 속하면서 동일한 Amazon Redshift 인스턴스와 통신하고 있습니다. 이때 Subnet-1은 퍼블릭 서브넷 또는 프라이빗 서브넷이 될 수 있습니다.

Job-1은 Amazon Simple Storage Service(Amazon S3) Bucket-1의 데이터를 변환하여 Amazon Redshift에 작성하고 있습니다. Job-2는 Bucket-2의 데이터를 사용해 동일하게 실행하고 있습니다. Job-1은 AWS Identity and Access Management(IAM) 역할인 Role-1(그림에 보이지 않음)을 사용해 Bucket-1에 대한 액세스 권한을 얻습니다. Job-2는 Role-2(그림에 보이지 않음)를 사용해 Bucket-2에 대한 액세스 권한을 얻습니다.

두 작업은 서로 클러스터와 통신할 수 있는 네트워크 경로가 있기 때문에 각 데이터에 액세스할 수 있습니다. 예를 들어 Job-2는 Bucket-1의 데이터에 액세스할 수 있습니다. 위 다이어그램에서는 빨간색으로 표시된 선이 네트워크 경로입니다.

이러한 상황을 방지하려면 `Job-1`과 `Job-2`에 서로 다른 보안 구성을 연결하는 것이 좋습니다. 보안 구성을 연결하면 AWS Glue에서 생성되는 인증서를 통해 작업 간 데이터 액세스가 차단됩니다. 보안 구성은 더미 구성이 될 수 있습니다. 이 말은 Amazon S3 데이터, Amazon CloudWatch 데이터 또는 작업 북마크를 암호화하지 않고 보안 구성을 생성할 수 있다는 것을 의미합니다. 세 가지 암호화 옵션 모두 비활성화가 가능합니다.

보안 구성에 대한 자세한 내용은 the section called “AWS Glue에서 작성한 데이터 암호화” 단원을 참조하십시오.

보안 구성을 작업에 연결하려면

1. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.
2. 해당 작업의 Configure the job properties(작업 속성 구성) 페이지에서 Security configuration, script libraries, and job parameters(보안 구성, 스크립트 라이브러리 및 작업 파라미터) 섹션을 확장합니다.
3. 목록에서 보안 구성을 선택합니다.

AWS Glue에 대한 문서 기록

변경 사항	설명	날짜
AWS Glue에 대한 새 SaaS 기본 커넥터 14개 추가 지원	AWS Glue에 14개의 SaaS 기본 커넥터가 추가되었습니다. 자세한 내용은 AWS Glue 연결 추가를 참조하세요.	2025년 1월 30일
AWS Glue에 대한 새 SaaS 기본 커넥터 16개 추가 지원	AWS Glue에 16개의 SaaS 기본 커넥터가 더 추가되었습니다. 자세한 내용은 AWS Glue 연결 추가를 참조하세요.	2024년 12월 17일
자동 열 통계 생성	이제 AWS Glue Data Quality는 Data Catalog 및 ETL에서 Amazon SageMaker AI LakeHouse 테이블과 AWS Lake Formation 관리형 Iceberg, Delta 및 HUDI 테이블을 지원합니다. 자세한 내용은 AWS Glue Data Quality를 참조하세요.	2024년 12월 6일
제로 ETL 통합 지원	제로 ETL은 ETL 데이터 파이프라인을 구축할 필요성을 최소화하는 AWS의 완전 관리형 통합 세트입니다. 자세한 내용은 제로 ETL 통합을 참조하세요.	2024년 12월 3일
재사용 가능한 연결 지원	새로운 AWS Glue 연결 스키마는 AWS Glue, Amazon Athena, Amazon SageMaker Unified Studio 등과 같은 AWS 서비스 및 애플리케이션 전반에서 데이터 연결을 관리하는	2024년 12월 3일

	통일된 방법을 제공합니다. 자세한 내용은 데이터에 연결을 참조하세요.	
AWS Glue 버전 5.0을 지원합니다.	AWS Glue 버전 5.0에 대한 지원 정보가 추가되었습니다. 기능에는 Apache Spark 3.52 업데이트, Java 17 업데이트, 오픈 테이블 형식 업데이트, Spark 네이티브 세분화된 액세스 제어, Sagemaker Lakehouse 및 데이터 웨어하우스 추상화 통합, Sagemaker Unified Studio 지원 등이 포함됩니다. 자세한 내용은 AWS Glue 릴리스 정보와 AWS Glue 버전 5.0으로 AWS Glue 작업 마이그레이션을 참조하세요.	2024년 12월 3일
AWS Glue Iceberg REST 엔드포인트를 사용하여 AWS Glue Data Catalog에 연결	AWS Glue의 Iceberg REST 엔드포인트는 Apache Iceberg REST 사양에 명시된 API 작업을 지원합니다. Iceberg REST 클라이언트를 사용하여 분석 엔진에서 실행되는 애플리케이션을 Data Catalog에 호스팅되는 REST 카탈로그에 연결할 수 있습니다. 자세한 내용은 Data Catalog 액세스를 참조하세요.	2024년 12월 3일
자동 열 통계 생성	AWS Glue Data Catalog에서 새 테이블에 대한 열 통계를 자동으로 생성합니다. 자세한 내용은 자동 열 통계 생성을 참조하세요.	2024년 12월 3일

AWS Glue에서 Apache Spark에 대한 생성형 AI 업그레이드 지원	AWS Glue에서 Spark 업그레이드를 통해 데이터 엔지니어와 개발자는 생성형 AI를 사용하여 기존 AWS Glue Spark 작업을 최신 Spark 릴리스로 업그레이드 및 마이그레이션할 수 있습니다. 자세한 내용은 AI를 사용하여 분석 업그레이드를 참조하세요.	2024년 11월 22일
AWS Glue에서 Apache Spark에 대한 생성형 AI 문제 해결 지원	AWS Glue에서 Apache Spark 작업에 대한 생성형 AI 문제 해결을 통해 데이터 엔지니어와 과학자는 Spark 애플리케이션에서 문제를 쉽게 진단하고 수정할 수 있습니다. 자세한 내용은 AI를 사용하여 Spark 작업 문제 해결을 참조하세요.	2024년 11월 22일
Iceberg 옵티마이저가 VPC에서 Amazon S3 버킷에 액세스하도록 지원	AWS Glue Data Catalog에서는 Iceberg 테이블 옵티마이저를 통해 AWS Glue 네트워크 연결을 사용하여 특정 가상 프라이빗 클라우드(VPC)에서 Amazon S3 버킷에 액세스할 수 있도록 지원합니다. 자세한 내용은 Iceberg 테이블 최적화를 참조하세요.	2024년 11월 20일
AWS Glue에 대한 9개의 새로운 SaaS 기본 커넥터 지원	AWS Glue에 9개의 SaaS 기본 커넥터가 더 추가되었습니다. 자세한 내용은 AWS Glue 연결 추가를 참조하세요.	2024년 11월 19일

AWS Glue에 대한 10개의 새 SaaS 기본 커넥터 지원	AWS Glue에 10개의 SaaS 기본 커넥터가 추가되었습니다. 자세한 내용은 AWS Glue 연결 추가를 참조하세요.	2024년 11월 15일
AWS Glue ETL 작업에 대한 작업 실행 대기열 지원	나중에 서비스 할당량 때문에 작업을 즉시 실행할 수 없는 경우 작업 실행 대기열을 켜서 작업을 실행할 수 있습니다. 자세한 내용은 AWS Glue에서 Spark 작업에 대한 작업 속성 구성을 참조하세요.	2024년 9월 3일
업데이트된 정책 변경 사항	소유자 태그 키가 있는 세션의 태그 온 생성을 지원하는 데 필요한 AWS Glue 세션 사용자 제한 노트북 정책 및 AWS Glue 세션 사용자 제한 노트북 서비스 역할 정책의 문서화된 변경 사항. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2024년 8월 30일
이제 이상 탐지 기능과 동적 규칙을 정식 버전으로 사용할 수 있습니다.	AWS Glue Data Quality는 기계 학습 알고리즘을 활용하여 과거 동향을 학습한 다음 미래 값을 예측하여 이상을 탐지합니다. 동적 규칙을 사용하면 동적 임계값을 제공할 수 있습니다. 자세한 내용은 Iceberg 테이블을 위한 쿼리 성능 최적화를 참조하세요.	2024년 8월 7일

업데이트된 정책 변경 사항	소유자 태그 키가 있는 세션의 태그 온 생성을 지원하는 데 필요한 AWS Glue 세션 사용자 제한 정책 및 AWS Glue 세션 사용자 제한 서비스 역할 정책의 문서화된 변경 사항. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2024년 8월 5일
Iceberg 테이블의 열 통계 생성은 지금 일반적으로 이용 가능합니다	AWS Glue에서는 Iceberg 테이블의 각 열에 대한 고유 값의 수(NDV) 계산 및 업데이트를 지원합니다. 자세한 내용은 AWS Glue Data Quality의 이상 탐지와 동적 규칙을 참조하세요.	2024년 7월 9일
AWS Glue 사용 프로필 지원	관리자는 개발자, 테스터, 제품 팀 등 계정 내의 다양한 사용자 계층에 대한 AWS Glue 사용 프로필을 생성할 수 있습니다. 이러한 유연성 덕분에 관리자는 각 사용자 클래스마다 서로 다른 사용량 및 비용 관리 방식을 적용할 수 있습니다. 자세한 내용은 AWS Glue 사용 프로필 설정을 참조하세요.	2024년 6월 18일
AWS Glue for Spark용 Salesforce 커넥터 지원	Salesforce의 새 AWS Glue 커넥터에 대한 정보가 추가되었습니다. 이 기능은 AWS Glue 4.0 이상 버전에서 AWS Glue for Spark를 사용하여 읽고 쓸 수 있도록 해줍니다. 자세한 내용은 Salesforce에 연결을 참조하세요.	2024년 5월 22일

AWS Glue의 Amazon Q 데이터 통합(GA)	AWS Glue의 Amazon Q 데이터 통합은 AWS Glue의 새로운 생성형 AI 기능으로, 데이터 엔지니어와 ETL 개발자가 자연어를 사용하여 데이터 통합 작업을 구축할 수 있도록 지원합니다. 엔지니어와 개발자는 Q에 작업 작성과 문제 해결을 요청하고 AWS Glue 및 데이터 통합에 관해 질문할 수 있습니다. 자세한 내용은 AWS Glue의 Amazon Q 데이터 통합을 참조하세요. 이 기능에는 `AwsGlueSessionUserRestrictedPolicy`, `AwsGlueSessionUserRestrictedNotebookServiceRole` 및 `AwsGlueSessionUserRestrictedServiceRole` AWS 관리형 정책에 대한 업데이트가 포함되어 있습니다. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2024년 4월 30일

AWS Glue의 Amazon Q 데이터 통합(미리 보기)	AWS Glue의 Amazon Q 데이터 통합은 AWS Glue의 새로운 생성형 AI 기능으로, 데이터 엔지니어와 ETL 개발자가 자연어를 사용하여 데이터 통합 작업을 구축할 수 있도록 지원합니다. 엔지니어와 개발자는 Q에 작업 작성과 문제 해결을 요청하고 AWS Glue 및 데이터 통합에 관해 질문할 수 있습니다. 자세한 내용은 AWS Glue의 Amazon Q 데이터 통합을 참조하세요. 이 기능에는 `AwsGlueSessionUser RestrictedNotebook Policy` AWS 관리형 정책에 대한 업데이트가 포함되어 있습니다. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2024년 1월 30일
AWS Glue 스트리밍 설명서 업데이트	AWS Glue 스트리밍을 위한 새롭고 재구성된 콘텐츠로 새로운 단원을 추가했습니다. 이 콘텐츠는 AWS Glue에서 스트리밍이 작동하는 방식, 실시간 데이터 처리의 특성, 스트리밍 작업을 모니터링하는 방법을 설명합니다. 자세한 내용은 AWS Glue 스트리밍을 참조하세요.	2023년 12월 27일

세분화된 민감한 데이터 탐지 사용 지원	민감한 데이터 탐지 변환은 사용자가 정의하거나 AWS Glue에서 사전 정의한 엔터티를 탐지, 마스크 또는 제거하는 기능을 제공합니다. 또한 세분화된 액션을 통해 엔터티별로 특정 액션을 적용할 수 있습니다. 자세한 내용은 세분화된 민감한 데이터 탐지 사용을 참조하세요.	2023년 11월 26일
AWS Glue 관찰성 지표를 사용한 작업 모니터링 지원	AWS Glue 관찰성 메트릭을 사용하면 Apache Spark의 AWS Glue 내부에서 일어나는 일에 대한 통찰력을 얻어 문제의 분류 및 분석을 개선할 수 있습니다. 자세한 내용은 AWS Glue 관찰성 지표를 사용한 모니터링을 참조하세요.	2023년 11월 26일
AWS Glue Data Quality의 이상 탐지 지원	AWS Glue Data Quality 이상 탐지는 시간 경과에 따른 데이터 통계에 기계 학습(ML) 알고리즘을 적용하여 규칙으로 탐지하기 어려운 비정상적인 패턴과 숨겨진 데이터 품질 문제를 탐지합니다. 자세한 내용은 AWS Glue Data Quality의 이상 탐지를 참조하세요.	2023년 11월 26일

기본 Spark UI 로깅 동작으로 업데이트	Spark UI 로그를 생성하는 Spark 작업은 이제 AWS Glue 콘솔에서 Spark UI를 지원하기 위해 다른 파일 이름 패턴으로 작성됩니다. 이렇게 해도 CloudWatch 로그 동작은 변경되지 않습니다. 작업 구성을 업데이트하여 기존 동작으로 되돌릴 수 있습니다. 자세한 내용은 Apache Spark 웹 UI를 사용하여 작업 모니터링을 참조하세요.	2023년 11월 17일
AWS Glue for Spark의 새 데이터 소스 지원	이제 Amazon OpenSearch Service, Azure SQL, Azure Cosmos for NoSQL, SAP HANA Teradata Vantage, Vertica에 대한 연결이 AWS Glue 내에서 기본적으로 지원됩니다. 또한 이러한 데이터 소스에 대한 연결은 이제 AWS Glue 스튜디오 시각적 편집기에서 MongoDB와 함께 사용할 수 있습니다. Spark 지원을 위한 AWS Glue에 대한 자세한 내용은 AWS Glue for Spark에서 ETL 연결에 대한 연결 유형 및 옵션을 참조하고, AWS Glue Studio 시각적 편집기 사용에 대한 자세한 내용은 AWS Glue 연결 추가를 참조하세요.	2023년 11월 17일

열 통계 생성 지원	추가 데이터 파이프라인을 설정하지 않고도 Parquet, ORC, JSON, ION, CSV 및 XML과 같은 데이터 형식의 AWS Glue Data Catalog 테이블에 대한 열 수준 통계를 계산할 수 있습니다. 자세한 내용은 열 통계 작업을 참조하세요.	2023년 11월 16일
Iceberg 테이블의 데이터 압축 지원	AWS 분석 서비스(예: Amazon Athena 및 Amazon EMR)와 AWS Glue ETL 작업에서 읽기 성능을 향상시키기 위해 데이터 카탈로그는 데이터 카탈로그의 Iceberg 테이블에 대해 관리형 압축(작은 Amazon S3 객체를 큰 객체로 압축하는 프로세스)을 제공합니다. 자세한 내용은 Iceberg 테이블 최적화를 참조하세요.	2023년 11월 13일
작업 실행 대기 동작 업데이트	표준 Spark 및 Python 쉘 작업 실행이 이제 특정 상황에서 `WAITING`으로 즉시 전환되지 않고 `FAILED`로 전환됩니다. 자세한 내용은 AWS Glue 작업 실행 상태를 참조하세요.	2023년 11월 8일
AWS Glue Studio 사용자 설명서가 AWS Glue 개발자 설명서로 통합됨	AWS Glue Studio 사용 설명서가 개발자 안내서로 이동하여 AWS Glue 콘솔 및 AWS Glue Studio 프로그래밍 액세스에 대한 AWS Glue Studio의 단일 통합 사용 설명서가 생성되었습니다.	2023년 10월 25일

AWSGlueServiceNotebookRole AWS 관리형 정책으로 업데이트	AWSGlueServiceNotebookRole AWS 관리형 정책의 마이너 업데이트에 대한 정보가 추가되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2023년 10월 9일
AWS Glue Studio에서 다섯 가지 새로운 기본 제공 변환 지원	AWS Glue Studio에서는 5가지 새로운 기본 제공 변환(레코드 일치, null 행 제거, JSON 열 구문 분석, JSON 경로 추출, 정규식 추출기)을 지원합니다. 자세한 내용은 AWS Glue 관리형 데이터 변환 노드 편집을 참조하세요.	2023년 8월 11일
AWSGlueServiceRole AWS 관리형 정책으로 업데이트	AWSGlueServiceRole AWS 관리형 정책의 마이너 업데이트에 대한 정보가 추가되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2023년 8월 4일
Apache Hudi 테이블 크롤링에 대한 지원	AWS Glue를 사용하여 Amazon S3 버킷에서 Hudi 테이블을 크롤링하고 Hudi 테이블을 AWS Glue Data Catalog에 등록하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 크롤링할 수 있는 데이터 스토어는 무엇인가요? 및 Crawler properties를 참조하세요.	2023년 7월 21일

AWSGlueConsoleFullAccess AWS 관리형 정책 업데이트	AWSGlueConsoleFullAccess AWS 관리형 정책의 마이너 업데이트에 대한 정보가 추가되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2023년 7월 14일
Apache Iceberg 테이블 크롤링에 대한 지원	AWS Glue를 사용하여 Amazon S3 버킷에서 Iceberg 테이블을 크롤링하고 Iceberg 테이블을 AWS Glue Data Catalog에 등록하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 크롤링할 수 있는 데이터 스토어는 무엇인가요? 및 Crawler properties를 참조하세요.	2023년 7월 7일
Ray를 사용하는 AWS Glue에 대한 지원	AWS Glue 작업을 지원할 수 있는 새로운 엔진, Ray를 포함하는 AWS Glue에 대한 정보가 추가되었습니다. Spark 콘텐츠와 함께 기존 AWS Glue를 재구성하여 모호함을 없앴습니다.	2023년 5월 30일

AWS Glue Data Quality에 대한 지원(정식 출시)	AWS Glue Data Quality가 정식 출시되었습니다. AWS Glue Data Quality는 데이터 품질을 평가하고 모니터링하는 데 도움이 됩니다. 데이터 카탈로그에서 AWS Glue Data Quality를 사용하는 방법에 대한 자세한 내용은 AWS Glue Data Quality를 참조하세요. AWS Glue Studio의 AWS Glue Data Quality에 대해 알아보려면 AWS Glue Studio에서 데이터 품질 평가를 참조하세요.	2023년 5월 24일
Apache Spark 작업에 대한 대규모 작업자 유형 지원	이제 Apache Spark 작업에 대한 G.4X 및 G.8X 작업자 유형을 사용할 수 있습니다. 이러한 작업자 유형은 워크로드에 가장 까다로운 변환, 집계, 조인 및 쿼리가 포함된 작업에 적합합니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2023년 5월 8일
테이블 크롤링 시 파티션 인덱스 생성에 대한 지원	크롤러가 감지하는 테이블의 파티션 인덱스 생성을 크롤러에서 지원하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 파티션 인덱스 크롤러 구성 옵션 설정을 참조하세요.	2023년 4월 24일
리소스 사용량 지표에 대한 지원	Amazon CloudWatch에서 서비스의 리소스 사용량을 확인하고 경보를 구성하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 리소스 모니터링을 참조하세요.	2023년 4월 7일

AWSGlueConsoleFullAccess AWS 관리형 정책 업데이트	AWSGlueConsoleFullAccess AWS 관리형 정책의 마이너 업데이트에 대한 정보가 추가되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2023년 3월 28일
AWS Glue를 AWS SDK와 함께 사용하기 위한 지침(예제 포함) 추가	AWS Glue 개발자 안내서에는 AWS Glue와 AWS SDK를 함께 사용하는 데 도움이 되는 정보를 제공하는 두 개의 새 섹션이 있습니다. 자세한 내용은 AWS Glue와 AWS SDK를 함께 사용 및 AWS SDK를 사용한 AWS Glue에 대한 코드 예제를 참조하세요.	2023년 2월 23일
AWS Glue를 사용하는 IAM 설명서 업데이트	AWS Glue를 사용하는 IAM 사용에 대한 정보를 재구성하고 추가했습니다. 자세한 내용은 AWS Glue의 자격 증명 및 액세스 관리를 참조하세요.	2023년 2월 15일
AWS Glue 버전 4.0에서 ETL 작업 스트리밍 실행 지원	Glue 버전 4.0에서 ETL 작업 스트리밍을 실행하는 데 대한 지원과 Kafka 클러스터 또는 Apache Kafka 클러스터용 Amazon 관리형 스트리밍 및 Amazon Kinesis Data Streams에 연결하기 위한 새로운 옵션에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue에서 ETL 작업 스트리밍 추가 및 AWS Glue의 ETL에 대한 연결 유형 및 옵션을 참조하세요.	2023년 2월 8일

MongoDB Atlas 데이터 소스 크롤링 지원	AWS Glue를 사용하여 MongoDB Atlas 데이터 소스를 크롤링하는 데 사용하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 어떤 데이터 스토어를 크롤링할 수 있나요?, MongoDB 및 MongoDB Atlas 연결 속성, MongoDB 또는 MongoDB Atlas 연결 사용을 참조하세요.	2023년 2월 6일
기본 Delta Lake 커넥터를 사용하여 Delta Lake 테이블 크롤링 지원	AWS Glue를 사용하여 기본 Delta Lake 커넥터를 통해 Delta Lake 테이블을 크롤링하는 작업에 대한 정보를 추가했습니다. 이 기능을 사용하면 AWS 쿼리 엔진을 통해 델타 트랜잭션 로그를 직접 쿼리하고 시간 여행 및 ACID 보장과 같은 기능을 사용할 수 있으며, Amazon S3 트랜잭션 파일의 Delta Lake 메타데이터를 데이터 카탈로그에 동기화하여 Lake Formation의 쿼리에 대한 열 권한을 활성화할 수 있습니다. 자세한 내용은 Delta Lake 데이터 스토어에 대한 구성 옵션을 지정하는 방법. Delta Lake 테이블 쿼리하기를 참조하세요.	2022년 12월 15일

AWS Glue 데이터 품질 지원(미리 보기)	이제 AWS Glue 데이터 품질(미리 보기)을 지원할 수 있습니다. AWS Glue 데이터 품질은 AWS Glue 3.0을 사용할 때 데이터 품질을 평가하고 모니터링하는 데 도움이 됩니다. 데이터 카탈로그에서 AWS Glue 데이터 품질을 사용하는 방법에 대한 자세한 내용은 AWS Glue 데이터 품질(미리 보기)을 참조하세요. AWS Glue Studio의 AWS Glue Data Quality에 대해 알아보려면 AWS Glue Studio에서 데이터 품질 평가를 참조하세요.	2022년 11월 30일
새로운 기능과 향상된 성능을 갖춘 새로운 Amazon Redshift Spark 커넥터 지원	이제 데이터 수집 및 변환 파이프라인의 일부로 Amazon Redshift에서 데이터를 읽고 쓰는 Apache Spark 애플리케이션을 빌드하기 위해 AWS Glue ETL 작업에 사용할 수 있는 새로운 JDBC 드라이버가 포함된 Amazon Redshift Spark 커넥터를 지원할 수 있습니다. 자세한 내용은 Amazon Redshift 간 데이터 이동을 참조하세요.	2022년 11월 29일

AWS Glue 버전 4.0을 지원합니다.	AWS Glue 버전 4.0에 대한 지원 정보가 추가되었습니다. 기능으로는 Apache Hudi, Delta Lake 및 Apache Iceberg에서 개방형 데이터 레이크 프레임워크를 기본적으로 지원하고, Amazon S3를 사용하여 셔플링 및 탄력적인 스토리지 용량을 지원하기 위해 Amazon S3 기반 클라우드 셔플 스토리지 플러그인(Apache Spark 플러그인)을 기본적으로 지원합니다. 자세한 내용은 AWS Glue 릴리스 정보와 AWS Glue 버전 4.0으로 AWS Glue 작업 마이그레이션을 참조하세요.	2022년 11월 28일
AWS Glue Studio에서는 이제 사용자 지정 시각적 변환을 제공합니다.	고객은 사용자 지정 시각적 변환을 통해 팀 간에 비즈니스별 ETL 로직을 정의, 재사용, 공유할 수 있습니다. 자세한 내용은 사용자 지정 시각적 변환을 참조하세요.	2022년 11월 28일
AWS Glue 크롤러를 사용하여 JDBC 데이터 스토어용 메타데이터 게시 지원	이제 AWS Glue 크롤러를 사용하여 주석 및 원시 유형과 같은 메타데이터를 JDBC 데이터 스토어용 데이터 카탈로그에 게시할 수 있습니다. 자세한 내용은 크롤러가 데이터 카탈로그 테이블에 설정한 파라미터, 크롤러 속성 및 JDBCTarget 구조를 참조하세요.	2022년 11월 18일

Snowflake 데이터 스토어 크롤링 지원	이제 AWS Glue를 사용하여 Snowflake 테이블 및 뷰를 크롤링하고 메타데이터를 데이터 카탈로그에 테이블 항목으로 게시할 수 있습니다. Amazon S3의 Snowflake 외부 테이블의 경우 크롤러는 Amazon S3 위치 및 외부 테이블의 파일 형식 유형을 크롤링하고 테이블 파라미터로 채웁니다. 자세한 내용은 어떤 데이터 스토어를 크롤할 수 있나요?, AWS Glue 연결 속성 및 크롤러가 데이터 카탈로그 테이블에 설정한 파라미터를 참조하세요.	2022년 11월 18일
Spark 애플리케이션의 향상된 셔플 관리 지원	이제 Apache Spark의 새로운 클라우드 셔플 스토리지 플러그인을 지원할 수 있습니다. 자세한 내용은 Amazon S3의 AWS Glue Spark 셔플 플러그인 및 Apache Spark용 클라우드 셔플 스토리지 플러그인을 참조하세요.	2022년 11월 15일
크롤링 Amazon S3 이벤트 알림을 가속화할 때 Data Catalog 대상에 대한 지원이 추가됨	Amazon S3 대상에 대한 기존 지원과 더불어 이제 Amazon S3 이벤트 알림을 사용한 데이터 Data Catalog의 크롤링 가속화에 대한 지원이 제공됩니다. 자세한 내용은 Amazon S3 이벤트 알림을 사용하여 크롤링 가속화를 참조하세요.	2022년 10월 13일

크롤러가 생성할 수 있는 최대 테이블 수 지정 가능	이제 크롤러가 생성할 수 있는 최대 테이블 수를 지정할 수 있습니다. 자세한 내용을 알아보려면 크롤러가 생성할 수 있는 최대 테이블 수를 지정하는 방법을 참조하세요.	2022년 9월 6일
AWS Glue에서 Python 셸 작업의 Python 3.9 지원	AWS Glue의 Python 셸 작업에서 Python 3.9와 호환되는 스크립트 실행과 사용할 사전 패키징된 라이브러리 세트 선택에 대해 지원이 제공됩니다. 자세한 내용은 AWS Glue의 Python 셸 작업을 참조하세요.	2022년 8월 11일
예비 용량에 관한 긴급하지 않거나 시간에 민감하지 않은 AWS Glue 작업 실행 지원	이제 사전 프로덕션 작업, 테스트 및 일회성 데이터 로드와 같이 긴급하지 않은 작업에 대한 유연한 작업 실행 구성 지원이 제공됩니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2022년 8월 9일
스트리밍 작업에 대한 새 작업자 유형 지원	이제 볼륨이 낮은 스트리밍 작업에 대한 `G.025X` 작업자 유형의 사용이 지원됩니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2022년 7월 14일
AWS Glue 연결에서의 Kafka SASL 사용 지원	이제 AWS Glue 연결에서 Kafka SASL 사용이 지원됩니다. 자세한 내용은 클라이언트 인증을 위한 AWS Glue Kafka 연결 속성을 참조하세요.	2022년 7월 5일

Protobuf 스키마에 대한 Apache Kafka 커넥터 지원	Protobuf 스키마에 대한 Apache Kafka 커넥터가 이제 지원됩니다. 자세한 내용은 AWS Glue Schema Registry를 참조하세요.	2022년 6월 9일
AWS Glue 작업에서 Auto Scaling 지원(GA)	AWS Glue 버전 3.0 작업에 Auto Scaling을 사용하여 컴퓨팅 리소스 크기를 동적으로 조정하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue에서 Auto Scaling 사용을 참조하세요.	2022년 4월 14일
AWS Glue 작업 스크립트 AWS Glue 개발 및 테스트에 대한 설명서 업데이트	AWS Glue에 대한 사용 가능한 개발 및 테스트 방법에 대한 정보가 재구성 및 추가되었으며, Docker를 통해 개발하기 위한 지침을 포함합니다. 자세한 내용은 AWS Glue 작업 스크립트 개발 및 테스트를 참조하세요.	2022년 3월 14일
AWS Glue Schema Registry의 지원되는 데이터 형식으로 프로토콜 버퍼(Protobuf) 추가	AVRO 및 JSON 외에 지원되는 데이터 형식으로 Protobuf에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue Schema Registry를 참조하세요.	2022년 2월 25일
Delta Lake 테이블 크롤링에 대한 지원	AWS Glue를 사용하여 Delta Lake 테이블을 크롤링하는 작업에 대한 정보를 추가했습니다. 자세한 내용은 Delta Lake 데이터 스토어에 대한 구성 옵션을 지정하는 방법을 참조하십시오.	2022년 2월 24일

AWS Glue 작업 인사이트 지원	AWS Glue 작업 인사이트를 사용하여 작업 디버깅 및 AWS Glue 작업의 최적화를 간소화하는 방법이 추가되었습니다. 자세한 내용은 Monitoring with AWS Glue job insights(작업 인사이트를 사용한 모니터링)를 참조하세요.	2022년 2월 8일
VPC 엔드포인트를 사용한 Amazon S3 기반 데이터 카탈로그 테이블의 크롤링 지원	Amazon S3 데이터 스토어 외에도 Amazon S3 기반 데이터 카탈로그 테이블이 보안, 감사 또는 제어를 위해 Amazon Virtual Private Cloud 환경(Amazon VPC)에 의해서만 액세스되도록 구성할 수 있습니다. 자세한 내용은 Crawling an Amazon S3 Data Store or Amazon S3 backed Data Catalog tables using a VPC Endpoint(VPC 엔드포인트를 사용하여 Amazon S3 데이터 스토어 또는 Amazon S3 기반 데이터 카탈로그 테이블 크롤링)를 참조하세요.	2022년 2월 3일
Lake Formation 관리형 테이블 지원	ACID 트랜잭션, 자동 데이터 압축, 시간 이동 쿼리를 지원하는 Lake Formation 관리형 테이블에 대한 AWS Glue 지원 정보를 추가했습니다. 자세한 내용은 AWS Glue API 및 AWS Lake Formation 개발자 가이드를 참조하세요.	2021년 11월 30일

대화형 세션과 노트북에 대한 새로운 AWS 관리형 정책이 추가됨	대화형 세션과 노트북에 AWS Glue를 사용하기 위해 IAM에서 제공하는 향상된 보안에 대한 새로운 관리형 정책입니다. 자세한 내용은 AWS Glue에 대한 AWS 관리형 정책을 참조하세요.	2021년 11월 30일
이제 스트리밍 작업에서 Glue Schema Registry가 지원됨	Glue Schema Registry의 일부인 테이블에 액세스하는 스트리밍 작업을 생성할 수 있습니다. 자세한 내용은 AWS Glue Schema Registry 및 AWS Glue에서 스트리밍 ETL 작업 추가를 참조하세요.	2021년 11월 15일
새로운 기계 학습 기능 지원	증분 일치 및 일치 점수를 포함하여 일치 항목 찾기 기계 학습 변환의 새로운 기능에 대한 정보를 추가했습니다. 자세한 내용은 증분 일치 항목 찾기 및 일치 신뢰도 점수를 사용하여 일치 항목의 품질 추정을 참조하세요.	2021년 10월 31일
(프라이빗 평가판) AWS Glue 유연한 작업 지원	시작 및 완료 시간이 다를 수 있는 시간에 민감하지 않은 작업에 해당되는 유연한 실행 클래스를 사용한 AWS Glue Spark 작업 구성에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2021년 10월 29일

Amazon S3 이벤트 알림을 사용하여 크롤링 가속화 지원	Amazon S3 이벤트 알림을 사용하여 크롤링을 가속화하는 방법에 대한 정보를 추가했습니다. 자세한 내용은 Amazon S3 이벤트 알림을 사용하여 크롤링 가속화를 참조하세요.	2021년 10월 15일
액세스 제어 및 VPC와 관련된 추가 보안 구성 옵션	AWS Glue에서 새로운 액세스 제어 권한을 구성하는 방법 및 VPC 구성에 대한 정보를 추가했습니다. 자세한 내용은 AWS Glue의 AWS 태그, 조건 키 또는 컨텍스트 키를 사용하여 설정을 제어하는 자격 증명 기반 정책(IAM 정책), 모든 AWS 호출이 VPC를 통과하도록 구성을 참조하세요.	2021년 10월 13일
VPC 엔드포인트 정책 지원	AWS Glue의 Virtual Private Cloud(VPC) 엔드포인트 정책 지원에 대한 정보를 추가했습니다. 자세한 내용은 AWS Glue 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.	2021년 10월 11일
이제 중국에서 Glue Studio 사용 가능	이제 중국 베이징 및 닝샤 리전에서 AWS Glue Studio를 사용할 수 있습니다.	2021년 10월 11일

AWS Glue Studio에서는 대화형 작업 편집을 위한 노트북 작성 기능을 제공합니다.	노트북을 사용하여 코드를 작성 및 실행하고, 결과를 시각화하고, 인사이트를 공유할 수 있습니다. 일반적으로 데이터 과학자는 실험 및 데이터 탐색 작업에 노트북을 사용합니다. 자세한 내용은 노트북 사용을 참조하세요.	2021년 10월 1일
이제 스트리밍 소스에 직접 액세스 가능	시각적 편집기에서 ETL 작업에 데이터 원본을 추가할 때 데이터 카탈로그 데이터베이스와 테이블을 사용하지 않고 데이터 스트림에 액세스하기 위한 정보를 제공할 수 있습니다.	2021년 9월 30일
AWS Glue 버전 지원 정책이 문서화됨	AWS Glue 버전 지원 정책 및 특정 AWS Glue 버전의 수명 종료 단계에 대한 정보를 추가했습니다. 자세한 내용은 AWS Glue 버전 지원 정책을 참조하세요.	2021년 9월 24일
이제 데이터 미리 보기에서 사용자 지정 커넥터 사용 가능	사용자 지정 커넥터를 사용하여 데이터 원본 노드를 편집할 때 데이터 미리 보기(Data preview) 탭을 선택하여 데이터 집합을 미리 볼 수 있습니다. 자세한 내용은 사용자 지정 커넥터를 참조하세요.	2021년 9월 24일

AWS Glue 대화형 세션 지원(프라이빗 평가판)	(프라이빗 평가판) AWS Glue 대화형 세션을 사용하여 Jupyter Notebook에서 클라우드의 Spark 워크로드를 실행하는 방법에 대한 정보를 추가했습니다. 대화형 세션은 AWS Glue 2.0 이상을 사용할 때 AWS Glue 추출, 변환, 로드 코드를 개발하는 데 선호되는 방법입니다. 자세한 내용은 Jupyter Notebook에 대해 AWS Glue 대화형 세션 설정 및 실행을 참조하세요.	2021년 8월 24일
블루프린트에서 워크플로 생성 지원(GA)	블루프린트에서 일반적인 추출, 변환, 로드 사용 사례를 코딩한 다음 블루프린트에서 워크플로를 생성하는 방법에 대한 정보가 추가되었습니다. 데이터 분석가가 복잡한 ETL 프로세스를 쉽게 생성하고 실행할 수 있습니다. 자세한 정보는 AWS Glue에서 블루프린트와 워크플로를 사용하여 복잡한 ETL 활동 수행을 참조하세요.	2021년 8월 23일

AWS Glue 버전 3.0을 지원합니다.	Apache Spark ETL 작업 실행을 위한 Apache Spark 3.0 엔진 업그레이드, 기타 최적화 및 업그레이드를 지원하는 AWS Glue 버전 3.0에 대한 지원 정보가 추가되었습니다. 자세한 내용은 AWS Glue 릴리스 정보와 AWS Glue 버전 3.0으로 AWS Glue 작업 마이그레이션을 참조하세요. 이 릴리스의 다른 기능에는 AWS Glue 셔플 관리자, SIMD 벡터화 CSV 리더 및 카탈로그 파티션 조건자가 있습니다. 자세한 내용은 Amazon S3의 AWS Glue Spark 셔플 관리자, AWS Glue의 ETL 입력 및 출력의 포맷 옵션 및 카탈로그 파티션 조건자를 사용한 서버 측 필터링을 참조하세요.	2021년 8월 18일
AWS GovCloud (US) Region	이제 AWS GovCloud (US) Region에서 AWS Glue Studio를 사용할 수 있습니다.	2021년 8월 18일
AWS Glue Studio에서 Python 셸 작성 사용 가능	새 작업을 생성할 때 이제 Python 셸 작업을 생성하도록 선택할 수 있습니다. 자세한 내용은 작업 생성 프로세스 시작 및 AWS Glue Studio에서 Python 셸 작업 편집을 참조하세요.	2021년 8월 13일

Amazon EventBridge 이벤트로 워크플로 시작 지원	이벤트 중심 아키텍처에서 AWS Glue가 이벤트 소비자가 될 수 있는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 Amazon EventBridge 이벤트로 AWS Glue 워크플로 시작 및 워크플로를 시작한 EventBridge 이벤트 보기를 참조하세요.	2021년 7월 14일
AWS Glue Schema Registry의 지원되는 데이터 포맷으로 JSON 추가	AVRO 외에 지원되는 데이터 포맷으로서의 JSON에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue Schema Registry를 참조하세요.	2021년 6월 30일
데이터 카탈로그 테이블 없이 AWS Glue 스트리밍 작업 생성	`create_data_frame_from_options` Python 함수 또는 Scala 스크립트용 `getSource` 는 데이터 카탈로그 테이블을 요구하는 대신 데이터 스트림을 직접 참조하는 스트리밍 ETL 작업 생성을 지원합니다.	2021년 6월 15일
AWS Glue Machine Learning 변환에서 이제 AWS Key Management Service 키 지원	콘솔, CLI 또는 AWS Glue API를 사용하여 AWS Glue Machine Learning 변환을 구성할 때 보안 구성 또는 AWS KMS 키를 지정할 수 있습니다. 자세한 내용은 Machine Learning 변환에 데이터 암호화 사용 및 AWS Glue Machine Learning API를 참조하세요.	2021년 6월 15일

AWSGlueConsoleFullAccess AWS 관리형 정책 업데이트	AWSGlueConsoleFullAccess AWS 관리형 정책의 마이너 업데이트에 대한 정보가 추가되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 AWS Glue 업데이트를 참조하세요.	2021년 6월 10일
작업을 생성하고 편집하는 동안 작업의 데이터 세트 보기	작업 다이어그램의 노드에 대한 새로운 [데이터 미리 보기 (Data preview)] 탭을 사용하여 해당 노드에서 처리된 데이터의 샘플을 볼 수 있습니다. 자세한 내용은 시각적 작업 편집기에서 데이터 미리 보기 사용을 참조하세요.	2021년 6월 7일
크롤러 출력의 테이블 위치를 나타내는 값 지정 지원	크롤러의 출력을 구성할 때 테이블 위치를 나타내는 값 지정에 대한 정보가 추가되었습니다. 자세한 내용은 테이블 위치를 지정하는 방법을 참조하세요.	2021년 6월 4일
Amazon S3 데이터 스토어를 크롤링할 때 데이터 집합의 파일 샘플 크롤링 지원	Amazon S3를 크롤링할 때 파일 샘플을 크롤링하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 크롤러 속성을 참조하십시오.	2021년 5월 10일

AWS Glue 최적화 parquet 라이터 지원	DynamicFrames에 AWS Glue 최적화 parquet 라이터를 사용하여 `parquet` 분류가 있는 테이블을 생성하거나 업데이트하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue ETL 작업의 데이터 카탈로그에서 테이블 생성, 스키마 업데이트 및 새 파티션 추가 및 AWS Glue의 ETL 입력 및 출력 포맷 옵션을 참조하세요.	2021년 5월 4일
Kafka 클라이언트 인증 암호 지원	AWS Glue의 스트리밍 ETL 작업이 Apache Kafka 스트림 생성자를 사용한 SSL 클라이언트 인증서 인증을 지원하는 방법에 대한 정보가 추가되었습니다. 이제 인증할 때 AWS Glue가 사용하는 Apache Kafka 클러스터에 대한 AWS Glue 연결을 정의하는 동안 사용자 정의 인증서를 제공할 수 있습니다. 자세한 내용은 AWS Glue 연결 속성 및 연결 API를 참조하세요.	2021년 4월 28일
스트리밍 ETL 작업에서 다른 계정의 Amazon Kinesis Data Streams 데이터 사용 지원	다른 계정의 Amazon Kinesis Data Streams에서 데이터를 사용하기 위해 스트리밍 ETL 작업을 생성하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue에서 스트리밍 ETL 작업 추가를 참조하십시오.	2021년 3월 30일

SQL 변환 사용 가능	SQL 변환 노드를 사용하여 SQL 쿼리 형식으로 고유한 변환을 작성할 수 있습니다. 자세한 내용은 SQL 쿼리를 사용하여 데이터 변환을 참조하세요.	2021년 3월 23일
블루프린트에서 워크플로 생성 지원(공개 평가판)	(공개 미리 보기) 블루프린트에서 일반적인 추출, 변환, 로드 사용 사례를 코딩한 다음 블루프린트에서 워크플로를 생성하는 방법에 대한 정보가 추가되었습니다. 데이터 분석가가 복잡한 ETL 프로세스를 쉽게 생성하고 실행할 수 있습니다. 자세한 정보는 AWS Glue에서 블루프린트와 워크플로를 사용하여 복잡한 ETL 활동 수행을 참조하세요.	2021년 3월 22일
데이터 대상에 커넥터 사용 가능	이제 데이터 대상에 대해 사용자 정의 또는 AWS Marketplace 커넥터를 사용할 수 있습니다. 자세한 내용은 사용자 정의 커넥터로 작업 작성을 참조하세요.	2021년 3월 15일
AWS Glue 기계 학습 변환을 위한 열 중요도 지표 지원	AWS Glue 기계 학습 변환 작업 시 열 중요도 지표 보기에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 콘솔에서 Machine Learning 변환 작업을 참조하세요.	2021년 2월 5일

이제 AWS Glue Studio에서 작업 예약 사용 가능	AWS Glue Studio에서 시간 기반의 작업 실행 일정을 정의할 수 있습니다. 콘솔을 사용하여 기본 일정을 생성하거나 Unix와 같은 cron 구문을 사용하여 더 복잡한 일정을 정의할 수 있습니다. 자세한 내용은 작업 실행 예약을 참조하세요.	2020년 12월 21일
AWS Glue 사용자 정의 커넥터 출시	AWS Glue 사용자 정의 커넥터를 사용하면 AWS Marketplace에서 커넥터를 검색하고 구독할 수 있습니다. Apache Spark Datasource, Athena 연합 쿼리 및 JDBC API용으로 구축된 커넥터를 플러그 인하기 위한 AWS Glue Spark 런타임 인터페이스도 출시되었습니다. 자세한 내용은 AWS Glue Studio에서 커넥터 및 연결 사용을 참조하세요.	2020년 12월 21일
AWS Glue 버전 2.0에서 스트리밍 ETL 작업 실행 지원	Glue 버전 2.0에서 스트리밍 ETL 작업 실행 지원에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue에서 스트리밍 ETL 작업 추가를 참조하십시오.	2020년 12월 18일
제한된 실행으로 워크로드 분할 지원	워크로드 분할을 사용하여 데이터 집합 크기의 상한 또는 ETL 작업 실행에서 처리되는 파일 수를 구성하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 제한된 실행으로 워크로드 분할을 참조하세요.	2020년 11월 23일

향상된 파티션 관리 지원	새 API를 사용하여 기존 테이블에 파티션 인덱스를 추가하거나 삭제하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 파티션 인덱스 작업을 참조하세요.	2020년 11월 23일
AWS Glue Schema Registry 지원	AWS Glue Schema Registry를 사용하여 스키마를 중앙에서 검색, 제어 및 발전시키는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue Schema Registry를 참조하세요.	2020년 11월 19일
스트리밍 ETL 작업에서 Grok 입력 형식 지원	로그 파일과 같은 스트리밍 소스에 Grok 패턴을 적용하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 스트리밍 소스에 Grok 패턴 적용을 참조하세요.	2020년 11월 17일
AWS Glue 콘솔에서 워크플로에 태그 추가 지원	AWS Glue 콘솔을 사용하여 워크플로를 생성할 때 태그를 추가하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 콘솔을 사용하여 워크플로 생성 및 구축을 참조하세요.	2020년 10월 27일
증분 크롤러 실행 지원	마지막 실행 이후 추가된 Amazon S3 폴더만 크롤링하는 증분 크롤러 실행 지원에 대한 정보가 추가되었습니다. 자세한 내용은 증분 크롤을 참조하세요.	2020년 10월 21일

스트리밍 ETL 데이터 소스에 대한 스키마 감지 지원. Avro 스트리밍 ETL 데이터 소스 및 자체 관리형 Kafka 지원	AWS Glue의 스트리밍 추출, 변환, 로드 작업은 이제 수신 레코드의 스키마를 자동으로 감지하고 레코드별로 스키마 변경을 처리할 수 있습니다. 이제 자체 관리형 Kafka 데이터 원본이 지원됩니다. 스트리밍 ETL 작업은 이제 데이터 원본에서 Avro 포맷을 지원합니다. 자세한 내용은 AWS Glue의 스트리밍 ETL, 스트리밍 ETL 작업에 대한 작업 속성 정의 및 Avro 스트리밍 소스에 대한 참고 사항 및 제한 사항을 참조하세요.	2020년 10월 7일
MongoDB 및 DocumentDB 데이터 소스 크롤링 지원	MongoDB 및 Amazon DocumentDB(MongoDB 호환) 데이터 원본 크롤링 지원에 대한 정보가 추가되었습니다. 자세한 내용은 크롤러 정의를 참조하세요.	2020년 10월 5일
FIPS 규정 준수 지원	AWS Glue를 사용하여 데이터에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 고객을 위한 FIPS 엔드포인트에 대한 정보가 추가되었습니다. 자세한 내용은 FIPS 규정 준수를 참조하세요.	2020년 9월 23일

AWS Glue Studio에서는 작업 생성 및 모니터링을 위한 사용하기 쉬운 시각적 인터페이스를 제공합니다.	이제 간단한 그래프 기반 인터페이스를 사용하여 데이터를 이동 및 변환하고 AWS Glue에서 실행하는 작업을 작성할 수 있습니다. 그런 다음 AWS Glue Studio의 작업 실행 대시보드를 사용하여 ETL 실행을 모니터링하고 작업이 의도한 대로 작동하는지 확인할 수 있습니다. 자세한 내용은 AWS Glue Studio 사용 설명서를 참조하세요.	2020년 9월 23일
쿼리 성능 향상을 위한 테이블 인덱스 생성 지원	테이블에서 파티션의 하위 집합을 검색할 수 있도록 테이블 인덱스 생성에 대한 정보가 추가되었습니다. 자세한 내용은 파티션 인덱스 작업을 참조하세요.	2020년 9월 9일
AWS Glue 버전 2.0에서 Apache Spark ETL 작업을 실행할 때 시작 시간 단축 지원.	시작 시간 단축, 로깅 변경, 작업 수준에서 추가 Python 모듈 지정 지원과 함께 Apache Spark ETL 작업 실행을 위한 업그레이드된 인프라를 제공하는 AWS Glue 버전 2.0 지원에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 릴리스 정보 및 단축된 시작 시간으로 Spark ETL 작업 실행을 참조하세요.	2020년 8월 10일

동시 워크플로 실행 수 제한 지원	특정 워크플로에 대한 동시 워크플로 실행 수를 제한하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 콘솔을 사용하여 워크플로 생성 및 구축을 참조하세요.	2020년 8월 10일
VPC 엔드포인트를 사용한 Amazon S3 데이터 스토어 크롤링 지원	보안, 감사 또는 제어 목적으로 Amazon Virtual Private Cloud 환경(Amazon VPC)에서만 액세스할 수 있도록 Amazon S3 데이터 스토어를 구성하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 VPC 엔드포인트를 사용하여 Amazon S3 데이터 스토어 크롤링을 참조하세요.	2020년 8월 7일
워크플로 실행 재개 지원	하나 이상의 노드(작업 또는 크롤러)가 성공적으로 완료되지 않아 부분적으로만 완료된 워크플로 실행을 재개하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 워크플로 실행 복구 및 재개를 참조하세요.	2020년 7월 27일
AWS Glue의 Kafka 연결에서 프라이빗 CA 인증서 사용 지원	AWS Glue에서 Kafka 연결에 대한 프라이빗 CA 인증서 사용을 지원하는 새로운 연결 옵션에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue에서 ETL 관련 연결 유형 및 옵션과 AWS Glue가 사용하는 특정 파라미터를 참조하세요.	2020년 7월 20일

다른 계정의 DynamoDB 데이터 읽기 지원	다른 AWS 계정의 DynamoDB 테이블에서 데이터 읽기에 대한 AWS Glue 지원에 대한 정보가 추가되었습니다. 자세한 내용은 다른 계정의 DynamoDB 데이터에서 읽기를 참조하세요.	2020년 7월 17일
AWS Glue 버전 1.0 이상에서 DynamoDB 라이터 연결 지원	DynamoDB 라이터 지원에 대한 정보와 DynamoDB에서 읽거나 쓸 수 있는 새로운 연결 옵션 또는 업데이트된 연결 옵션이 추가되었습니다. 자세한 내용은 AWS Glue의 ETL 연결 유형 및 옵션을 참조하세요.	2020년 7월 17일
AWS Glue 및 Lake Formation을 모두 사용하여 리소스 링크 및 교차 계정 액세스 제어 지원	리소스 링크라는 새 데이터 카탈로그 객체에 대한 내용과 AWS Glue 및 AWS Lake Formation이 모두 있는 계정 간에 데이터 카탈로그 리소스 공유를 관리하는 방법에 대한 내용이 추가되었습니다. 자세한 내용은 교차 계정 액세스 권한 부여 및 테이블 리소스 링크를 참조하세요.	2020년 7월 7일
DynamoDB 데이터 스토어를 크롤링할 때 레코드 샘플링 지원	DynamoDB 데이터 저장소를 크롤링할 때 구성할 수 있는 새 속성에 대한 정보가 추가되었습니다. 자세한 내용은 크롤러 속성을 참조하십시오.	2020년 6월 12일

워크플로우 실행 중지 지원	특정 워크플로우에 대한 워크플로우 실행을 중지하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 워크플로우 실행 중지를 참조하십시오.	2020년 5월 14일
Spark 스트리밍 ETL 작업 지원	스트리밍 데이터 원본을 사용하여 추출, 변환 및 로드(ETL) 작업을 생성하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue에서 스트리밍 ETL 작업 추가를 참조하십시오.	2020년 4월 27일
ETL 작업 실행 후 데이터 카탈로그에서 테이블 생성, 스키마 업데이트, 새 파티션 추가 지원	데이터 카탈로그에서 테이블 생성, 스키마 업데이트 및 새 파티션 추가를 활성화하여 ETL 작업의 결과를 확인하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue ETL 작업의 데이터 카탈로그에서 테이블 생성, 스키마 업데이트 및 새 파티션 추가를 참조하세요.	2020년 4월 2일
AWS Glue에서 Apache Avro 데이터 형식을 ETL 입력 및 출력으로 버전 지정 지원	AWS Glue에서 Apache Avro 데이터 포맷을 ETL 입력 및 출력으로 버전을 지정하는 것에 대한 정보가 추가되었습니다. 기본 버전은 1.7입니다. `version` 포맷 옵션을 사용해 Avro 버전 1.8을 지정하여 논리적 읽기/쓰기를 활성화할 수 있습니다. 자세한 내용은 AWS Glue에서 ETL 입력 및 출력의 포맷 옵션을 참조하세요.	2020년 3월 31일

Amazon S3에 Parquet 데이터를 쓸 수 있도록 EMRFS S3 최적화 커미터 지원	AWS Glue 작업을 생성하거나 업데이트할 때 Amazon S3에 Parquet 데이터를 쓸 수 있도록 새 플래그를 설정해 EMRFS S3 최적화 커미터를 활성화하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue가 사용하는 특정 파라미터를 참조하십시오.	2020년 3월 30일
AWS 리소스 태그로 관리되는 리소스로 기계 학습 변환 지원	AWS Glue에서 AWS 리소스 태그를 사용하여 기계 학습 변환에 대한 액세스를 관리하고 제어하는 방법에 대한 정보가 추가되었습니다. AWS Glue의 작업, 트리거, 엔드포인트, 크롤러 및 기계 학습 변환에 AWS 리소스 태그를 할당할 수 있습니다. 자세한 내용은 AWS Glue의 AWS 태그를 참조하세요.	2020년 3월 2일
재정의할 수 없는 작업 인수 지원	트리거에서 또는 작업을 실행할 때 재정의할 수 없는 특수 작업 파라미터에 대한 지원 정보가 추가되었습니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2020년 2월 12일

Amazon S3의 데이터 집합을 사용할 수 있도록 새로운 변환 지원	Apache Spark 애플리케이션에서 Amazon S3의 데이터 집합을 사용할 수 있도록 새로운 변환(병합, 제거 및 전환) 및 Amazon S3 스토리지 클래스 제외 사항에 대한 정보가 추가되었습니다. Python의 경우 이러한 변환 지원에 대한 자세한 내용은 mergeDynamicFrame 및 Amazon S3에서 데이터 집합 작업을 참조하세요. Scala의 경우는 mergeDynamicFrames 및 AWS Glue Scala GlueContext API를 참조하세요.	2020년 1월 16일
ETL 작업에서 새 파티션 정보로 데이터 카탈로그 업데이트 지원	ETL(추출, 변환 및 로드) 스크립트를 코딩하여 새 파티션 정보로 AWS Glue Data Catalog를 업데이트하는 방법에 대한 정보가 추가되었습니다. 이 기능을 사용하면 새 파티션을 확인하기 위해 작업 완료 후 크롤러를 다시 실행할 필요가 없습니다. 자세한 내용은 새 파티션으로 데이터 카탈로그 업데이트를 참조하세요.	2020년 1월 15일
새 자습서: SageMaker AI 노트북 사용	Amazon SageMaker 노트북을 사용하여 ETL 및 기계 학습 스크립트를 개발하는 방법을 보여주는 자습서가 추가되었습니다. 자습서: 개발 엔드포인트와 함께 Amazon SageMaker 노트북 사용을 참조하십시오.	2020년 1월 3일

MongoDB 및 Amazon DocumentDB(MongoDB 호환)에서의 읽기 지원	MongoDB 및 Amazon DocumentDB(MongoDB와 호환)에서의 읽기 및 쓰기에 대한 새 연결 유형 및 연결 옵션에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue의 ETL 연결 유형 및 옵션을 참조하세요.	2019년 12월 17일
다양한 수정 및 설명	전체적으로 수정 및 설명을 추가했습니다. 알려진 문제 장에서 항목을 제거했습니다. 데이터 카탈로그 암호화 설정을 지정하고 보안 구성을 생성할 때 AWS Glue가 대칭 고객 마스터 키(CMK)만 지원하는 경고가 추가되었습니다. AWS Glue가 Amazon DynamoDB에 쓰기를 지원하지 않는다는 메모를 추가했습니다.	2019년 12월 9일
사용자 지정 JDBC 드라이버 지원	MySQL 버전 8 및 Oracle Database 버전 18과 같이 AWS Glue에서 기본적으로 지원하지 않는 JDBC 드라이버로 데이터 원본과 대상에 연결하는 데 대한 정보가 추가되었습니다. 자세한 내용은 JDBC connectionType 값을 참조하십시오.	2019년 11월 25일

SageMaker AI 노트북을 다른 개발 엔드포인트에 연결하도록 지원	SageMaker AI 노트북을 다른 개발 엔드포인트에 연결하는 방법에 대한 정보가 추가되었습니다. 새로운 개발 엔드포인트로 전환하기 위한 새로운 콘솔 작업 및 새로운 SageMaker AI IAM 정책을 설명하기 위한 업데이트입니다. 자세한 내용은 AWS Glue 콘솔에서 노트북 작업 및 Amazon SageMaker AI 노트북용 IAM 정책 생성을 참조하세요.	2019년 11월 21일
기계 학습 변환에서 AWS Glue 버전 지원	AWS Glue의 어떤 버전이 기계 학습 변환과 호환되는지 나타내기 위해 기계 학습 변환에서 AWS Glue 버전을 정의하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 콘솔에서 기계 학습 변환 작업을 참조하세요.	2019년 11월 21일
작업 북마크 되돌리기 지원	이전 작업 실행으로 작업 북마크를 되돌리는 방법에 대한 정보를 추가했습니다. 이제 후속 작업 실행 시 북마크로 지정된 작업 실행의 데이터만 다시 처리합니다. 두 북마크 간에 작업을 실행할 수 있도록 허용하는 `job-bookmark-pause` 옵션에 대한 두 개의 새로운 하위 옵션 설명이 있습니다. 자세한 내용은 작업 북마크를 사용해 처리된 데이터 추적 및 AWS Glue에서 사용되는 특정 파라미터를 참조하세요.	2019년 10월 22일

데이터 스토어에 연결하기 위한 사용자 지정 JDBC 인증서 지원	AWS Glue 데이터 소스 또는 대상에 SSL을 연결하기 위해 AWS Glue의 사용자 지정 JDBC 인증서 지원에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 콘솔에서 연결 작업을 참조하세요.	2019년 10월 10일
Python Wheel 지원	Python 셸 작업에 대한 종속성으로 AWS Glue의 wheel 파일(egg 파일과 함께) 지원에 대한 정보가 추가되었습니다. 자세한 내용은 자체 Python 라이브러리 제공을 참조하십시오.	2019년 9월 26일
AWS Glue에서 개발 엔드포인트의 버전 관리 지원	개발 엔드포인트에서 `Glue version`을 정의하는 방법에 대한 정보가 추가되었습니다. `Glue version`은 AWS Glue에서 지원하는 Apache Spark 및 Python의 버전을 결정합니다. 자세한 내용은 개발 엔드포인트 추가를 참조하십시오.	2019년 9월 19일
Spark UI를 사용한 AWS Glue 모니터링 지원	Apache Spark UI를 사용하여 AWS Glue 작업 시스템에서 실행 중인 AWS Glue ETL 작업과 AWS Glue 개발 엔드포인트의 Spark 애플리케이션을 모니터링하고 디버그하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 Spark UI를 사용한 AWS Glue 모니터링을 참조하세요.	2019년 9월 19일

퍼블릭 AWS Glue ETL 라이브러리를 사용해 로컬 ETL 스크립트를 개발할 수 있도록 지원 개선	AWS Glue 버전 1.0이 현재 지원되고 있다는 것을 반영하기 위해 AWS Glue ETL 라이브러리 콘텐츠가 업데이트되었습니다. 자세한 내용은 AWS Glue ETL 라이브러리를 사용해 로컬에서 ETL 스크립트 개발 및 테스트를 참조하세요.	2019년 9월 18일
작업 실행 시 Amazon S3 스토리지 클래스를 제외할 수 있도록 지원	Amazon S3에서 파일 또는 파티션을 읽어오는 AWS Glue ETL 작업을 실행할 때 Amazon S3 스토리지 클래스를 제외하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 Amazon S3 스토리지 클래스 제외를 참조하세요.	2019년 8월 29일
퍼블릭 AWS Glue ETL 라이브러리를 사용해 로컬 ETL 스크립트를 개발할 수 있도록 지원	네트워크에 연결할 필요 없이 로컬에서 Python 및 Scala ETL 스크립트를 개발하고 테스트하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue ETL 라이브러리를 사용해 로컬에서 ETL 스크립트 개발 및 테스트를 참조하세요.	2019년 8월 28일
알려진 문제	AWS Glue의 알려진 문제에 대한 정보가 추가되었습니다. 자세한 내용은 알려진 AWS Glue 문제 단원을 참조하십시오.	2019년 8월 28일

AWS Glue에서 기계 학습 변환 지원	사용자 지정 변환 생성을 위해 AWS Glue에서 제공하는 기계 학습 기능에 대해 정보를 추가했습니다. 작업을 생성할 때 이러한 변환을 만들 수 있습니다. 자세한 내용은 AWS Glue에서의 기계 학습 변환을 참조하십시오.	2019년 8월 8일
공유 Amazon Virtual Private Cloud 지원	공유 Amazon Virtual Private Cloud에 대한 AWS Glue 지원 관련 정보가 추가되었습니다. 자세한 내용은 공유 Amazon VPC를 참조하십시오.	2019년 8월 6일
AWS Glue에서 버전 관리 지원	작업 속성에서 `Glue version`을 정의하는 방법에 대한 정보가 추가되었습니다. AWS Glue 버전은 AWS Glue에서 지원하는 Apache Spark 및 Python의 버전을 결정합니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2019년 7월 24일
개발 엔드포인트의 추가 구성 옵션 지원	메모리 집약적 워크로드가 있는 개발 엔드포인트의 구성 옵션에 대한 정보를 추가했습니다. 실행기당 더 많은 메모리를 제공하는 두 가지 새로운 구성 중에서 선택할 수 있습니다. 자세한 내용은 AWS Glue 콘솔상의 개발 엔드포인트 작업을 참조하십시오.	2019년 7월 24일

워크플로를 사용한 ETL(추출, 전송, 로드) 활동 수행 지원	AWS Glue가 단일 엔터티로 실행하고 추적할 수 있는 복잡한 다중 작업 추출, 변환, 로드 활동을 설계하기 위해 워크플로라는 새로운 구조 사용에 대한 정보를 추가했습니다. 자세한 내용은 AWS Glue에서 워크플로를 사용하여 복잡한 ETL 활동 수행을 참조하세요.	2019년 6월 20일
Python 셸 작업의 Python 3.6 지원	Python 쉘 작업의 Python 3.6 지원에 대한 정보를 추가했습니다. Python 2.7 또는 Python 3.6을 작업 속성으로 지정할 수 있습니다. 자세한 내용은 AWS Glue에서 Python 셸 작업 추가를 참조하세요.	2019년 6월 5일
Virtual Private Cloud(VPC) 엔드포인트 지원	VPC에서 인터페이스 엔드포인트를 통해 AWS Glue에 직접 연결하는 방법에 대한 정보가 추가되었습니다. VPC 인터페이스 엔드포인트를 사용하는 경우 VPC와 AWS Glue 간의 통신은 AWS 네트워크에서 완전하고 안전하게 수행됩니다. 자세한 내용은 VPC 엔드포인트와 함께 AWS Glue 사용을 참조하세요.	2019년 6월 4일

AWS Glue 작업에 대한 실시간 지속 로깅 지원	드라이버 로그, 각 실행기 로그 및 Spark 작업 진행률 표시줄을 포함한 CloudWatch의 실시간 Apache Spark 작업 로그를 활성화하고 보는 작업에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue 작업에 대한 지속 로깅을 참조하십시오.	2019년 5월 28일
크롤러 소스로 기존 데이터 카탈로그 테이블 지원	크롤러 소스로 기존 데이터 카탈로그 테이블의 목록 지정에 대한 정보가 추가되었습니다. 이제 크롤러는 테이블 스키마에 대한 변경 사항을 감지하고 테이블 정의를 업데이트하며 새 데이터를 사용할 수 있게 되면 새 파티션을 등록할 수 있습니다. 자세한 내용은 크롤러 속성을 참조하십시오.	2019년 5월 10일
메모리 집약적 작업에 대한 추가 구성 옵션 지원	메모리 집약적 워크로드가 있는 Apache Spark 작업의 구성 옵션에 대한 정보를 추가했습니다. 실행기당 더 많은 메모리를 제공하는 두 가지 새로운 구성 중에서 선택할 수 있습니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2019년 4월 5일
CSV 사용자 지정 분류자 지원	사용자 지정 CSV 분류자를 사용하여 다양한 유형의 CSV 데이터 스키마 추론에 대한 정보를 추가했습니다. 자세한 내용은 사용자 지정 분류자 작성을 참조하십시오.	2019년 3월 26일

AWS 리소스 태그 지원	AWS Glue 리소스에 대한 액세스를 관리 및 제어하는 데 도움이 되도록 AWS 리소스 태그의 사용 방법에 대한 정보가 추가되었습니다. AWS Glue의 작업, 트리거, 엔드포인트 및 크롤러에 AWS 리소스 태그를 할당할 수 있습니다. 자세한 내용은 AWS Glue의 AWS 태그를 참조하세요.	2019년 3월 20일
Spark SQL 작업에 데이터 카탈로그 지원	AWS Glue Data Catalog를 외부 Apache Hive 메타스토어로 사용하도록 AWS Glue 작업 및 개발 엔드포인트를 구성하는 방법에 대한 정보를 추가했습니다. 이렇게 하면 작업 및 개발 엔드포인트가 AWS Glue Data Catalog에 저장된 테이블에 대해 Apache Spark SQL 쿼리를 직접 실행할 수 있습니다. 자세한 내용은 Spark SQL 작업에 대한 AWS Glue Data Catalog 지원을 참조하세요.	2019년 3월 14일
Python 셸 작업 지원	Python 셸 작업과 새로운 필드 Maximum capacity(최대 용량)에 대한 정보가 추가되었습니다. 자세한 내용은 AWS Glue에서 Python 셸 작업 추가를 참조하세요.	2019년 1월 18일

데이터베이스와 테이블을 변경할 때 알림 지원	데이터베이스, 테이블 및 파티션 API 직접 호출의 변경 사항에 대해 생성된 이벤트 정보가 추가되었습니다. 이러한 이벤트에 응답하도록 CloudWatch Events에서 작업을 구성할 수 있습니다. 자세한 내용은 CloudWatch Events를 사용한 AWS Glue 자동화를 참조하세요.	2019년 1월 16일
연결 암호 암호화 지원	연결 객체에 사용되는 암호를 암호화하는 방법에 대한 정보를 추가했습니다. 자세한 내용은 Encrypting Connection Passwords 단원을 참조하십시오.	2018년 12월 11일
리소스 수준 권한 및 리소스 기반 정책에 대한 지원	AWS Glue에서 리소스 수준 권한 및 리소스 기반 정책을 사용하는 방법에 대한 정보를 추가했습니다. 자세한 내용은 AWS Glue의 보안에 수록된 주제를 참조하십시오.	2018년 10월 15일
SageMaker AI 노트북 지원	AWS Glue 개발 엔드포인트에서 SageMaker AI 노트북을 사용하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 노트북 관리를 참조하십시오.	2018년 10월 5일
암호화 지원	AWS Glue의 암호화를 사용하는 방법에 대한 정보를 추가했습니다. 자세한 내용은 저장 데이터 암호화, 전송 데이터 암호화 및 AWS Glue의 암호화 설정을 참조하세요.	2018년 8월 24일

Apache Spark 작업 지표 지원	ETL 작업의 디버깅 및 프로파일링 향상을 위해 Apache Spark 측정치 사용에 대한 정보를 추가했습니다. 드라이버 및 실행기의 읽고 쓴 바이트 수, 메모리 사용량 및 CPU 부하 같은 실행 시간 측정치와, AWS Glue 콘솔의 실행기 간의 데이터 셔플을 쉽게 추적할 수 있습니다. 자세한 내용은 CloudWatch 지표를 사용하여 AWS Glue 모니터링, 작업 모니터링 및 디버깅 및 AWS Glue 콘솔에서 작업 처리를 참조하세요.	2018년 7월 13일
데이터 원본으로서 DynamoDB 지원	DynamoDB 크롤링 및 이를 ETL 작업 데이터 원본으로 사용하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 크롤러를 사용하여 테이블 카탈로그 작성 및 연결 파라미터를 참조하십시오.	2018년 7월 10일
노트북 서버 생성 절차 업데이트	개발 엔드포인트와 연결된 Amazon EC2 인스턴스에서 노트북 서버를 생성하는 방법에 대한 정보가 업데이트되었습니다. 자세한 내용은 개발 엔드포인트와 연결된 노트북 서버 생성 단원을 참조하십시오.	2018년 7월 9일
RSS에서 현재 사용 가능한 업데이트	이제 AWS Glue Developer Guide에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.	2018년 6월 25일

작업 지연 알림 지원	작업 중의 지연 임곗값을 구성하는 내용이 추가됩니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2018년 5월 25일
크롤러를 구성하여 새 열 추가	크롤러, MergeNewColumns의 새 구성 옵션에 대한 정보를 추가했습니다. 더 자세한 내용은 크롤러 구성하기를 참조하십시오.	2018년 5월 7일
작업 제한 시간 지원	작업 중의 타임아웃 임곗값을 설정하는 내용이 추가됩니다. 자세한 내용은 AWS Glue의 작업 추가를 참조하십시오.	2018년 10월 4일
추가 실행 상태를 기반으로 Scala ETL 스크립트 및 트리거 작업 지원	ETL 프로그래밍 언어로써 사용되는 Scala에 대한 자세한 내용이 포함되었습니다. 또한, 트리거 API는 현재 (모든 조건과 더불어) 조건이 맞으면 API의 시작을 지원합니다. 작업도 ("succeeded" 작업 실행과 더불어) "failed" 또는 "stopped" 작업 실행을 기반으로 시작됩니다.	2018년 1월 12일

이전 업데이트

다음 표에서는 2018년 1월 이전 AWS Glue 개발자 안내서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	날짜
XML 데이터 원본 및 새로운 크롤러 구성 옵션 지원	파티션 변경에 따른 XML 데이터 원본 및 새로운 크롤러 옵션을 분류하는 추가된 정보.	2017년 11월 16일
새로운 변환, 추가 Amazon RDS 데이터베이스 엔진 지원 및 개발 엔드포인트 개선	맵 및 필터 변환, Amazon RDS Microsoft SQL Server 및 Amazon RDS Oracle에 대한 지원, 개발 엔드포인트를 위한 새로운 기능에 대한 정보가 추가되었습니다.	2017년 9월 29일
AWS Glue 최초 릴리스	AWS Glue Developer Guide가 처음으로 릴리스되었습니다.	2017년 8월 14일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하십시오.