SSIS ETL 작업을 로 마이그레이션 AWS

AWS 권장 가이드

AWS 권장 가이드: SSIS ETL 작업을 로 마이그레이션 AWS

Table of Contents

SSIS ETL 작업을 로 마이그레이션 AWS

Durga Prasad 및 Harpreet Singh, Amazon Web Services(AWS)

2021년 12월(문서 기록)

Amazon Web Services(AWS)는 추출, 변환 및 로드(ETL) 또는 추출, 로드 및 변환(ELT) 작업과 빅 데이터 워크로드를 개발하고 실행하기 위한 서비스를 제공합니다. Microsoft SQL Server Integration Services(SSIS)는 ETL 작업을 빌드하기 위한 그래픽 인터페이스가 있는 온프레미스 ETL 도구입니다. 대상 상태 아키텍처에 따라 AWS Glue 그래픽 인터페이스 또는 Python 라이브러리와 함께 Apache Spark 프레임워크를 사용하여 AWS 클라우드에서 ETL 작업을 빌드할 수 있습니다.

이 가이드에서는 온프레미스에서 로 SSIS ETL/ELT 작업을 마이그레이션하는 단계를 설명합니다 AWS. 마이그레이션 작업을 줄이고 경험을 개선하기 위한 마이그레이션 단계, 모범 사례 및 권장 사항에 대해 설명합니다. 이 정보는 모든 대상 AWS 아키텍처에 적용됩니다.

이 가이드는 다음과 같은 프로그램 또는 프로젝트 관리자, 제품 소유자, 솔루션 아키텍트 및 개발자를 위한 것입니다.

- 현대화 또는 비용 절감과 같은 다양한 이유로 SSIS에서 AWS 로 마이그레이션합니다.
- 주로 ETL 및 빅 데이터 워크로드에 AWS 서비스를 사용합니다.
- 서버리스 모델과 유연한 요금을 활용할 수 있는 클라우드 기반 솔루션을 찾습니다.

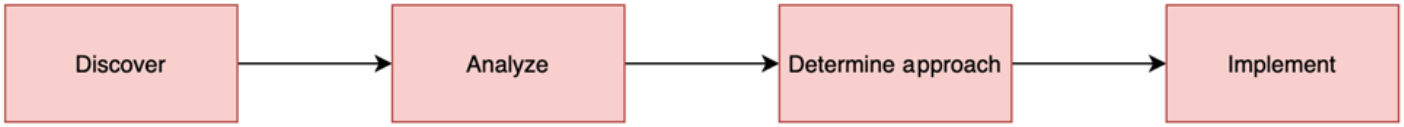
목표 비즈니스 성과

SSIS 작업을 AWS 클라우드로 마이그레이션하면 다음과 같은 주요 결과를 달성할 수 있습니다.

- 비용 효율적인 방식으로 기존 온프레미스 ETL 프로세스 현대화
- 조직의 정보 요구 사항에 더 빠르게 대응
- 유지 관리 및 운영 오버헤드를 줄이기 위해 기존 프로세스를 병합하고 미사용 프로세스를 사용 중지합니다.
- 조직의 향후 데이터 요구 사항을 충족하는 기반 구축

마이그레이션 단계

SSIS ETL 프로세스를 AWS 클라우드로 마이그레이션하는 작업은 검색, 분석, 접근 방식 결정 및 구현이라는 네 가지 주요 단계로 구성됩니다.



이러한 단계는 다음 섹션에서 설명합니다.

- 검색 단계
- 분석 단계
- 마이그레이션 접근 방식 결정
- 구현 단계

탐색 단계

검색 단계에서 마이그레이션하려는 SSIS 패키지 목록을 생성합니다 AWS. 개발 팀마다 ETL 작업 개발을 위한 다양한 스타일, 표준 및 패턴을 따릅니다. 조직의 기존 문서를 검토하여 이러한 패턴을 이해하는 것이 좋습니다. 그러나 설명서는 불완전한 경우가 많습니다. ETL 스크립트에서 중요한 정보의 추출을 자동화할 수 있습니다. 이를 통해 수작업과 시간을 절약하고, 인적 오류를 줄이고, 마이그레이션 접근 방식을 표준화할 수 있습니다. 다음은 추출하려는 몇 가지 중요한 세부 정보입니다.

- 총 제어 흐름 작업 수
- 제어 흐름 작업의 세부 정보
- 총 데이터 흐름 작업 수
- 사용된 데이터 흐름 변환
- 이벤트 핸들러
- 연결 관리자

이 정보를 사용하여 조직에서 사용되는 ETL 패턴을 이해하고, 복잡성을 평가하고,이 정보를 마이그레이션할 적절한 AWS 서비스를 식별할 수 있습니다.

SSIS에서 이러한 ETL 세부 정보를 마이그레이션하면 대부분의 마이그레이션 작업이 이루어집니다. 그러나 추가 속성은 설계 및 아키텍처 결정에 대한 인사이트를 제공할 수 있습니다. 이러한 SSIS 속성 중 일부는 다음과 같습니다.

- SSIS에서 장애 지점부터 작업을 다시 시작하는 데 사용되는 체크포인트
- 오류가 있더라도 특정 사용 사례에서 SSIS 패키지가 성공하는 데 도움이 되는 변수를 전파합니다.
- 데이터베이스에서 읽는 데이터의 품질을 제어하는 트랜잭션 격리 수준
- 로깅 - 현재 설계에서 캡처되는 로그 유형과 해당 스토리지 위치를 이해합니다.

다음 표와 같이 검색 단계의 결과는 인벤토리일 수 있습니다.

inventory

count	Package	Flow	Task
1	SSIS_Package_1	control_flow	Microsoft.ExecuteSQLTask
1	SSIS_Package_1	data_flow	Microsoft.BDD
1	SSIS_Package_1	data_flow	Microsoft.ConditionalSplit
2	SSIS_Package_1	data_flow	Microsoft.OLEDBDestination
1	SSIS_Package_1	data_flow	Microsoft.OLEDBSource
1	SSIS_Package_2	control_flow	Microsoft.DbMaintenanceFileCleanupTask
1	SSIS_Package_2	control_flow	Microsoft.ExecuteSQLTask
1	SSIS_Package_2	control_flow	Microsoft.ScriptTask
1	SSIS_Package_2	control_flow	STOCK:FOREACHLOOP
1	SSIS_Package_2	control_flow	STOCK:FORLOOP

이 인벤토리에는 다음 정보가 포함될 수 있습니다.

- 패키지: 마이그레이션할 SSIS 패키지의 이름
- 흐름: 흐름 또는 데이터 흐름 제어 https://docs.microsoft.com/en-us/sql/integration-services/data-flow/data-flow
- 작업: 제어 흐름 작업 또는 데이터 흐름 구성 요소의 이름
- 개수: SSIS 패키지에서 작업이 사용된 횟수

분석 단계

검색 단계의 정보를 분석하면 패턴을 이해하고 대상 아키텍처를 더 잘 설계하는 데 도움이 됩니다. 다음 포인터를 따라 분석 결과를 개선합니다.

- 패키지 수준의 로깅 옵션은 이벤트를 로깅하고 각 구성 요소에 대한 런타임 정보를 캡처합니다. 사용자 지정 로깅을 사용하면 실행 IDs 및 기타 런타임 값과 같은 추가 정보를 포함하도록 로그 파일을 구성할 수 있습니다.
- 분석, 수정 및 재처리를 위해 잘못된 레코드를 다른 스토리지 위치로 리디렉션합니다.
- 온프레미스 SSIS 환경에 구현된 데이터 아카이브 및 제거 전략을 이해합니다.
- SSIS에 포함된 작업(예: Send Mail 작업) 또는 사용자 지정 스크립트(예: 스크립트 작업)를 사용하여 알림 및 알림을 보냅니다.
- 손상된 데이터를 방지하기 위해 범위 내 변환의 동작을 이해합니다. 예를 들어, 조회 변환은 두 데이터 세트 간의 동등 조인이며 비교는 대/소문자를 구분합니다.

인벤토리의 각 항목을 기간(분 또는 시간) 또는 수준(단순, 중간 또는 복합) 측면에서 예상 복잡성에 매핑할 수 있습니다. 다음 표에 표시된이 확장된 인벤토리는 분석 단계의 결과입니다.

inventory

count	Package	Flow	Task	Effort	Complexity
1	SSIS_Package_1	control_flow	Microsoft.ExecuteSQLTask	30	S
1	SSIS_Package_1	data_flow	Microsoft.BDD	0	N/A
1	SSIS_Package_1	data_flow	Microsoft.ConditionalSplit	30	S
2	SSIS_Package_1	data_flow	Microsoft.OLEDBDestination	60	M
1	SSIS_Package_1	data_flow	Microsoft.OLEDBSource	30	S
1	SSIS_Package_2	control_flow	Microsoft.DbMaintenanceFileCleanupTask	60	M
1	SSIS_Package_2	control_flow	Microsoft.ExecuteSQLTask	30	S
1	SSIS_Package_2	control_flow	Microsoft.ScriptTask	120	C
1	SSIS_Package_2	control_flow	STOCK:FOREACHLOOP	30	S
1	SSIS_Package_2	control_flow	STOCK:FORLOOP	30	S

마이그레이션 접근 방식 결정

마이그레이션 접근 방식을 결정하려면 이전 단계의 기존 패턴에 대해 수행한 분석을 사용합니다. 조직의 향후 데이터 및 분석 요구 사항은 마찬가지로 중요한 고려 사항입니다. 기존 온프레미스 ETL 도구는 관계형 데이터 모델 및 구조화된 데이터를 처리합니다. 처리할 반정형 및 비정형 데이터가 있는 경우 마이그레이션에 AWS Glue 또는 Amazon EMR과 같은 서비스를 사용할 AWS 수 있습니다. 마이그레이션 접근 방식에 영향을 미칠 수 있는 다른 요인은 다음과 같습니다.

- 그래픽 인터페이스(예: AWS Glue Studio) 또는 사용자 지정 프레임워크(예: Spark/Python 라이브러리)를 사용할지 여부
- 온프레미스 소스 및 AWS 대상에 대한 보안 액세스 권한이 있는지 여부
- 팀에 필요한 기술 및 훈련
- 감사 및 규정 준수 요구 사항

빅 뱅, 단계별, 리프트 앤 시프트의 세 가지 마이그레이션 접근 방식 중에서 선택할 수 있습니다. 다음 표에서는 이러한 세 가지 접근 방식을 비교합니다.

접근 방식	설명	사용 사례	장단점
빅 뱅	특정 기간 내에 모든 SSIS 패키지를 마이그레이션합니다.	• 복잡성, 범위 및 대상 아키텍처는 명확합니다. • 팀에 필요한 기술이 있거나 학습 곡선이 얕습니다.	• 위험이 높습니다. • 단계적 접근 방식보다 시간이 적게 걸립니다. • AWS Glue Amazon EMR 또는 사용자 지정 프레임워크를 사용할 수 있습니다.
단계적	각 개별 패턴 및 복잡성에 대해 하나의 SSIS 패키지를 식별합니다. 패키지를 기존 아키텍처로 마이그레이션 AWS, 테스트 및 비교합니다.	• 시간은 제약 조건이 아닙니다. • 다양한 ETL 패턴에 대해 다양한 설계를 원합니다.	• 빅 뱅 접근 방식보다 덜 위험하지만 더 많은 시간과 노력이 필요합니다. • AWS Glue Amazon EMR 또는 사용자

접근 방식	설명	사용 사례	장단점
			지정 프레임워크를 사용할 수 있습니다.
리프트 앤 시프트	현재 아키텍처를 그대로 마이그레이션합니다 AWS.	• 온프레미스 하드웨어는 더 이상 지원되지 않습니다. • 마이그레이션을 즉시 계획할 리소스가 없습니다.	• 필요한 마이그레이션 작업 및 시간 최소화. • 기존 솔루션의 문제는 계속 켜져 있습니다 AWS. • SSIS 패키지는 있는 그대로 실행됩니다. 다른 ETL 도구나 프레임워크는 필요하지 않습니다.

성공적인 마이그레이션을 위해서는 소스 시스템과 대상 시스템의 데이터를 비교하는 것이 필수적입니다. 기존 프로덕션 시스템은 소스 시스템에서 정기적으로 업데이트를 받기 때문에이 비교가 혼동될 수 있습니다. 따라서 마이그레이션 접근 방식을 결정할 때 데이터 검증 전략도 결정하는 것이 좋습니다.

- 특정 날짜 및 시간에 소스 시스템의 프로덕션 환경에서 해당하는 모든 데이터베이스 및 파일을 백업합니다.
- 모든 작업이 백업된 소스 데이터에서 데이터를 성공적으로 로드한 후 대상 시스템의 프로덕션 환경에서 모든 데이터베이스를 백업합니다.
- 테스트 환경에서 소스 데이터를 복원하고 새 작업을 실행합니다.
- 소스 데이터베이스와 대상(이전 데이터베이스와 새 데이터베이스) 데이터베이스 간의 유효한 차이 백분율에 합의합니다. 예를 들어 1% 미만의 차이가 허용 가능하다고 결정할 수 있습니다.
- 다룰 모든 검증 규칙을 나열합니다.
- 최대한 비교를 자동화하고 모든 규칙을 다룹니다.

구현 단계

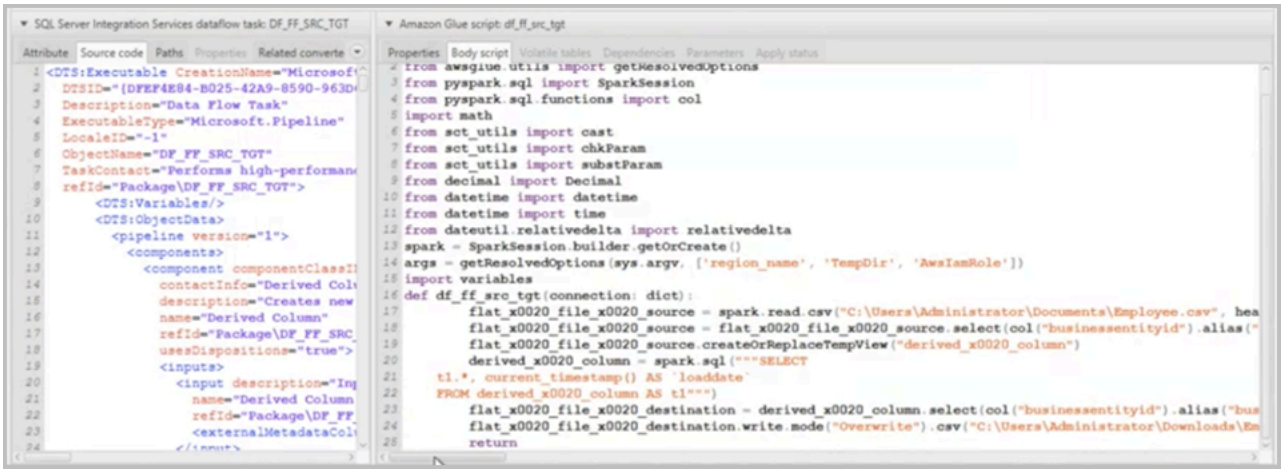
빅 뱅 또는 단계별 접근 방식을 따르는 마이그레이션에는 새로운 개발 및 테스트가 필요합니다. AWS Schema Conversion Tool (AWS SCT)는 SSIS 패키지에서 AWS Glue 작업을 자동으로 생성할 수 있습

니다. 이렇게 하면 마이그레이션 시간과 노력이 크게 줄어듭니다. 또는 그래픽 인터페이스 기반 개발에 AWS Glue Studio를 사용하거나 AWS Glue 또는 Amazon EMR에서 실행할 수 있는 Spark 라이브러리를 빌드할 수 있습니다.

다음 섹션에서는 AWS SCT AWS Glue및 Amazon EMR을 사용하는 데 유용한 포인터를 제공합니다.

AWS SCT

다음 화면 그림은에서 변환한 AWS Glue 작업 스크립트를 보여줍니다 AWS SCT.

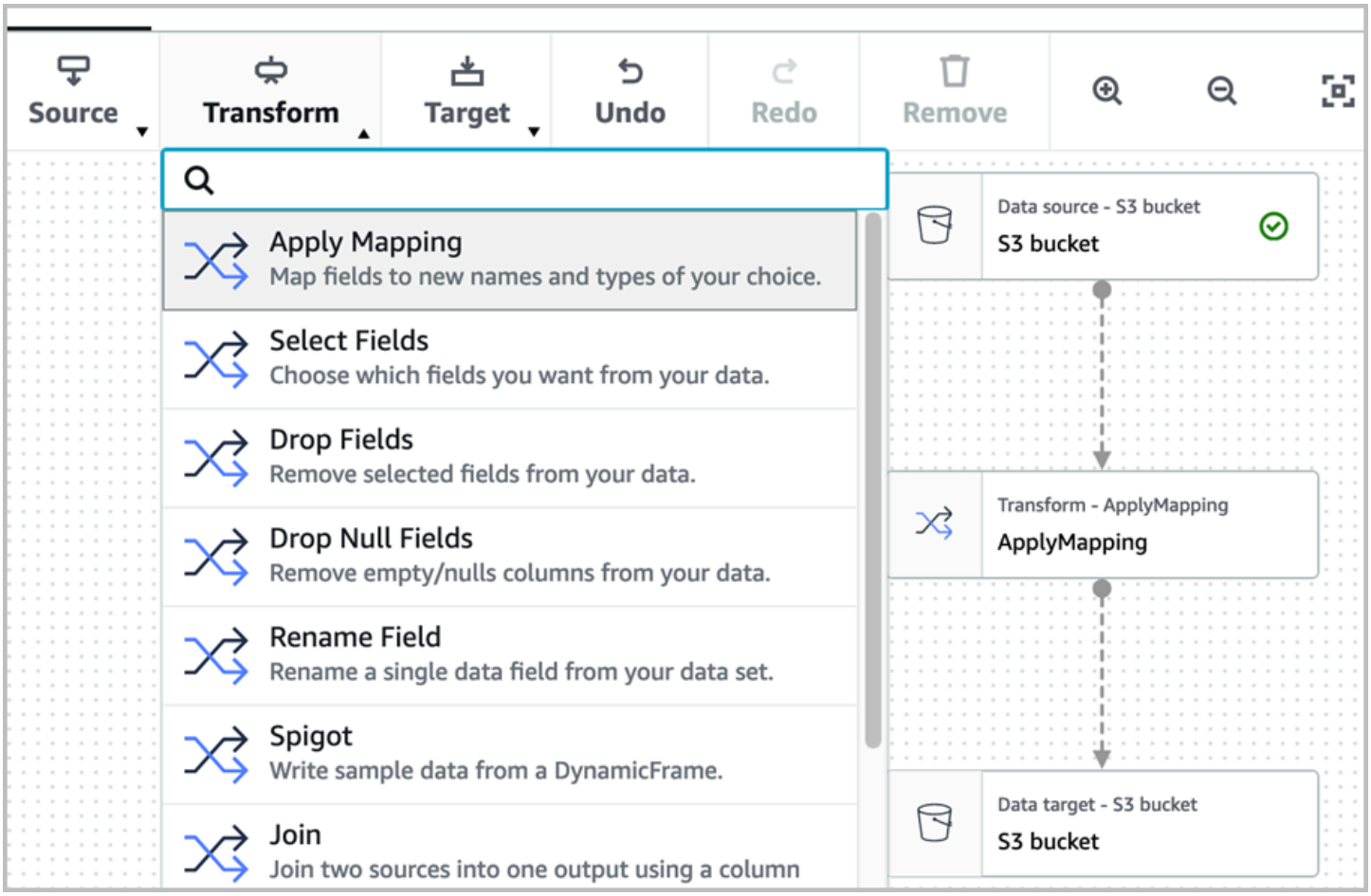


AWS SCT 는 SSIS 패키지를 AWS Glue 작업으로 대량으로 변환할 수 있습니다. 스크립트를 편집하여 기존 로직을 업데이트하거나 새 설계에 따라 새 로직을 추가할 수 있습니다. AWS SCT 변환된 스크립트의 이름 지정 규칙에 따라 스크립트를 사용자 지정하는 것이 좋습니다.

자세한 내용은 AWS SCT 설명서의를 AWS Glue 사용하여 SSIS를 로 변환 AWS SCT을 참조하세요.

AWS Glue

AWS Glue Studio는 다음 화면에 표시된 대로 그래픽 인터페이스와 SSIS와 유사한 개발 환경을 제공합니다.



그래픽 인터페이스를 사용하지 않으려면 AWS Glue 콘솔에서 필요한 Python 라이브러리를 사용하여 사용자 지정 스크립트를 실행할 수도 있습니다. 자세한 내용은 AWS Glue 설명서의 사용자 지정 스크립트 제공을 참조하세요.

AWS Glue 는 데이터 처리를 위한 기본 제공 변환 세트를 제공합니다. 이는 SSIS 데이터 흐름 변환과 유사합니다. 를 사용하여 SSIS ETL 작업을 마이그레이션할 때 AWS Glue다음 모범 사례를 따르세요.

- AWS Glue 변환에서 동등한 SSIS 변환으로 매핑을 준비합니다.
- 변환을 변환에 매핑할 수 없는 경우 Python 또는 Scala 사용자 지정 스크립트를 사용하여 AWS Glue 변환을 빌드합니다.
- 사용자 지정 로깅(예: 행 읽기, 행 쓰기 또는 잘못된 레코드)의 경우 Amazon CloudWatch 외에도 사용자 지정 스크립트를 사용합니다.
- 로컬에서 사용자 지정 스크립트를 개발하고 디버깅하려면 개발 엔드포인트를 추가합니다.

Amazon EMR

와 마찬가지로 EMR 클러스터에서 사용자 지정 스크립트(Python 또는 Scala로 작성됨) 또는 컴파일된 Python 라이브러리를 실행할 수 있습니다 AWS Glue. 다음 모범 사례를 따르세요.

- Spark 프레임워크를 사용하여 EMR 클러스터를 생성하는 동안 메모리 최적화 인스턴스 유형으로 시작합니다. (SSIS는 메모리 버퍼를 사용합니다.)
- 각 SSIS 작업 또는 변환과 동일한 일반 Python 메서드를 빌드합니다. 예를 들어 다음 그림에서 두 데이터 프레임을 입력으로 사용하는 메서드는 두 데이터 프레임의 일치하는 레코드를 출력으로 포함하는 세 번째 데이터 프레임을 생성합니다. 병합 조인 변환으로 작동합니다.

```
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def merge_join_spark(spark: SparkSession, df1: DataFrame, df2: DataFrame, join_type: str, join_column:str,
                     merge_fetch_columns: str):
    """Merge/Join directly
        spark: Current Spark session
        df1: First dataframe for merge join
        df2: Second dataframe for merge join
        join_type: Left/Inner/Outer join
        join_column : Column to be merge joined on
        merge_fetch_columns: Columns required in the output dataframe
    """
    try:

        spark.catalog.dropTempView("df1")
        spark.catalog.dropTempView("df2")
        df1.createOrReplaceTempView("df1")
        df2.createOrReplaceTempView("df2")
        merge_sql = f"select src.*, {merge_fetch_columns} from df1 {join_type} join df2 on df1.{join_column} = df2.{join_column}".format(
            merge_fetch_columns=merge_fetch_columns, join_type=join_type,join_column=join_column)
        logger.debug("Merge query is : " + merge_sql)
        output_df = spark.sql(merge_sql)
    except Exception as ex:
        logger.error("Merge Execution Failed for " + str(ex))
        raise Exception("Merge Execution Failed " + str(ex))
    finally:
        spark.catalog.dropTempView("df1")
        spark.catalog.dropTempView("df2")
    return output_df
```

테스트

데이터의 완전성과 정확성을 검증하려면 테스트 프레임워크가 필요합니다. 이 프레임워크는 작업을 마이그레이션하는 동안 수행한 모든 기존 시나리오와 모든 개선 사항을 포함해야 합니다 AWS.

- 완전성 검증:
  - 모든 작업이 대상 상태로 마이그레이션됩니다.
  - 각 작업에서 모든 기능이 마이그레이션됩니다.
  - 작업 실행 세부 정보, 오류 메시지, 잘못된 레코드, 행 수를 포함한 모든 유형의 로그를 사용할 수 있습니다.

- 정확성 검증:
  - 데이터의 품질은 기존 환경과 새 환경에서 일관됩니다.
  - 모든 테이블의 모든 열이 일치하거나 테이블이 개선됩니다 AWS.
  - 모든 감사 및 로깅 정보가 일치합니다.

또한 마이그레이션된 작업의 성능이 기존 작업의 성능과 일치하는지 확인해야 합니다.

모범 사례

SSIS 작업을 다음으로 마이그레이션할 때 AWS다음 일반 모범 사례를 따르세요.

- 데이터베이스를 Amazon Relational Database Service(RDS)와 같은 AWS 관리형 서비스로 마이그레이션하는 경우에서 적용하거나 관리하지 않는 유지 관리 작업(예: 백업, 복원, 통계 업데이트 또는 데이터베이스 무결성 확인)의 범위를 좁힙니다 AWS.
- 재사용 가능한 스크립트와 메서드를 구축하여 개발을 표준화하고 노력을 줄입니다.
- 항상 프로덕션 리소스와 일치하는 인프라 및 데이터 볼륨에서 마이그레이션된 작업을 테스트합니다.
- SSIS 패키지는 XML 형식입니다. XML 구문 분석기 유틸리티를 구축하여 가능한 경우 마이그레이션 활동을 자동화합니다. 다음 예제에서는 XML 형식의 SSIS 패키지를 보여줍니다.

```
    DTS:LastModifiedProductVersion="11.0.2100.60"
    DTS:LocaleID="1033"
    DTS:ObjectName="Package"
    DTS:PackageType="5"
    DTS:VersionBuild="1"
    DTS:VersionGUID="{EC16490E-6783-4BE6-92CA-FDD5BC644F88}">
    <DTS:Property
      DTS:Name="PackageFormatVersion">6</DTS:Property>
    <DTS:ConnectionManagers>
      <DTS:ConnectionManager
```

다음 그림은 샘플 XML 구문 분석기를 보여줍니다.

```
def getlistofallexecutables(directory):
    """
        Iterate the root directory with all SSIS packages (.dtsx)
        Get the control flow tasks and data flow task components
    """
    try:
        lst=[]
        for fileName in os.listdir(directory):
                if fileName.endswith('.dtsx'):
                        cfgfilename=os.path.basename(fileName).split('.')[0]
                        tree = etree.parse(directory+fileName)
                        root = tree.getroot()
                        prefix = '{www.microsoft.com/'
                        exetag = prefix + 'SqlServer/Dts}Executable'
                        dataflow='Microsoft.Pipeline'
                        childtag=prefix+ 'SqlServer/Dts}ExecutableType'
                        objdata=prefix+ 'SqlServer/Dts}ObjectData'
                        pipeline='pipeline'
                        components= 'components'
                        componentclassid="componentClassID"
                        for cnt, ele in enumerate(root.xpath(".//*")):
                                if ele.tag==exetag:
                                        attr=ele.attrib[childtag]#controlflow
                                        flow=cfgfilename+',control_flow,'
                                        if attr!=dataflow:
                                                lst.append(flow+attr)
                                        if attr==dataflow:
                                                for child0 in ele:
                                                        if child0.tag==objdata:
                                                                for child1 in child0:
                                                                        if child1.tag==pipeline:
                                                                                for child2 in child1:
                                                                                        if child2.tag==components:
                                                                                                for child3 in child2:
                                                                                                        df_component=child3.attrib[componentclassid]
                                                                                                        flow=cfgfilename+',data_flow,'
                                                                                                        lst.append(flow+df_component)
        return lst
    except Exception as e:
```

- 체크섬과 해시 값을 사용하여 정확성과 완전성을 테스트합니다.
- 사용자 지정 라이브러리를 빌드하는 경우 스레딩을 구현하여 작업을 병렬로 실행합니다. 이렇게 하면 작업 성능이 향상됩니다.
- 모든 작업이 일정 기간 AWS 동안 안정된 후에만 기존 환경을 폐기합니다.
- 로깅에 Amazon CloudWatch를 사용하여 로깅 사용자 지정 작업을 줄입니다.
- Amazon Simple Notification Service(Amazon SNS)를 사용하여 알림 전송을 위한 사용자 지정 작업을 대체합니다. Amazon Simple Email Service(Amazon SES)를 사용하여 사용자 지정 이메일 작업을 대체합니다.

FAQ

이 섹션에서는 SSIS ETL 작업을 로 마이그레이션하는 것과 관련하여 자주 묻는 질문에 대한 답변을 제공합니다 AWS.

마이그레이션 중에 SSIS 작업을 개선해야 합니까?

예. 기존 작업과 새 작업 모두에서 동시에 중요한 개선 사항 또는 버그 수정을 구현합니다. 마이그레이션 후 우선 순위가 낮은 변경 사항을 구현할 수 있습니다.

에서 작업을 모니터링하려면 어떻게 해야 합니까 AWS?

Amazon CloudWatch를 사용하여 작업을 모니터링하고 규칙 기반 알림을 보낼 수 있습니다.

온프레미스 SSIS 작업은 언제 폐기할 수 있나요?

두 환경 모두에서 데이터의 성능, 정확성 및 완전성이 동일할 때까지 이전 작업과 새 작업을 동시에 유지하는 것이 좋습니다. 또한 보고 시스템을 복제하여 두 환경의 보고서를 연결하고 비교할 수 있습니다. 보고서가 동일한 경우 SSIS 작업을 폐기할 수 있습니다.

다음 단계

다음 단계에서는 개념 증명을 구축하여 대상 아키텍처에 대한 배포 접근 방식을 결정하는 것이 좋습니다. 개념 증명의 경우:

- AWS Glue Studio 그래픽 인터페이스를 사용하여 AWS Glue 작업을 생성합니다.
- AWS Schema Conversion Tool (AWS SCT)를 사용하여 SSIS 패키지에서 AWS Glue 스크립트를 생성합니다.
- Spark 라이브러리를 사용하여 사용자 지정 마이그레이션 프레임워크를 구축합니다.
- 테스트 프레임워크를 구축합니다.
- 지속적 통합 및 지속적 배포(CI/CD)를 위한 도구 및 프로세스를 식별합니다.

관련 리소스

- 를 AWS Glue 사용하여 SSIS를 로 변환 AWS SCT
- Amazon RDS 백업 및 복원
- AWS Glue 설명서
- AWS Glue FAQs
- AWS Glue 작업에 대한 연속 로깅
- Amazon EMR 설명서
- Amazon EMR FAQs
- CloudWatch를 사용하여 지표 모니터링
- PyDeequ를 사용하여 대규모 데이터 마이그레이션 검증 가속화
- AWS 데이터 및 분석 역량 파트너

문서 기록

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
최초 게시	—	2021년 12월 6일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.