SSIS ETL ジョブの への移行 AWS

AWS 規範ガイダンス

AWS 規範ガイダンス: SSIS ETL ジョブの への移行 AWS

Table of Contents

SSIS ETL ジョブの への移行 AWS

Durga Prasad と Harpreet Singh、Amazon Web Services (AWS)

2021 年 12 月 (ドキュメント履歴)

Amazon Web Services (AWS) は、抽出、変換、ロード (ETL) または抽出、ロード、変換 (ELT) ジョブとビッグデータワークロードを開発および実行するためのサービスを提供します。Microsoft SQL Server Integration Services (SSIS) は、ETL ジョブを構築するためのグラフィカルインターフェイスを備えたオンプレミス ETL ツールです。ターゲット状態アーキテクチャに応じて、 AWS Glue グラフィカルインターフェイスまたは Apache Spark フレームワークと Python ライブラリを使用して、AWS クラウドで ETL ジョブを構築できます。

このガイドでは、SSIS ETL/ELT ジョブをオンプレミスから に移行する手順について説明します AWS。移行フェーズ、ベストプラクティス、推奨事項について説明し、移行の労力を減らし、エクスペリエンスを向上させます。この情報は、すべてのターゲット AWS アーキテクチャに適用されます。

このガイドは、プログラムまたはプロジェクトマネージャー、製品所有者、ソリューションアーキテクト、および以下の開発者を対象としています。

- モダナイゼーションやコスト削減など、 AWS さまざまな理由で SSIS から に移行する。
- 主に ETL およびビッグデータワークロード用の AWS サービスを使用します。
- サーバーレスモデルと柔軟な料金を活用するためのクラウドベースのソリューションを探しています。

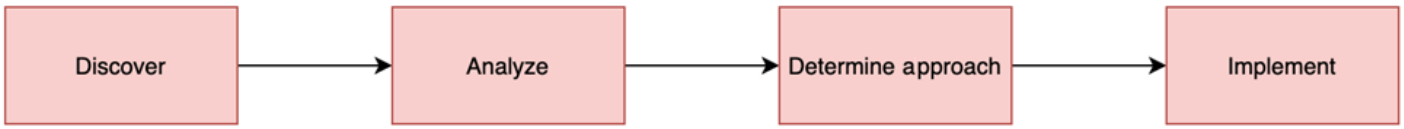
ターゲットを絞ったビジネス成果

SSIS ジョブを AWS クラウドに移行すると、次の主な成果を達成できます。

- 既存のオンプレミス ETL プロセスを費用対効果の高い方法でモダナイズする
- 組織の情報ニーズに迅速に対応
- 既存のプロセスをマージし、未使用のプロセスを廃止して、メンテナンスと運用のオーバーヘッドを削減する
- 組織の将来のデータ要件を満たすための基盤を構築する

移行フェーズ

SSIS ETL プロセスを AWS クラウドに移行するには、検出、分析、アプローチの決定、実装の 4 つの主要なフェーズがあります。



これらのフェーズについては、以下のセクションで説明します。

- 検出フェーズ
- 分析フェーズ
- 移行アプローチの決定
- 実装フェーズ

発見フェーズ

検出フェーズでは、移行する SSIS パッケージのリストを作成します AWS。さまざまな開発チームは、ETL ジョブを開発するためのさまざまなスタイル、標準、パターンに従います。これらのパターンを理解するには、組織の既存のドキュメントを確認することをお勧めします。ただし、多くの場合、ドキュメントは不完全です。ETL スクリプトからの重要な情報の抽出を自動化できます。これにより、手動作業と時間を節約し、人為的ミスを減らし、移行アプローチを標準化できます。以下に、抽出する重要な詳細をいくつか示します。

- コントロールフロータスクの合計数
- コントロールフロータスクの詳細
- データフロータスクの合計数
- 使用されるデータフロー変換
- イベントハンドラー
- 接続マネージャー

この情報を使用して、組織で使用される ETL パターンを理解し、その複雑さを評価し、この情報を移行する適切な AWS サービスを特定します。

SSIS からこれらの ETL の詳細を移行することは、移行作業の大部分を形成します。ただし、追加のプロパティは、設計とアーキテクチャの決定に関するインサイトを提供することができます。これらの SSIS プロパティの一部は次のとおりです。

- SSIS で障害発生時点からジョブを再起動するために使用されるチェックポイント
- エラーが発生しても SSIS パッケージが特定のユースケースで成功するのに役立つ変数を伝播する
- データベースから読み取られるデータの品質を制御するトランザクション分離レベル
- ログ記録: 現在の設計でキャプチャされているログのタイプとそのストレージの場所を把握する

次の表に示すように、検出フェーズの結果はインベントリになります。

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

このインベントリには、次の情報が含まれる場合があります。

- パッケージ: 移行する SSIS パッケージの名前
- フロー: フローまたはデータフローの制御
- タスク: コントロールフロータスクまたはデータフローコンポーネントの名前
- カウント: タスクが SSIS パッケージで使用された回数

分析フェーズ

検出フェーズの情報を分析すると、パターンを理解し、ターゲットアーキテクチャをより適切に設計できます。分析の結果を向上させるには、以下のポインタに従ってください。

- パッケージレベルのログ記録オプションは、イベントをログに記録し、各コンポーネントのランタイム情報をキャプチャします。カスタムログ記録を使用すると、実行 IDs やその他のランタイム値などの追加情報を含めるようにログファイルを設定できます。
- 不正なレコードを別のストレージロケーションにリダイレクトして、分析、修正、再処理を行います。
- オンプレミス SSIS 環境で実装されているデータアーカイブとパージ戦略を理解します。
- SSIS に含まれるタスク (メール送信タスクなど) またはカスタムスクリプト (スクリプトタスクなど) を使用してアラートと通知を送信します。
- データが破損しないように、範囲内の変換の動作を理解します。たとえば、ルックアップ変換は 2 つのデータセット間で等価結合であり、比較では大文字と小文字が区別されます。

インベントリの各エントリは、期間 (分または時間) またはレベル (シンプル、中程度、または複雑) の推定複雑さにマッピングできます。次の表に示すこの拡張インベントリは、分析フェーズの結果です。

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

移行アプローチの決定

移行アプローチを決定するには、前のフェーズで既存のパターンに対して実行した分析を使用します。組織の将来のデータと分析のニーズも同様に重要な考慮事項です。従来のオンプレミス ETL ツールは、リレーショナルデータモデルと構造化データを扱います。半構造化データと非構造化データを処理する場合は、移行に AWS Glue や Amazon EMR などのサービスを使用できます AWS 。移行アプローチに影響を与える可能性のあるその他の要因は次のとおりです。

- グラフィカルインターフェイス (AWS Glue Studio など) を使用するか、カスタムフレームワーク (Spark/Python ライブラリなど) を使用するか
- オンプレミスのソースと AWS ターゲットに安全にアクセスできるかどうか
- チームに必要なスキルとトレーニング
- 監査とコンプライアンスの要件

移行アプローチは、ビッグバン、フェーズド、リフトアンドシフトの 3 つから選択できます。次の表は、これら 3 つのアプローチを比較したものです。

アプローチ	説明	ユースケース	利点と欠点
ビッグバング	特定の期間内にすべての SSIS パッケージを移行します。	• 複雑さ、範囲、ターゲットアーキテクチャは明確です。 • チームに必要なスキルがあるか、学習曲線が浅い。	• 高リスク。 • 段階的アプローチよりも時間がかかります。 • AWS Glue、Amazon EMR、またはカスタムフレームワークを使用できます。
段階的	個別のパターンと複雑さごとに 1 つの SSIS パッケージを特定します。パッケージを既存のアーキテクチャに移行し、	• 時間は制約ではありません。 • ETL パターンごとに異なる設計が必要です。	• ビッグバングアプローチよりもリスクは低くなりますが、時間と労力がかかります。 • AWS Glue、Amazon EMR、または

アプローチ	説明	ユースケース	利点と欠点
	AWSテストして、結果を比較します。		カスタムフレームワークを使用できます。
リフトアンドシフト	現在のアーキテクチャをそのまま移行します AWS。	• オンプレミスハードウェアはサポートされなくなりました。 • 移行をすぐに計画するためのリソースがない。	• 移行にかかる時間と労力が最も少ない。 • 既存のソリューションの問題は引き続き発生します AWS。 • SSIS パッケージはそのまま実行されます。他の ETL ツールやフレームワークは必要ありません。

移行を成功させるには、ソースシステムとターゲットシステム上のデータの比較が不可欠です。既存の本稼働システムはソースシステムから定期的に更新されるため、この比較はわかりにくい場合があります。このため、移行アプローチを決定するときは、データ検証戦略も決定することをお勧めします。

- ソースシステムの本番環境から、該当するすべてのデータベースとファイルのバックアップを特定の日時で取得します。
- バックアップされたソースデータからすべてのジョブが正常にデータをロードした後、ターゲットシステムの本番稼働環境からすべてのデータベースのバックアップを作成します。
- テスト環境でソースデータを復元し、新しいジョブを実行します。
- ソースデータベースとターゲットデータベース (古いデータベースと新しいデータベース) の有効な違いの割合について合意します。例えば、1% 未満の差は許容できると判断できます。
- 対象となるすべての検証ルールを一覧表示します。
- 比較を可能な限り自動化し、すべてのルールをカバーします。

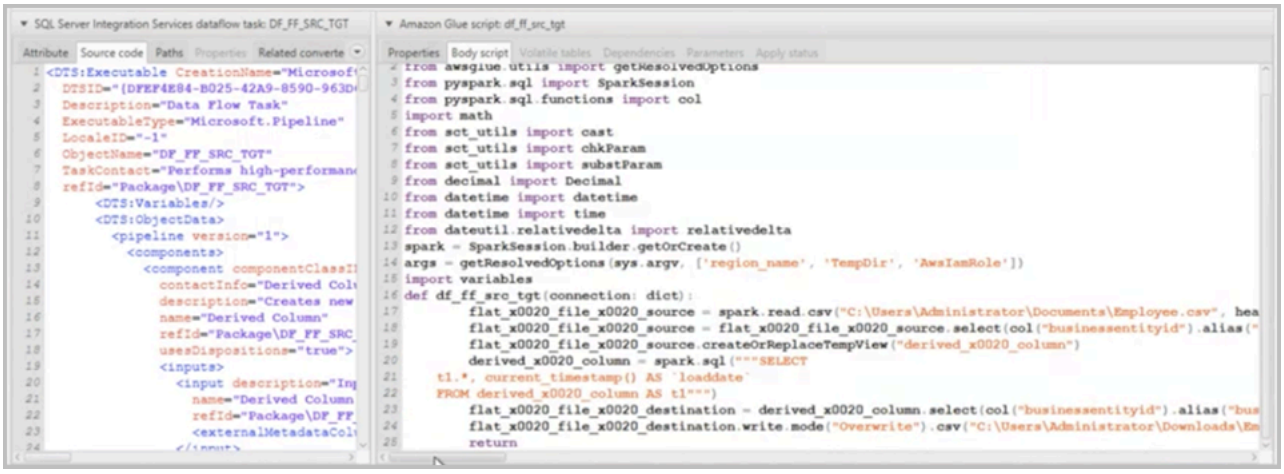
実装フェーズ

大規模なアプローチや段階的なアプローチに従う移行には、新しい開発とテストが必要です。 AWS Schema Conversion Tool (AWS SCT) は SSIS パッケージから AWS Glue ジョブを自動的に生成できます。これにより、移行時間と労力が大幅に削減されます。または、 AWS Glue Studio を使用してグラフィカルインターフェイスベースの開発を行ったり、 AWS Glue または Amazon EMR で実行できる Spark ライブラリを構築したりできます。

以下のセクションでは、 AWS SCT、 AWS Glue、および Amazon EMR を使用するための便利なポインタを提供します。

AWS SCT

次の画面図は、 によって変換された AWS Glue ジョブスクリプトを示しています AWS SCT。

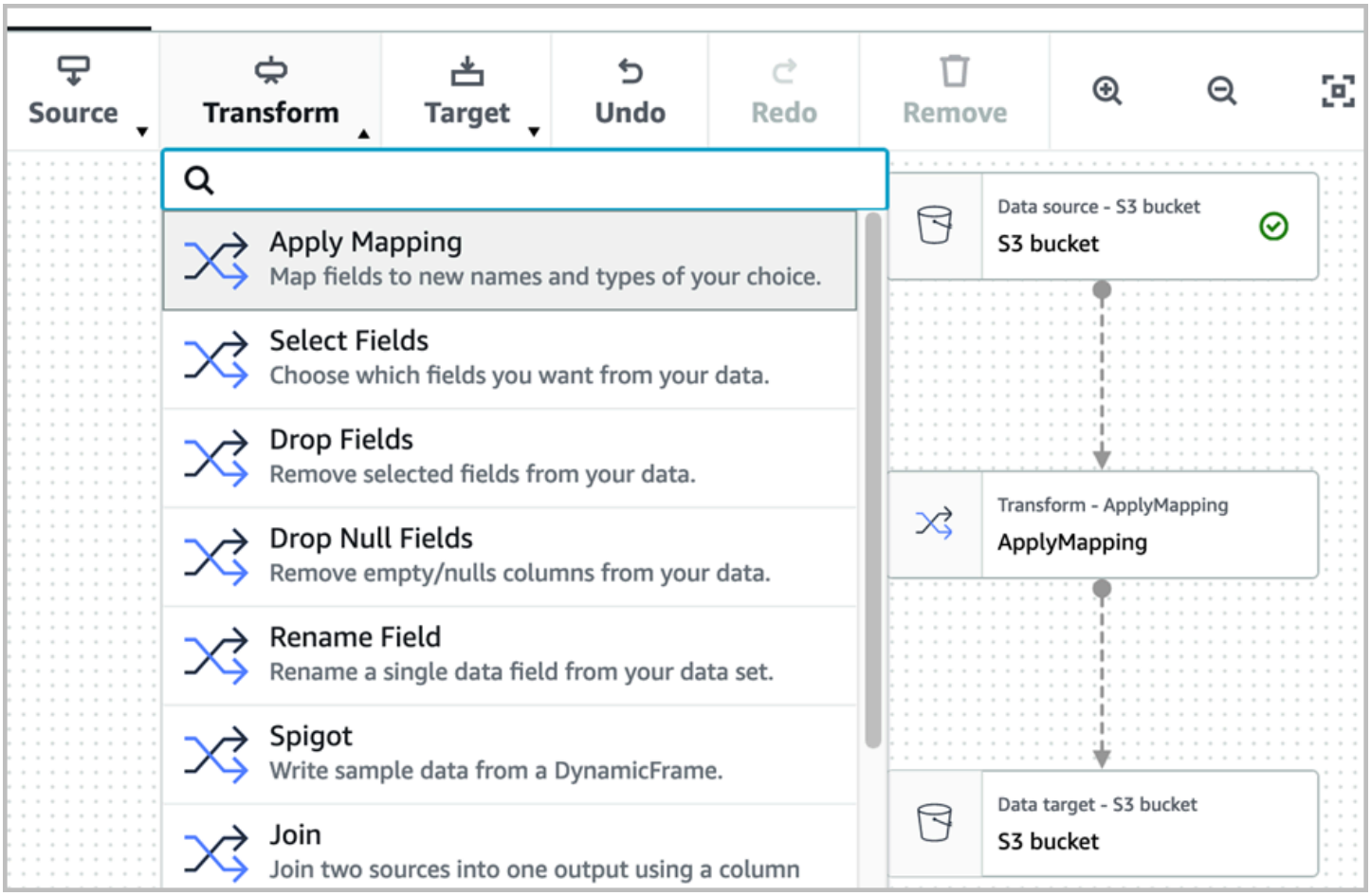


AWS SCT は SSIS パッケージを AWS Glue ジョブに一括変換できます。スクリプトを編集して既存のロジックを更新したり、新しい設計に基づいて新しいロジックを追加したりできます。 AWS SCT 変換されたスクリプトの命名規則に従って、スクリプトをカスタマイズすることをお勧めします。

詳細については、 AWS SCT ドキュメントの「 AWS Glue を使用して SSIS を に変換 AWS SCTする」を参照してください。

AWS Glue

AWS Glue Studio は、次の画面に示すように、グラフィカルインターフェイスと SSIS に似た開発エクスペリエンスを提供します。



グラフィカルインターフェイスを使用しない場合は、 AWS Glue コンソールから必要な Python ライブラリを使用してカスタムスクリプトを実行することもできます。詳細については、 AWS Glue ドキュメントの「独自のカスタムスクリプトの提供」を参照してください。

AWS Glue には、データを処理するための一連の組み込み変換が用意されています。これらは SSIS データフロー変換に似ています。SSIS ETL ジョブを移行するときは、次のベストプラクティスに従ってください AWS Glue。

- AWS Glue 変換から同等の SSIS 変換へのマッピングを準備します。
- 変換を変換にマッピングできない場合は、Python または Scala カスタムスクリプトを使用して AWS Glue 変換を構築します。
- カスタムログ記録 (読み取り行、書き込み行、不正なレコードなど) には、Amazon CloudWatch に加えてカスタムスクリプトを使用します。
- 開発エンドポイントを追加して、カスタムスクリプトをローカルで開発およびデバッグします。

Amazon EMR

カスタムスクリプト (Python または Scala で記述) またはコンパイルされた Python ライブラリを EMR クラスターで実行できます AWS Glue。次のベストプラクティスに従ってください。

- Spark フレームワークで EMR クラスターを作成するときに、メモリ最適化インスタンスタイプから始めます。(SSIS はメモリバッファを使用します)。
- 各 SSIS タスクまたは変換と同等の汎用 Python メソッドを構築します。たとえば、次の図では、2 つのデータフレームを入力として受け取るメソッドは、2 つのデータフレームから一致するレコードを出力として持つ 3 番目のデータフレームを生成します。これはマージ結合変換として機能します。

```
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
```

テスト

データの完全性と正確性を検証するには、テストフレームワークが必要です。このフレームワークは、既存のシナリオと、ジョブの移行中に行った改善をすべて網羅している必要があります AWS。

- 完全性の検証:
  - すべてのジョブはターゲット状態に移行されます。
  - すべての機能は各ジョブで移行されます。

  - ジョブ実行の詳細、エラーメッセージ、不良レコード、行数など、すべてのタイプのログを使用できます。
- 正確性の検証:
  - データの品質は、既存の環境と新しい環境で一貫しています。
  - すべてのテーブルのすべての列が一致しているか、テーブルが改善されています AWS。
  - すべての監査情報と口グ情報が一致します。

また、移行されたジョブのパフォーマンスが既存のジョブのパフォーマンスと一致していることを確認する必要があります。

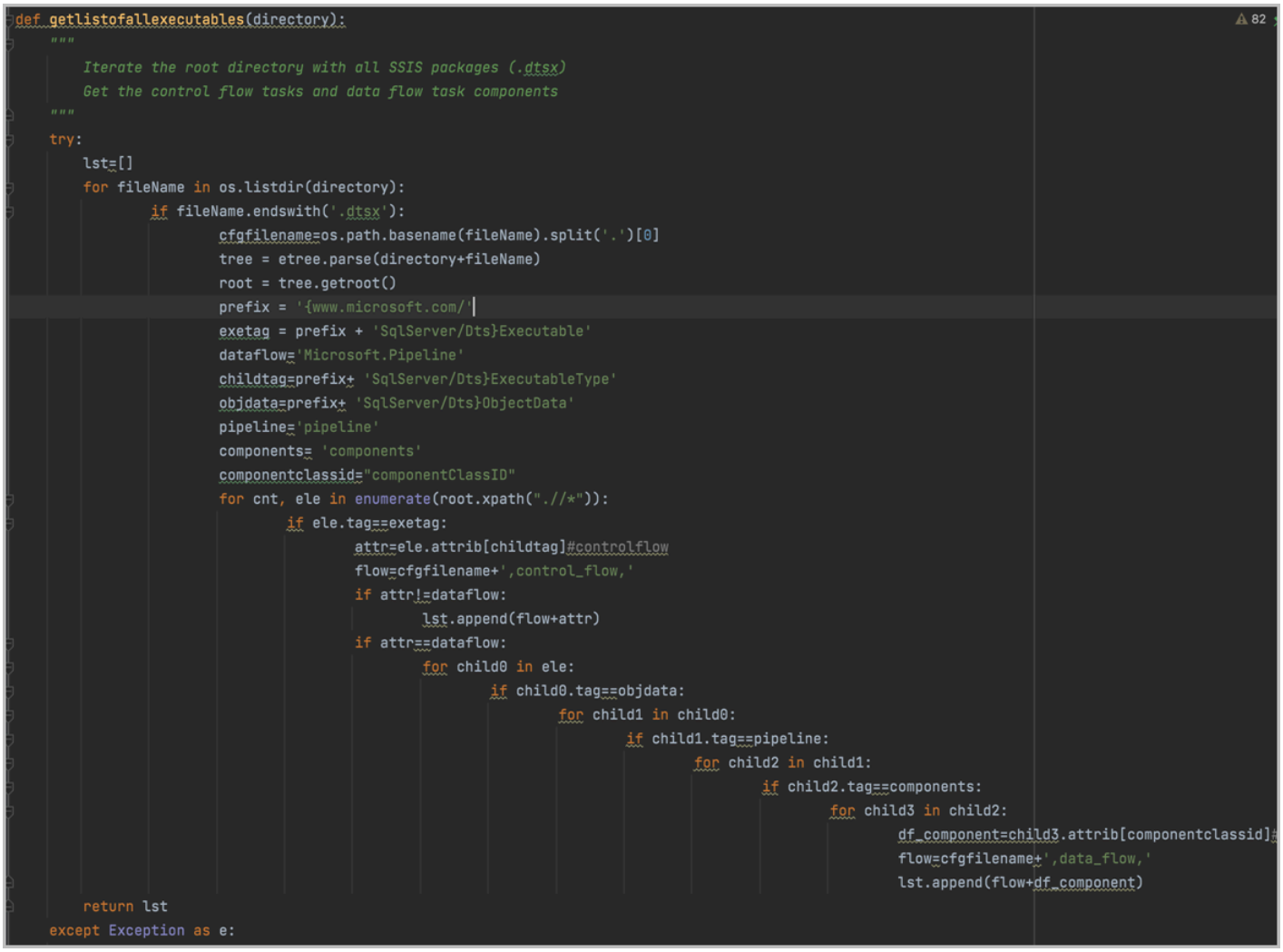
ベストプラクティス

SSIS ジョブを に移行するときは、以下の一般的なベストプラクティスに従ってください AWS。

- データベースを Amazon Relational Database Service (Amazon RDS) などの AWS マネージドサービスに移行する場合は、適用または管理されていないメンテナンスタスク (バックアップ、復元、統計の更新、データベースの整合性の確認など) を破棄します AWS。
- 再利用可能なスクリプトとメソッドを構築して、開発を標準化し、労力を削減します。
- 移行されたジョブは、本番稼働用リソースと一致するインフラストラクチャとデータボリュームで常にテストしてください。
- SSIS パッケージは XML 形式です。XML パーサーユーティリティを構築して、可能な場合は移行アクティビティを自動化します。次の例は、XML 形式の SSIS パッケージを示しています。

```
   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
```

次の図は、サンプル XML パーサーを示しています。



```
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:
```

- チェックサムとハッシュ値を使用して、正確性と完全性をテストします。
- カスタムライブラリを構築する場合は、スレッドを実装してタスクを並行して実行します。これにより、ジョブのパフォーマンスが向上します。
- すべてのジョブが一定期間安定している場合にのみ、既存の環境 AWS を廃止します。
- Amazon CloudWatch を ログ記録に使用して、ログ記録のカスタマイズ作業を減らします。
- Amazon Simple Notification Service (Amazon SNS) を使用して、通知を送信するためのカスタムタスクを置き換えます。Amazon Simple Email Service (Amazon SES) を使用して、カスタム E メールタスクを置き換えます。

よくある質問

このセクションでは、SSIS ETL ジョブの への移行に関してよく寄せられる質問に対する回答を提供します AWS。

移行中に SSIS ジョブを強化すべきですか？

はい。既存のジョブと新しいジョブの両方で、重要な機能強化やバグ修正を同時に実装します。優先度の低い変更は、移行後に実装できます。

でジョブをモニタリングするにはどうすればよいですか AWS？

Amazon CloudWatch を使用してジョブをモニタリングし、ルールベースのアラートを送信できます。

オンプレミス SSIS ジョブはいつ廃止できますか？

データのパフォーマンス、正確性、完全性が両方の環境で同じになるまで、古いジョブと新しいジョブを並行して保持することをお勧めします。レポートシステムをレプリケートして、両方の環境のレポートを接続して比較することもできます。レポートが同じであれば、SSIS ジョブを廃止できます。

次のステップ

次のステップとして、ターゲットアーキテクチャのデプロイアプローチを決定するための概念実証を構築することをお勧めします。概念実証の場合：

- AWS Glue Studio グラフィカルインターフェイスを使用して AWS Glue ジョブを作成します。
- AWS Schema Conversion Tool （ AWS SCT) を使用して SSIS パッケージから AWS Glue スクリプトを生成します。
- Spark ライブラリを使用してカスタム移行フレームワークを構築します。
- テストフレームワークを構築します。
- 継続的インテグレーションと継続的デプロイ (CI/CD) のためのツールとプロセスを特定します。

関連リソース

- AWS Glue を使用して SSIS を に変換する AWS SCT
- Amazon RDS のバックアップと復元
- AWS Glue ドキュメント
- AWS Glue よくある質問
- AWS Glue ジョブの継続的な口グ記録
- Amazon EMR ドキュメント
- Amazon EMR FAQs
- CloudWatch でメトリクスをモニタリングする
- PyDeequ を使用して大規模なデータ移行検証を高速化する
- AWS データおよび分析コンピテンシーパートナー

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィード をサブスクライブできます。

変更	説明	日付
初版発行	—	2021 年 12 月 6 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。