Migrasi pekerjaan SSIS ETL ke AWS

AWS Panduan Preskriptif

AWS Panduan Preskriptif: Migrasi pekerjaan SSIS ETL ke AWS

Table of Contents

Migrasi pekerjaan SSIS ETL ke AWS

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

Desember 2021 (riwayat dokumen)

Amazon Web Services (AWS) menyediakan layanan untuk mengembangkan dan menjalankan tugas ekstrak, transformasi, dan pemuatan (ETL) atau mengekstrak, memuat, dan mengubah (ELT) pekerjaan dan beban kerja big data. Microsoft SQL Server Integration Services (SSIS) adalah alat ETL lokal yang memiliki antarmuka grafis untuk membangun pekerjaan ETL. Bergantung pada arsitektur status target, Anda dapat menggunakan antarmuka AWS Glue grafis atau kerangka Apache Spark dengan pustaka Python untuk membangun pekerjaan ETL di Cloud. AWS

Panduan ini menjelaskan langkah-langkah untuk memigrasi pekerjaan SSIS ETL/ELT dari satu tempat ke lokasi. AWS Ini membahas fase migrasi, praktik terbaik, dan rekomendasi untuk mengurangi upaya migrasi dan meningkatkan pengalaman. Informasi ini berlaku untuk AWS arsitektur target apa pun.

Panduan ini untuk manajer program atau proyek, pemilik produk, arsitek solusi, dan pengembang yang:

- Bermigrasi dari SSIS ke karena berbagai AWS alasan, seperti modernisasi atau penghematan biaya.
- Terutama menggunakan AWS layanan untuk beban kerja ETL dan big data.
- Mencari solusi berbasis cloud untuk memanfaatkan model tanpa server dan harga yang fleksibel.

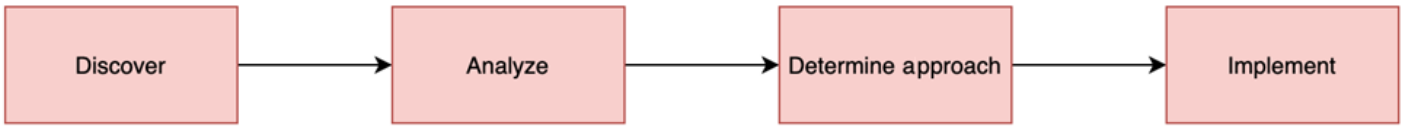
Hasil bisnis yang ditargetkan

Memigrasi pekerjaan SSIS Anda ke AWS Cloud membantu Anda mencapai hasil utama berikut:

- Memodernisasi proses ETL lokal yang ada dengan cara yang hemat biaya
- Menanggapi kebutuhan informasi organisasi lebih cepat
- Gabungkan proses yang ada dan hentikan proses yang tidak digunakan untuk mengurangi pemeliharaan dan overhead operasional
- Bangun fondasi untuk memenuhi persyaratan data masa depan organisasi Anda

Fase migrasi

Migrasi proses ETL SSIS ke AWS Cloud terdiri dari empat fase utama: temukan, analisis, tentukan pendekatan, dan implementasikan.



Fase-fase ini dibahas dalam bagian berikut:

- Fase penemuan
- Fase analisis
- Menentukan pendekatan migrasi
- Fase implementasi

Fase penemuan

Pada fase penemuan, Anda membuat daftar paket SSIS yang ingin Anda migrasikan. AWS Tim pengembangan yang berbeda mengikuti gaya, standar, dan pola yang berbeda untuk mengembangkan pekerjaan ETL. Kami menyarankan Anda meninjau dokumen organisasi yang ada untuk memahami pola-pola ini. Namun, dokumentasinya seringkali tidak lengkap. Anda dapat mengotomatiskan ekstraksi informasi penting dari skrip ETL. Ini menghemat upaya dan waktu manual, mengurangi kesalahan manusia, dan menstandarisasi pendekatan migrasi. Berikut adalah beberapa detail penting yang ingin Anda ekstrak:

- Jumlah total tugas aliran kontrol
- Detail tugas aliran kontrol
- Jumlah total tugas aliran data
- Transformasi aliran data yang digunakan
- Penangan acara
- Manajer koneksi

Gunakan informasi ini untuk memahami pola ETL yang digunakan di organisasi Anda, untuk mengevaluasi kompleksitasnya, dan untuk mengidentifikasi AWS layanan yang sesuai untuk memigrasikan informasi ini.

Migrasi detail ETL ini dari SSIS membentuk sebagian besar upaya migrasi. Namun, properti tambahan dapat memberikan wawasan tentang keputusan desain dan arsitektur. Beberapa properti SSIS ini adalah:

- Pos pemeriksaan, yang digunakan dalam SSIS untuk memulai kembali pekerjaan dari titik kegagalan
- Menyebarkan variabel, yang membantu paket SSIS berhasil dalam kasus penggunaan tertentu, bahkan ketika ada kesalahan
- Tingkat isolasi transaksi, yang mengontrol kualitas data yang dibaca dari database
- Logging, untuk memahami jenis log yang ditangkap oleh desain saat ini dan lokasi penyimpanannya

Hasil dari fase penemuan dapat berupa inventaris, seperti yang ditunjukkan tabel berikut.

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

Inventaris ini mungkin mencakup informasi berikut:

- Package: Nama paket SSIS yang akan dimigrasikan
- Aliran: Mengontrol aliran atau aliran data
- Tugas: Nama tugas aliran kontrol atau komponen aliran data
- Hitung: Berapa kali tugas digunakan dalam paket SSIS

Fase analisis

Menganalisis informasi dari fase penemuan membantu Anda memahami pola dan merancang arsitektur target dengan lebih baik. Ikuti petunjuk ini untuk meningkatkan hasil analisis:

- Opsi logging di tingkat paket mencatat peristiwa dan menangkap informasi runtime untuk setiap komponen. Gunakan pencatatan khusus sehingga Anda dapat mengonfigurasi file log untuk menyertakan informasi tambahan, seperti ID eksekusi dan nilai runtime lainnya.
- Arahkan catatan buruk ke lokasi penyimpanan yang berbeda untuk analisis, koreksi, dan pemrosesan ulang.
- Memahami strategi arsip dan pembersihan data yang diterapkan di lingkungan SSIS lokal Anda.
- Kirim peringatan dan pemberitahuan dengan menggunakan tugas yang disertakan dengan SSIS (seperti tugas Kirim Surat) atau skrip khusus (seperti tugas Skrip).
- Memahami perilaku transformasi dalam ruang lingkup untuk menghindari data yang rusak. Misalnya, transformasi Lookup adalah gabungan antara dua kumpulan data, dan perbandingannya peka huruf besar/kecil.

Anda dapat memetakan setiap entri dalam inventaris ke perkiraan kompleksitas, baik dalam hal durasi (menit atau jam) atau level (sederhana, sedang, atau kompleks). Inventaris yang diperluas ini, ditunjukkan pada tabel berikut, merupakan hasil dari fase analisis.

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

Menentukan pendekatan migrasi

Untuk memutuskan pendekatan migrasi, Anda menggunakan analisis yang Anda lakukan pada pola yang ada di fase sebelumnya. Kebutuhan data dan analitik masa depan organisasi Anda adalah pertimbangan yang sama pentingnya. Alat ETL lokal tradisional menangani model data relasional dan data terstruktur. Jika Anda memiliki data semi-terstruktur dan tidak terstruktur untuk diproses, Anda dapat menggunakan AWS layanan seperti atau AWS Glue Amazon EMR untuk migrasi. Faktor-faktor lain yang dapat mempengaruhi pendekatan migrasi meliputi:

- Apakah Anda ingin menggunakan antarmuka grafis (seperti AWS Glue Studio) atau kerangka kerja khusus (seperti pustaka Spark/Python)
- Apakah Anda memiliki akses aman ke sumber dan AWS target lokal
- Keterampilan dan pelatihan yang dibutuhkan untuk tim
- Persyaratan audit dan kepatuhan

Anda dapat memilih dari tiga pendekatan migrasi: big bang, phased, dan lift and shift. Tabel berikut membandingkan ketiga pendekatan ini.

Pendekatan	Deskripsi	Kasus penggunaan	Keuntungan dan kerugian
Dentuman besar	Migrasikan semua paket SSIS dalam jangka waktu tertentu.	• Kompleksitas, ruang lingkup, dan arsitektur target jelas. • Tim memiliki keterampilan yang dibutuhkan, atau kurva belajarnya dangkal.	• Risiko tinggi. • Membutuhkan waktu lebih sedikit daripada pendekatan bertahap. • Anda dapat menggunakan AWS Glue, Amazon EMR, atau kerangka kerja khusus.
Bertahap	Identifikasi satu paket SSIS untuk setiap pola dan kompleksitas yang berbeda. Migrasikan paket ke AWS, uji, dan bandingkan hasil dengan arsitektur yang ada.	• Waktu bukanlah kendala. • Anda menginginkan desain yang berbeda untuk pola ETL yang berbeda.	• Kurang berisiko daripada pendekatan big bang tetapi membutuhkan lebih banyak waktu dan usaha. • Anda dapat menggunakan AWS Glue, Amazon EMR, atau kerangka kerja khusus.
Angkat dan geser	Migrasikan arsitektur saat ini sebagaimana adanya. AWS	• Perangkat keras lokal Anda tidak lagi didukung. • Anda tidak memiliki sumber daya untuk merencanakan migrasi segera.	• Minimal jumlah upaya migrasi dan waktu yang dibutuhkan. • Masalah dengan solusi yang ada tetap ada AWS.

Pendekatan	Deskripsi	Kasus penggunaan	Keuntungan dan kerugian
			• Paket SSIS dijalankan apa adanya. Tidak ada alat atau kerangka kerja ETL lain yang diperlukan.

Perbandingan data pada sumber dan sistem target sangat penting untuk migrasi yang sukses. Karena sistem produksi yang ada mendapat pembaruan rutin dari sistem sumber, perbandingan ini mungkin menjadi membingungkan. Untuk alasan ini, saat Anda menentukan pendekatan migrasi, sebaiknya Anda juga memutuskan strategi validasi data Anda.

- Ambil cadangan semua database dan file yang berlaku dari lingkungan produksi pada sistem sumber pada tanggal dan waktu tertentu.
- Ambil cadangan semua database dari lingkungan produksi pada sistem target setelah semua pekerjaan berhasil memuat data dari data sumber yang dicadangkan.
- Kembalikan data sumber di lingkungan pengujian, dan jalankan pekerjaan baru.
- Setujui persentase perbedaan yang valid antara basis data sumber dan target (lama dan baru). Misalnya, Anda mungkin memutuskan bahwa perbedaan kurang dari 1% dapat diterima.
- Buat daftar semua aturan validasi yang akan dicakup.
- Otomatiskan perbandingan sebanyak mungkin, dan tutupi semua aturan.

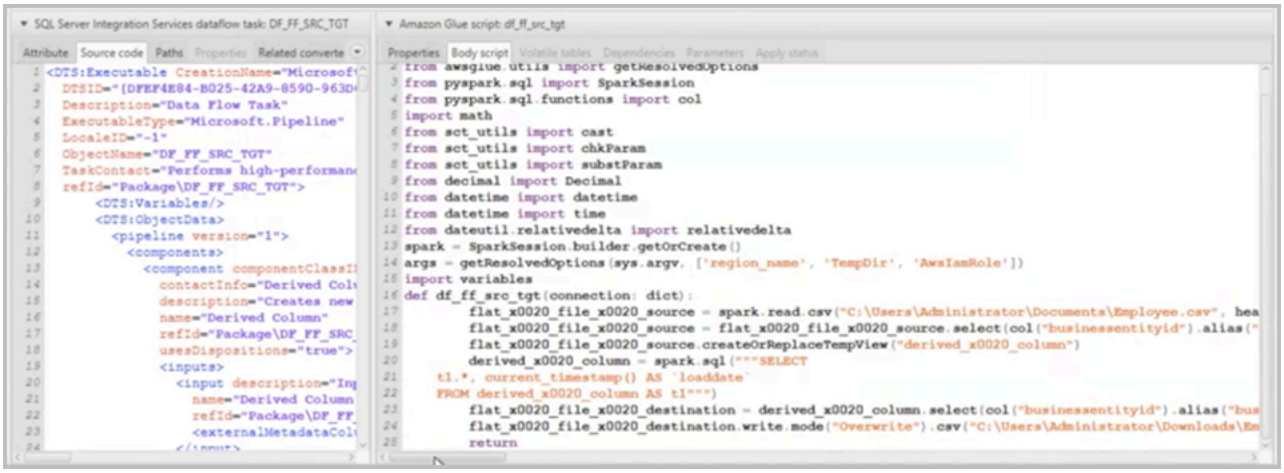
Tahap implementasi

Migrasi yang mengikuti big bang atau pendekatan bertahap membutuhkan pengembangan dan pengujian baru. The AWS Schema Conversion Tool (AWS SCT) dapat secara otomatis menghasilkan AWS Glue pekerjaan dari paket SSIS. Ini mengurangi waktu dan upaya migrasi secara signifikan. Atau, Anda dapat menggunakan AWS Glue Studio untuk pengembangan berbasis antarmuka grafis, atau membangun pustaka Spark yang dapat Anda jalankan di salah satu atau Amazon EMR. AWS Glue

Bagian berikut memberikan petunjuk yang berguna untuk menggunakan AWS SCT, AWS Glue, dan Amazon EMR.

AWS SCT

Ilustrasi layar berikut menunjukkan skrip AWS Glue pekerjaan yang dikonversi oleh AWS SCT.

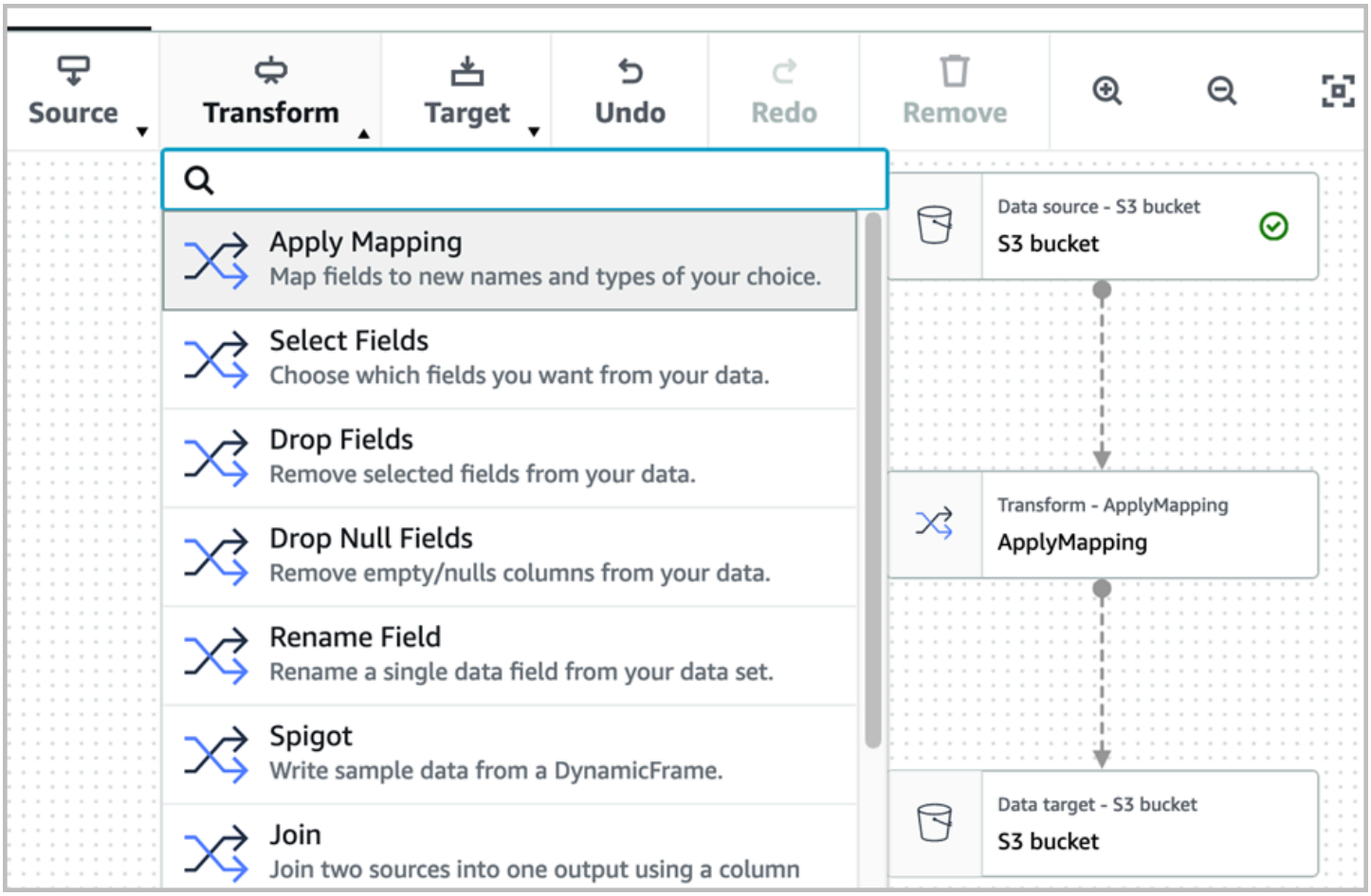


AWS SCT dapat mengonversi paket SSIS menjadi AWS Glue pekerjaan dalam jumlah besar. Anda dapat mengedit skrip untuk memperbarui logika yang ada atau menambahkan logika baru, berdasarkan desain baru Anda. Kami menyarankan Anda mengikuti konvensi penamaan dalam skrip yang AWS SCT dikonversi untuk menyesuaikan skrip.

Untuk informasi selengkapnya, lihat Mengonversi SSIS ke AWS Glue penggunaan AWS SCT dalam dokumentasi. AWS SCT

AWS Glue

AWS Glue Studio menyediakan antarmuka grafis dan pengalaman pengembangan yang mirip dengan SSIS, seperti yang diilustrasikan di layar berikut.



Jika Anda memilih untuk tidak menggunakan antarmuka grafis, Anda juga dapat menjalankan skrip kustom Anda dengan pustaka Python yang diperlukan dari konsol. AWS Glue Untuk informasi selengkapnya, lihat Menyediakan skrip kustom Anda sendiri dalam AWS Glue dokumentasi.

AWS Glue menyediakan satu set transformasi bawaan untuk memproses data Anda. Ini mirip dengan transformasi aliran data SSIS. Ikuti praktik terbaik ini saat Anda memigrasikan pekerjaan ETL SSIS Anda dengan menggunakan: AWS Glue

- Siapkan pemetaan dari AWS Glue transformasi ke transformasi SSIS yang setara.
- Jika transformasi Anda tidak dapat dipetakan ke AWS Glue transformasi, buat dengan menggunakan skrip kustom Python atau Scala.
- Untuk pencatatan kustom (seperti baris yang dibaca, baris yang ditulis, atau catatan buruk), gunakan skrip khusus selain Amazon CloudWatch.
- Tambahkan titik akhir pengembangan untuk mengembangkan dan men-debug skrip kustom secara lokal.

Amazon EMR

Anda dapat menjalankan skrip kustom (ditulis dengan Python atau Scala) atau pustaka Python yang dikompilasi dalam cluster EMR, seperti halnya. AWS Glue Ikuti praktik terbaik ini:

- Mulailah dengan jenis instans yang dioptimalkan memori sambil membuat cluster EMR dengan kerangka kerja Spark. (SSIS menggunakan buffer memori.)
- Bangun metode Python generik yang setara dengan setiap tugas atau transformasi SSIS. Misalnya, dalam ilustrasi berikut, metode yang mengambil dua kerangka data sebagai input menghasilkan kerangka data ketiga yang memiliki catatan yang cocok dari dua kerangka data sebagai output. Ini berfungsi sebagai transformasi gabungan gabungan.

```
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
```

Pengujian

Kerangka pengujian diperlukan untuk memvalidasi kelengkapan dan kebenaran data. Kerangka kerja ini harus mencakup semua skenario yang ada dan perbaikan apa pun yang Anda buat saat memigrasikan AWS pekerjaan Anda.

- Validasi kelengkapan:
  - Semua pekerjaan dimigrasikan ke status target mereka.
  - Semua fungsionalitas dimigrasikan di setiap pekerjaan.

    - Semua jenis log tersedia, termasuk rincian eksekusi pekerjaan, pesan kesalahan, catatan buruk, dan jumlah baris.
- Validasi kebenaran:
    - Kualitas data konsisten di lingkungan yang ada dan yang baru.
    - Semua kolom dari semua tabel cocok, atau tabel ditingkatkan AWS.
    - Semua informasi audit dan pencatatan cocok.

Anda juga harus memverifikasi bahwa kinerja pekerjaan yang dimigrasi sesuai dengan kinerja pekerjaan yang ada.

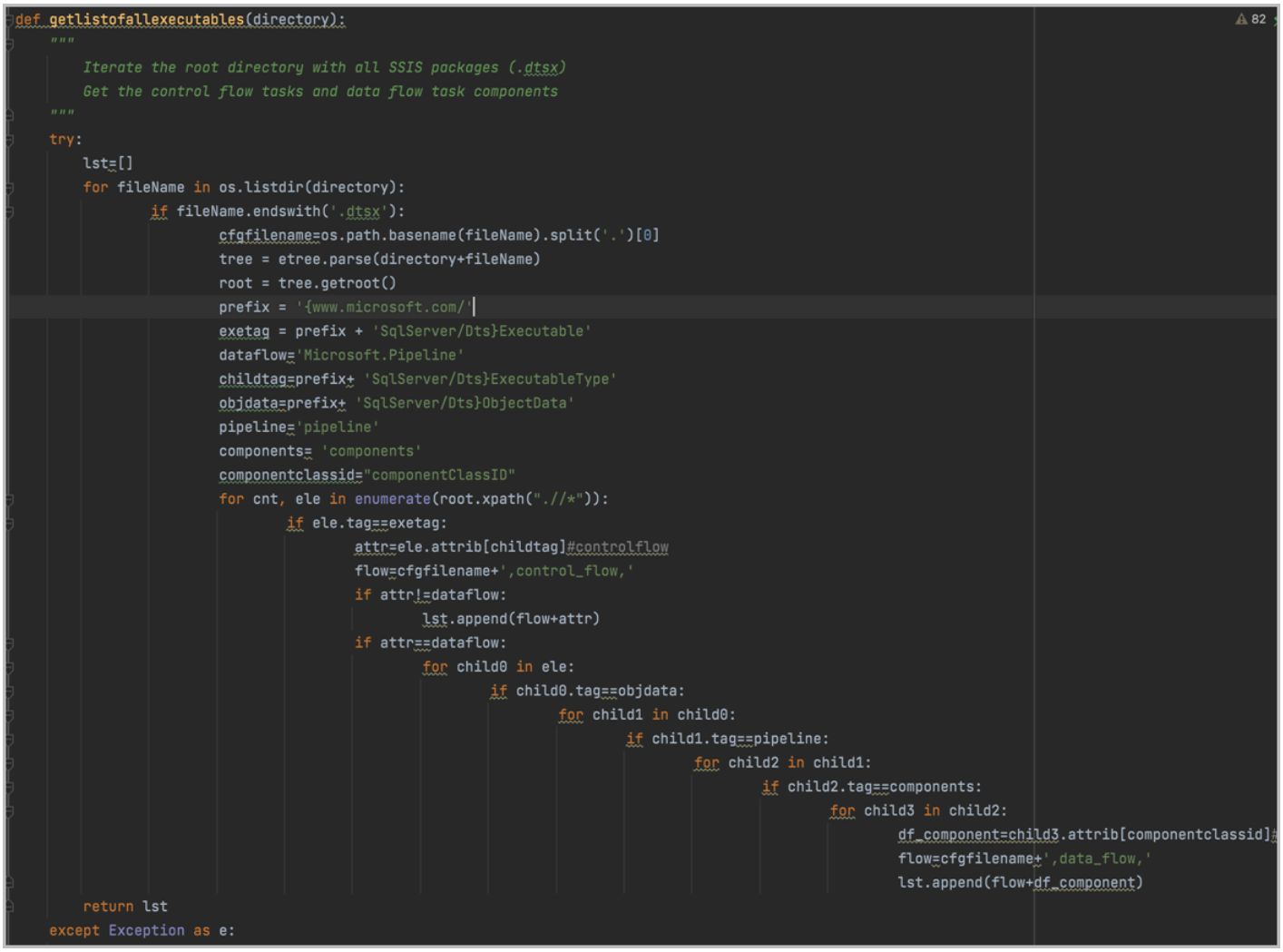
Praktik terbaik

Ikuti praktik terbaik umum ini saat memigrasikan pekerjaan SSIS Anda ke: AWS

- Jika Anda memigrasikan database ke layanan AWS terkelola seperti Amazon Relational Database Service (Amazon RDS), tugas pemeliharaan descope (seperti pencadangan, pemulihan, pembaruan statistik, atau periksa integritas database) yang tidak berlaku atau dikelola oleh. AWS
- Bangun skrip dan metode yang dapat digunakan kembali untuk menstandarisasi pengembangan dan mengurangi upaya.
- Selalu uji pekerjaan yang dimigrasi pada infrastruktur dan volume data yang sesuai dengan sumber daya produksi.
- Paket SSIS dalam format XHTML. Bangun utilitas parser XMLuntuk mengotomatiskan aktivitas migrasi jika memungkinkan. Contoh berikut menunjukkan paket SSIS dalam format XHTML.

```
    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
```

Ilustrasi berikut menunjukkan contoh parser XHTML.



```
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:
```

- Gunakan checksum dan nilai hash untuk menguji kebenaran dan kelengkapan.
- Jika Anda membangun pustaka khusus, terapkan threading untuk menjalankan tugas secara paralel. Ini meningkatkan kinerja pekerjaan.
- Menonaktifkan lingkungan yang ada hanya setelah semua pekerjaan stabil AWS untuk jangka waktu tertentu.
- Gunakan Amazon CloudWatch untuk logging, untuk mengurangi upaya kustomisasi logging.
- Gunakan Amazon Simple Notification Service (Amazon SNS) untuk mengganti tugas khusus untuk mengirim notifikasi. Gunakan Amazon Simple Email Service (Amazon SES) untuk mengganti tugas email khusus.

Pertanyaan yang Sering Diajukan

Bagian ini memberikan jawaban atas pertanyaan yang sering diajukan tentang memigrasi pekerjaan ETL SSIS Anda ke. AWS

Haruskah saya meningkatkan pekerjaan SSIS selama migrasi?

Ya. Menerapkan penyempurnaan penting atau perbaikan bug di pekerjaan yang ada dan baru pada saat yang bersamaan. Perubahan prioritas rendah dapat diterapkan setelah migrasi.

Bagaimana saya bisa memantau pekerjaan AWS?

Anda dapat menggunakan Amazon CloudWatch untuk memantau pekerjaan dan mengirim peringatan berbasis aturan.

Kapan saya dapat menonaktifkan pekerjaan SSIS lokal saya?

Kami menyarankan Anda mempertahankan pekerjaan lama dan baru secara paralel untuk jangka waktu tertentu hingga kinerja, akurasi, dan kelengkapan data sama di kedua lingkungan. Anda juga dapat mereplikasi sistem pelaporan untuk menghubungkan dan membandingkan laporan dari kedua lingkungan. Anda dapat menonaktifkan pekerjaan SSIS Anda ketika laporannya identik.

Langkah selanjutnya

Sebagai langkah selanjutnya, kami menyarankan Anda membangun bukti konsep untuk menentukan pendekatan penerapan Anda untuk arsitektur target. Untuk bukti konsep Anda:

- Buat AWS Glue pekerjaan dengan menggunakan antarmuka grafis AWS Glue Studio.
- Gunakan AWS Schema Conversion Tool (AWS SCT) untuk menghasilkan AWS Glue skrip dari paket SSIS.
- Bangun kerangka kerja migrasi kustom dengan menggunakan pustaka Spark.
- Membangun kerangka pengujian.
- Identifikasi alat dan proses untuk integrasi berkelanjutan dan penyebaran berkelanjutan (CI/CD).

Sumber daya terkait

- Mengonversi SSIS untuk menggunakan AWS GlueAWS SCT
- Pencadangan dan pemulihan Amazon RDS
- AWS Glue dokumentasi
- AWS Glue FAQs
- Pencatatan terus menerus untuk AWS Glue pekerjaan
- Dokumentasi Amazon EMR
- Amazon EMR FAQs
- Pantau metrik dengan CloudWatch
- Mempercepat validasi migrasi data skala besar menggunakan PyDeequ
- AWS Mitra Kompetensi Data dan Analitik

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Publikasi awal	—	Desember 6, 2021

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.