



Membuat saluran pipa ML-siap produksi di AWS

AWS Panduan Preskriptif



AWS Panduan Preskriptif: Membuat saluran pipa ML-siap produksi di AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Pengantar	1
.....	1
.....	3
.....	4
Menggunakan pekerjaan pemrosesan	4
Menggunakan klausa penjaga utama	6
Pengujian unit	6
.....	8
Menggunakan Step Functions SDK	8
Memperluas Step Functions SDK	9
.....	10
Implementasi dengan AWS CloudFormation	10
Memodifikasi output dari Step Functions SDK	11
.....	12
.....	14
Tingkat otomatisasi yang berbeda	14
Platform yang berbeda untuk beban kerja ML	15
Mesin yang berbeda untuk orkestrasi pipa	16
.....	17
Sumber daya tambahan	18
Riwayat dokumen	19
.....	xx

Membuat saluran pipa ML-siap produksi di AWS

Josiah Davis, Verdi March, Yin Song, Baichuan Sun, Chen Wu, dan Wei Yih Yap, Amazon Web Services (AWS)

Januari 2021 ([riwayat dokumen](#))

Proyek pembelajaran mesin (ML) membutuhkan upaya multi-tahap yang signifikan yang mencakup pemodelan, implementasi, dan produksi untuk memberikan nilai bisnis dan untuk memecahkan masalah dunia nyata. Banyak alternatif dan opsi penyesuaian tersedia di setiap langkah, dan ini membuatnya semakin menantang untuk menyiapkan model ML untuk produksi dalam batasan sumber daya dan anggaran Anda. Selama beberapa tahun terakhir di Amazon Web Services (AWS), tim Ilmu Data kami telah bekerja dengan berbagai sektor industri dalam inisiatif ML. Kami mengidentifikasi titik-titik nyeri yang dimiliki oleh banyak AWS pelanggan, yang berasal dari masalah organisasi dan tantangan teknis, dan kami telah mengembangkan pendekatan optimal untuk memberikan solusi ML siap produksi.

Panduan ini ditujukan untuk ilmuwan data dan insinyur ML yang terlibat dalam implementasi pipa ML. Ini menjelaskan pendekatan kami untuk memberikan jaringan pipa ML siap produksi. Panduan ini membahas bagaimana Anda dapat bertransisi dari menjalankan model ML secara interaktif (selama pengembangan) ke menerapkannya sebagai bagian dari pipeline (selama produksi) untuk kasus penggunaan ML Anda. Untuk tujuan ini, kami juga telah mengembangkan satu set contoh template (lihat [proyek proyek MLMax](#)), untuk mempercepat pengiriman solusi HTML kustom ke produksi, sehingga Anda dapat memulai dengan cepat tanpa harus membuat terlalu banyak pilihan desain.

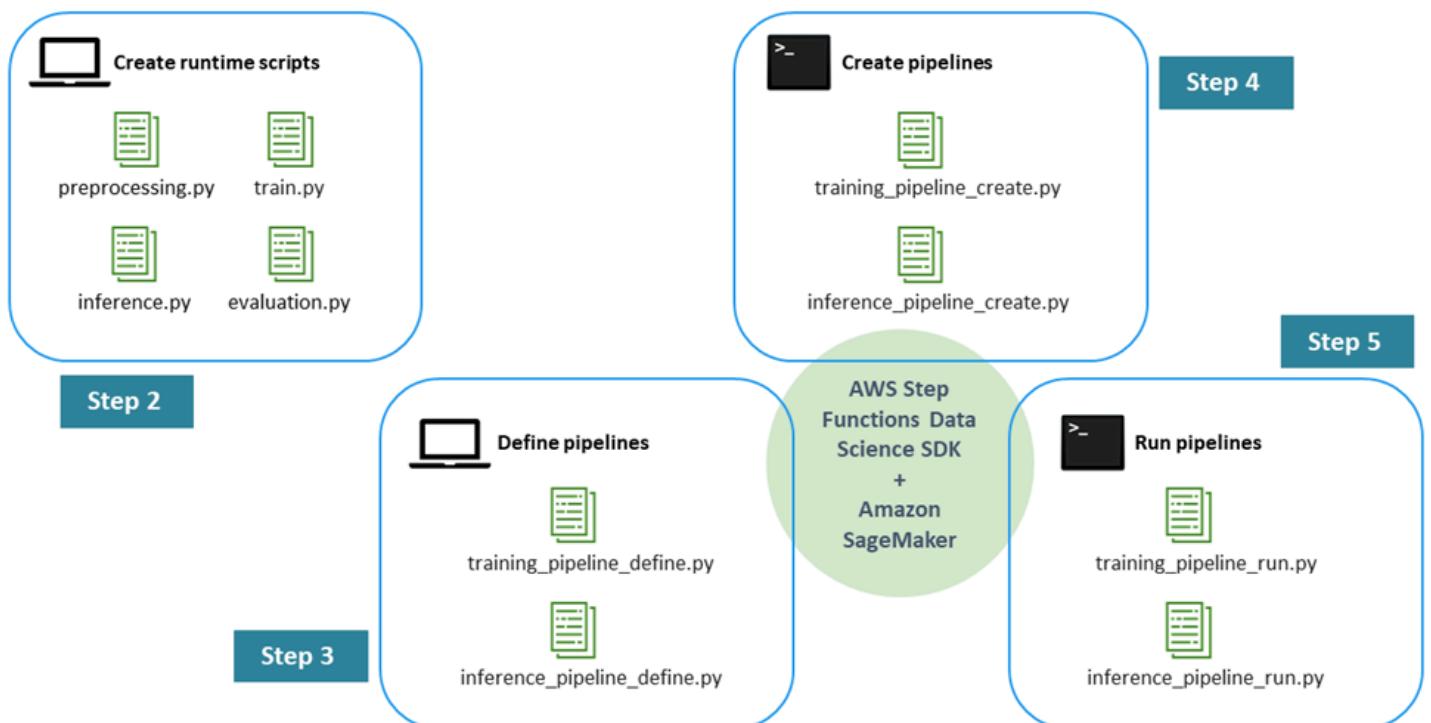
Gambaran umum

Proses pembuatan pipa ML siap produksi terdiri dari langkah-langkah berikut:

- [Langkah 1](#). Lakukan EDA dan kembangkan model awal — Ilmuwan data membuat data mentah tersedia di Amazon Simple Storage Service (Amazon S3), melakukan analisis data eksplorasi (EDA), mengembangkan model ML awal, dan mengevaluasi kinerja inferensinya. Anda dapat melakukan kegiatan ini secara interaktif melalui notebook Jupyter.
- [Langkah 2](#). Buat skrip runtime — Anda mengintegrasikan model dengan skrip Python runtime sehingga dapat dikelola dan disediakan oleh kerangka kerja HTML (dalam kasus kami, Amazon AI). SageMaker ini adalah langkah pertama dalam menjauh dari pengembangan interaktif model mandiri menuju produksi. Secara khusus, Anda mendefinisikan logika untuk preprocessing, evaluasi, pelatihan, dan inferensi secara terpisah.

- [Langkah 3](#). Tentukan pipeline — Anda menentukan placeholder input dan output untuk setiap langkah pipeline. Nilai konkret untuk ini akan diberikan nanti, selama runtime (langkah 5). Anda fokus pada pipeline untuk pelatihan, inferensi, validasi silang, dan pengujian kembali.
- [Langkah 4](#). Buat pipeline — Anda membuat infrastruktur yang mendasarinya, termasuk instans mesin AWS Step Functions status secara otomatis (hampir satu klik), dengan menggunakan AWS CloudFormation.
- [Langkah 5](#). Jalankan pipeline — Anda menjalankan pipeline yang ditentukan pada langkah 4. Anda juga menyiapkan metadata dan data atau lokasi data untuk mengisi nilai konkret untuk input/output placeholder yang Anda tentukan di langkah 3. Ini termasuk skrip runtime yang ditentukan pada langkah 2 serta model hyperparameters.
- [Langkah 6](#). Perluas pipeline — Anda menerapkan proses integrasi berkelanjutan dan penerapan berkelanjutan (CI/CD), pelatihan ulang otomatis, inferensi terjadwal, dan ekstensi pipeline yang serupa.

Diagram berikut menggambarkan langkah-langkah utama dalam proses ini.



Steps for creating the training and inference pipeline

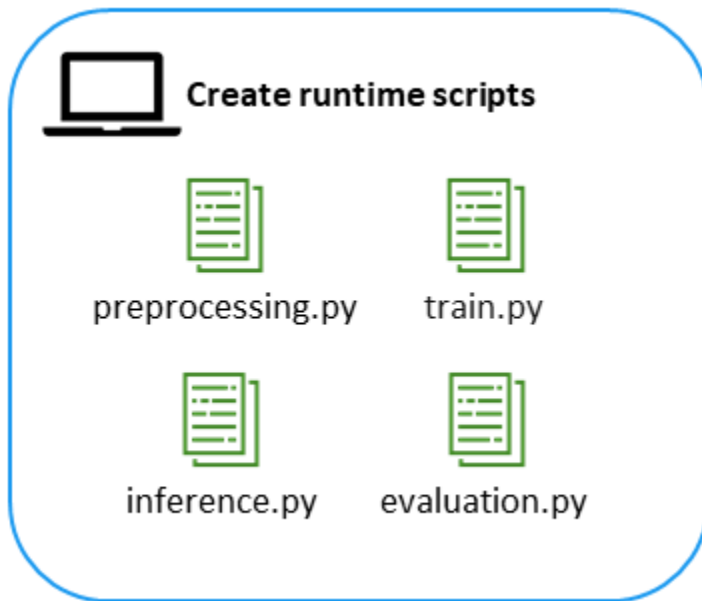
Langkah 1. Lakukan EDA dan kembangkan model awal

Pada langkah ini, ilmuwan data melakukan analisis data eksplorasi (EDA) untuk memahami kasus penggunaan dan data ML. Mereka kemudian mengembangkan model ML (misalnya, model klasifikasi dan regresi) untuk memecahkan masalah dalam kasus penggunaan tertentu. Selama pengembangan model, ilmuwan data sering membuat asumsi tentang input dan output, seperti format data, siklus hidup data, dan lokasi output menengah. Asumsi ini harus didokumentasikan sehingga dapat digunakan untuk verifikasi selama pengujian unit pada langkah 2.

Meskipun langkah ini berfokus pada pengembangan model, ilmuwan data seringkali harus menulis kode pembantu dalam jumlah minimum untuk preprocessing, pelatihan, evaluasi, dan inferensi. Ilmuwan data harus dapat menjalankan kode ini di lingkungan pengembangan. Kami juga merekomendasikan untuk menyediakan argumen runtime opsional sehingga kode pembantu ini dapat dikonfigurasi secara dinamis untuk berjalan di lingkungan lain tanpa perubahan manual yang ekstensif. Ini akan mempercepat integrasi antara model dan pipa pada langkah 2 dan 3. Misalnya, kode untuk membaca data mentah harus dienkapsulasi dalam fungsi sehingga data dapat diproses sebelumnya secara konsisten.

Kami menyarankan Anda memulai dengan kerangka kerja seperti [scikit-learn](#), [Keras](#), [XGBoostPyTorch](#), atau [TensorFlow](#) untuk mengembangkan model ML dan kode penolongnya. Misalnya, scikit-learn adalah pustaka HTML gratis yang ditulis dengan Python. Ini menyediakan konvensi API yang seragam untuk objek, dan mencakup empat objek utama - estimator, prediktor, transformator, dan model - yang mencakup transformasi data ringan, mendukung label dan rekayasa fitur, dan merangkum langkah-langkah pra-pemrosesan dan pemodelan. Objek-objek ini membantu menghindari proliferasi kode boilerplate dan mencegah validasi dan data pengujian bocor ke dalam kumpulan data pelatihan. Demikian pula, setiap framework ML memiliki implementasi sendiri dari artefak kunci ML, dan kami menyarankan Anda untuk mematuhi konvensi API dari framework yang Anda pilih ketika Anda mengembangkan model ML.

Langkah 2. Buat skrip runtime



Pada langkah ini, Anda mengintegrasikan model yang Anda kembangkan pada langkah 1 dan kode pembantu terkait ke dalam platform ML untuk pelatihan dan inferensi siap produksi. Secara khusus, ini melibatkan pengembangan skrip runtime sehingga model dapat dimasukkan ke dalam SageMaker AI. Skrip Python mandiri ini mencakup fungsi callback AI yang SageMaker telah ditentukan sebelumnya dan variabel lingkungan. Mereka dijalankan di dalam wadah SageMaker AI yang di-host di instance Amazon Elastic Compute Cloud (Amazon EC2). [Dokumentasi Amazon SageMaker AI Python SDK](#) memberikan informasi terperinci tentang bagaimana callback dan pengaturan tambahan ini bekerja sama untuk pelatihan dan inferensi. Bagian berikut memberikan rekomendasi tambahan untuk mengembangkan skrip runtime HTML, berdasarkan pengalaman kami bekerja dengan AWS pelanggan.

Menggunakan pekerjaan pemrosesan

SageMaker AI menyediakan dua opsi untuk melakukan inferensi model mode batch. Anda dapat menggunakan pekerjaan pemrosesan SageMaker AI atau pekerjaan transformasi batch. Setiap opsi memiliki kelebihan dan kekurangan.

Pekerjaan pemrosesan terdiri dari file Python yang berjalan di dalam wadah SageMaker AI. Pekerjaan pemrosesan terdiri dari logika apa pun yang Anda masukkan ke dalam file Python Anda. Ini memiliki kelebihan ini:

- Ketika Anda memahami logika dasar pekerjaan pelatihan, pekerjaan pemrosesan mudah diatur dan mudah dipahami. Mereka berbagi abstraksi yang sama dengan pekerjaan pelatihan (misalnya, menyetel jumlah instance dan distribusi data).
- Ilmuwan data dan insinyur ML memiliki kontrol penuh atas opsi manipulasi data.
- Ilmuwan data tidak harus mengelola logika I/O komponen apa pun kecuali untuk read/write fungsionalitas yang sudah dikenal.
- Agak lebih mudah untuk menjalankan file di lingkungan SageMaker non-AI, yang membantu pengembangan cepat dan pengujian lokal.
- Jika ada kesalahan, pekerjaan pemrosesan gagal segera setelah skrip gagal, dan tidak ada menunggu yang tidak terduga untuk mencoba lagi.

Di sisi lain, pekerjaan transformasi batch merupakan perpanjangan dari konsep titik akhir SageMaker AI. Saat runtime, pekerjaan ini mengimpor fungsi callback, yang kemudian menangani I/O untuk membaca data, memuat model, dan membuat prediksi. Pekerjaan transformasi Batch memiliki keuntungan sebagai berikut:

- Mereka menggunakan abstraksi distribusi data yang berbeda dari abstraksi yang digunakan oleh pekerjaan pelatihan.
- Mereka menggunakan file inti dan struktur fungsi yang sama untuk inferensi batch dan inferensi waktu nyata, yang nyaman.
- Mereka memiliki mekanisme toleransi kesalahan berbasis coba ulang bawaan. Misalnya, jika terjadi kesalahan pada sekumpulan catatan, itu akan mencoba lagi beberapa kali sebelum pekerjaan dihentikan sebagai kegagalan.

Karena transparansi, kemudahan penggunaannya di berbagai lingkungan, dan abstraksi bersama dengan pekerjaan pelatihan, kami memutuskan untuk menggunakan pekerjaan pemrosesan alih-alih pekerjaan transformasi batch dalam arsitektur referensi yang disajikan dalam panduan ini.

Anda harus menjalankan skrip runtime Python secara lokal sebelum Anda menerapkannya di cloud. Secara khusus, kami menyarankan Anda menggunakan klausa penjaga utama ketika Anda menyusun skrip Python Anda, dan melakukan pengujian unit.

Menggunakan klausa penjaga utama

Gunakan klausa penjaga utama untuk mendukung impor modul dan menjalankan skrip Python Anda. Menjalankan skrip Python satu per satu bermanfaat untuk men-debug dan mengisolasi masalah dalam pipeline ML. Kami merekomendasikan langkah-langkah berikut:

- Gunakan parser argumen dalam file pemrosesan Python untuk input/output menentukan file dan lokasinya.
- Berikan panduan utama dan fungsi uji untuk setiap file Python.
- Setelah Anda menguji file Python, masukkan ke dalam tahapan yang berbeda dari pipeline ML, apakah Anda menggunakan AWS Step Functions model atau pekerjaan pemrosesan SageMaker AI.
- Gunakan pernyataan Assert di bagian kritis skrip untuk memfasilitasi pengujian dan debugging. Misalnya, Anda dapat menggunakan pernyataan Assert untuk memastikan bahwa jumlah fitur dataset konsisten setelah pemuatan.

Pengujian unit

Pengujian unit skrip runtime yang ditulis untuk pipeline adalah tugas penting yang sering diabaikan dalam pengembangan pipeline ML. Ini karena pembelajaran mesin dan ilmu data adalah bidang yang relatif baru dan lambat untuk mengadopsi praktik rekayasa perangkat lunak yang mapan seperti pengujian unit. Karena pipa ML akan digunakan di lingkungan produksi, penting untuk menguji kode pipa sebelum menerapkan model ML ke aplikasi dunia nyata.

Unit pengujian skrip runtime juga memberikan manfaat unik berikut untuk model ML:

- Ini mencegah transformasi data yang tidak terduga. Sebagian besar jaringan pipa ML melibatkan banyak transformasi data, sehingga sangat penting untuk transformasi ini untuk melakukan seperti yang diharapkan.
- Ini memvalidasi reproduktifitas kode. Setiap keacakan dalam kode dapat dideteksi dengan pengujian unit dengan kasus penggunaan yang berbeda.
- Ini memberlakukan modularitas kode. Tes unit biasanya dikaitkan dengan ukuran cakupan pengujian, yang merupakan sejauh mana rangkaian pengujian tertentu (kumpulan kasus uji) menjalankan kode sumber suatu program. Untuk mencapai cakupan pengujian yang tinggi, pengembang memodulasi kode, karena sulit untuk menulis pengujian unit untuk sejumlah besar kode tanpa memecahnya menjadi fungsi atau kelas.

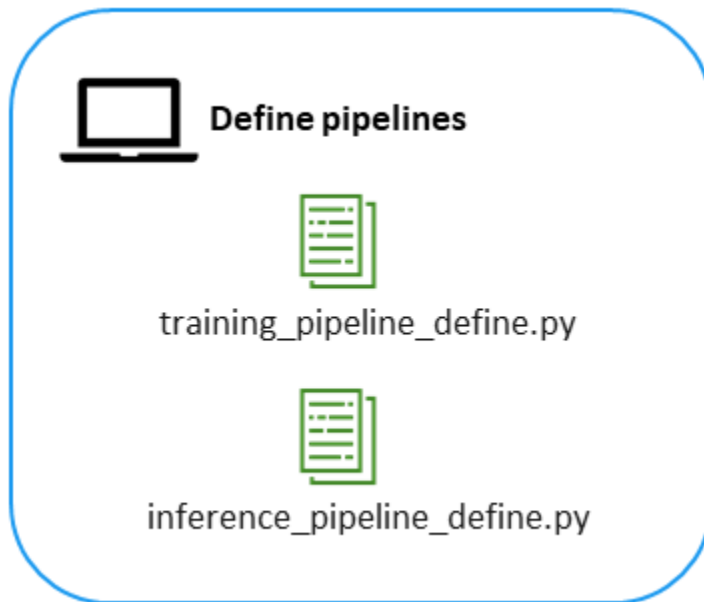
- Ini mencegah kode atau kesalahan berkualitas rendah dimasukkan ke dalam produksi.

Kami menyarankan Anda menggunakan kerangka pengujian unit yang matang seperti [pytest](#) untuk menulis kasus pengujian unit, karena lebih mudah untuk mengelola pengujian unit ekstensif dalam kerangka kerja.

 Important

Pengujian unit tidak dapat menjamin bahwa semua casing sudut diuji, tetapi dapat membantu Anda secara proaktif menghindari kesalahan sebelum Anda menerapkan model. Kami menyarankan Anda juga memantau model setelah penerapan, untuk memastikan keunggulan operasional.

Langkah 3. Tentukan pipa



Pada langkah ini, urutan dan logika tindakan yang akan dilakukan pipa ditentukan. Ini termasuk langkah-langkah diskrit serta input dan output logisnya. Misalnya, bagaimana keadaan data di awal pipa? Apakah itu berasal dari beberapa file yang berada pada tingkat granularitas yang berbeda atau dari satu file datar? Jika data berasal dari beberapa file, apakah Anda memerlukan satu langkah untuk semua file atau langkah terpisah untuk setiap file untuk menentukan logika pra-pemrosesan? Keputusan tergantung pada kompleksitas sumber data dan sejauh mana mereka diproses sebelumnya.

Dalam implementasi referensi kami, kami menggunakan [AWS Step Functions](#), yang merupakan orkestrator fungsi tanpa server, untuk menentukan langkah-langkah alur kerja. Namun, [kerangka kerja ML Max](#) juga mendukung sistem pipeline atau state machine lainnya seperti Apache AirFlow (lihat bagian [Different engine for pipeline orchestration](#)) untuk mendorong pengembangan dan penyebaran jaringan pipa ML.

Menggunakan Step Functions SDK

Untuk mendefinisikan pipeline ML, pertama-tama kita menggunakan API Python tingkat tinggi yang disediakan oleh [AWS Step Functions Data Science](#) SDK (Step Functions SDK) untuk mendefinisikan dua komponen utama dari pipeline: langkah dan data. Jika Anda menganggap pipeline sebagai grafik asiklik terarah (DAG), langkah-langkah mewakili node pada grafik, dan data ditampilkan sebagai tepi

terarah yang menghubungkan satu simpul (langkah) ke simpul berikutnya. Contoh khas langkah ML termasuk preprocessing, training, dan evaluasi. Step Functions SDK menyediakan sejumlah langkah bawaan (seperti [TrainingStep](#)) yang dapat Anda gunakan. Contoh data termasuk input, output, dan banyak dataset menengah yang dihasilkan oleh beberapa langkah dalam pipeline. Saat Anda mendesain pipeline ML, Anda tidak mengetahui nilai konkret item data. Anda dapat menentukan placeholder data yang berfungsi sebagai template (mirip dengan parameter fungsi) dan hanya berisi nama item data dan tipe data primitif. Dengan cara ini, Anda dapat merancang cetak biru pipa lengkap tanpa mengetahui nilai konkret data yang berjalan pada grafik terlebih dahulu. Untuk tujuan ini, Anda dapat menggunakan kelas placeholder di Step Functions SDK untuk memodelkan template data ini secara eksplisit.

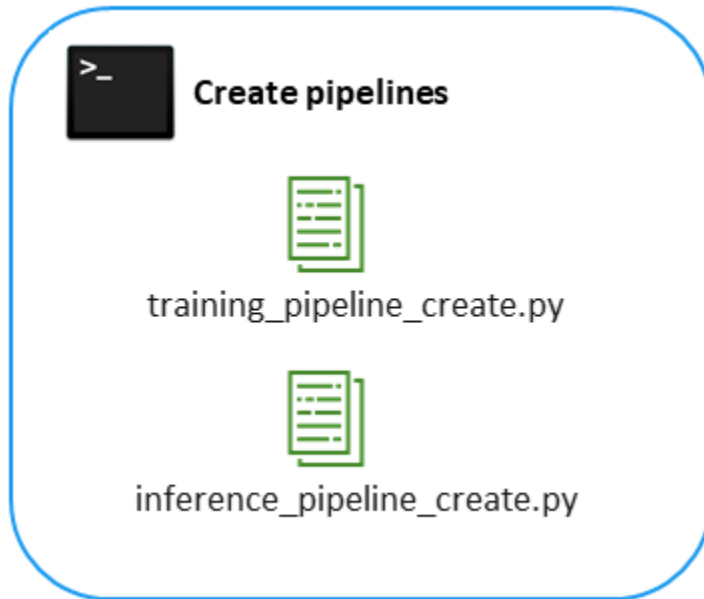
Saluran pipa ML juga memerlukan parameter konfigurasi untuk melakukan kontrol halus atas perilaku setiap langkah ML. Placeholder data khusus ini disebut placeholder parameter. Banyak dari nilainya tidak diketahui saat Anda mendefinisikan pipeline. Contoh placeholder parameter mencakup parameter terkait infrastruktur yang Anda tentukan selama desain pipeline (misalnya, Wilayah AWS atau URL gambar kontainer) dan parameter terkait pemodelan ML (seperti hyperparameters) yang Anda tentukan saat menjalankan pipeline.

Memperluas Step Functions SDK

Dalam implementasi referensi kami, salah satu persyaratan adalah memisahkan definisi pipa ML dari pembuatan dan penerapan pipa ML konkret dengan menggunakan pengaturan parameter tertentu. Namun, beberapa langkah bawaan di Step Functions SDK tidak memungkinkan kami untuk meneruskan semua parameter placeholder ini. Sebaliknya, nilai parameter diharapkan diperoleh secara langsung selama waktu desain pipeline melalui panggilan API konfigurasi SageMaker AI. Ini berfungsi dengan baik jika lingkungan waktu desain SageMaker AI identik dengan lingkungan runtime SageMaker AI, tetapi ini jarang terjadi dalam pengaturan dunia nyata. Kopling yang begitu erat antara waktu desain pipa dan runtime, dan asumsi bahwa infrastruktur platform ML akan tetap konstan, secara signifikan menghambat penerapan pipa yang dirancang. Faktanya, pipeline ML segera rusak ketika platform penerapan yang mendasarinya mengalami perubahan sekecil apa pun.

Untuk mengatasi tantangan ini dan menghasilkan pipeline HTML yang kuat (yang ingin kami rancang sekali dan jalankan di mana saja), kami menerapkan langkah-langkah khusus kami sendiri dengan memperluas beberapa langkah bawaan, termasuk `TrainingStep`, `ModelStep`, dan `TransformerStep`. Ekstensi ini disediakan dalam [proyek ML Max](#). Antarmuka langkah-langkah khusus ini mendukung placeholder parameter yang jauh lebih banyak yang dapat diisi dengan nilai konkret selama pembuatan pipa atau saat pipa berjalan.

Langkah 4. Buat pipa



Setelah Anda menentukan pipeline secara logis, saatnya membuat infrastruktur untuk mendukung pipeline. Langkah ini membutuhkan kemampuan berikut, minimal:

- Penyimpanan, untuk meng-host dan mengelola input dan output pipeline, termasuk kode, artefak model, dan data yang digunakan dalam pelatihan dan inferensi berjalan.
- Compute (GPU atau CPU), untuk pemodelan dan inferensi serta preprocessing data dan postprocessing.
- Orkestrasi, untuk mengelola sumber daya yang digunakan dan untuk menjadwalkan setiap proses reguler. Misalnya, model mungkin dilatih ulang secara periodik saat data baru tersedia.
- Pencatatan dan peringatan, untuk memantau akurasi model pipa, untuk pemanfaatan sumber daya, dan untuk pemecahan masalah.

Implementasi dengan AWS CloudFormation

Untuk membuat pipeline yang kami gunakan AWS CloudFormation, yang merupakan AWS layanan untuk menyebarkan dan mengelola infrastruktur sebagai kode. AWS CloudFormation Template menyertakan definisi Step Functions yang dibuat pada langkah sebelumnya dengan Step Functions SDK. Langkah ini mencakup pembuatan instance AWS-managed Step Functions, yang disebut mesin status Step Functions. Tidak ada sumber daya untuk pelatihan dan inferensi yang dibuat

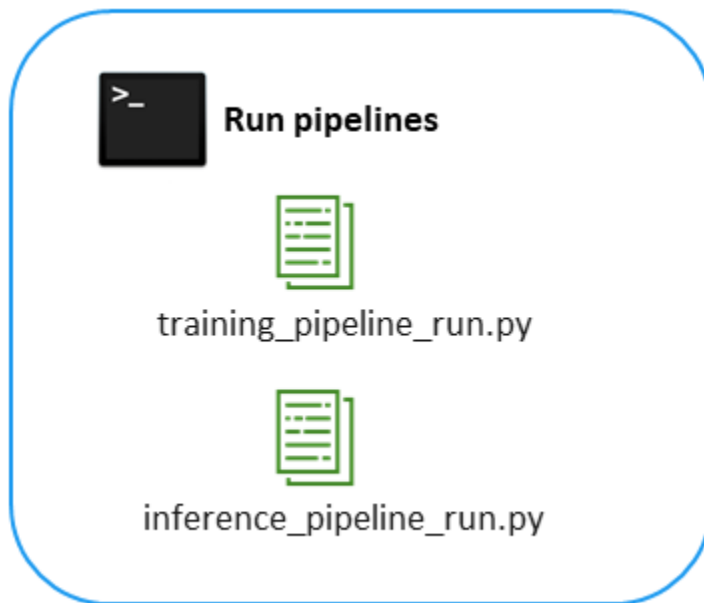
pada tahap ini, karena pekerjaan pelatihan dan inferensi berjalan sesuai permintaan, hanya ketika dibutuhkan, sebagai pekerjaan SageMaker AI. Langkah ini juga mencakup pembuatan peran AWS Identity and Access Management (IAM) untuk menjalankan Step Functions, menjalankan SageMaker AI, dan membaca dan menulis dari Amazon S3.

Memodifikasi output dari Step Functions SDK

Kami harus membuat beberapa modifikasi kecil pada CloudFormation output dari bagian sebelumnya. Kami menggunakan pencocokan string Python sederhana untuk melakukan hal berikut:

- Kami menambahkan logika untuk membuat `Parameters` bagian CloudFormation template. Ini karena kita ingin membuat dua peran dan mendefinisikan nama pipeline sebagai parameter bersama dengan lingkungan penerapan. Langkah ini juga mencakup sumber daya dan peran tambahan yang mungkin ingin Anda buat, seperti yang dibahas pada langkah 6.
- Kami memformat ulang tiga bidang untuk memiliki !Sub awalan dan tanda kutip yang diperlukan sehingga dapat diperbarui secara dinamis sebagai bagian dari proses penerapan:
 - `StateMachineNameProperti`, yang menamai mesin negara.
 - `DefinitionStringProperti`, yang mendefinisikan mesin negara.
 - `RoleArnProperti`, yang dikembalikan oleh mesin negara.

Langkah 5. Jalankan pipa



Langkah ini menjalankan jalur pelatihan atau inferensi yang dibuat di CloudFormation tumpukan pada langkah 4. Pipa tidak dapat dijalankan sampai parameter placeholder internalnya diisi dengan nilai konkret. Tindakan menetapkan nilai ke parameter placeholder ini adalah aktivitas utama langkah 5. Contoh parameter placeholder meliputi:

- Lokasi input, output, dan dataset menengah
- Lokasi Amazon S3 dari skrip runtime dan kode preprocessing atau evaluasi lainnya yang dikembangkan pada langkah 2 (misalnya, `sm_submit_url` untuk pipeline pelatihan)
- Nama AWS Daerah

Anda harus memastikan bahwa nilai jalur ini mengarah ke data atau kode yang valid sebelum Anda menjalankan pipeline. Misalnya, jika Anda mengisi parameter placeholder yang mewakili URL Amazon S3 dari skrip runtime Python, Anda harus mengunggah skrip tersebut ke URL tersebut. Orang yang menjalankan pipeline bertanggung jawab atas pemeriksaan konsistensi dan pengunggahan data. Orang yang mendefinisikan atau membuat pipa tidak perlu khawatir tentang semua ini.

Bergantung pada kematangan pipa, langkah ini dapat diotomatisasi untuk dijalankan secara reguler (mingguan atau bulanan). Otomasi juga membutuhkan pemantauan yang kuat, yang merupakan area penting tetapi di luar cakupan panduan ini. Untuk menjalankan jalur pelatihan, akan tepat

untuk memantau metrik evaluasi. Untuk pipa inferensi, akan tepat untuk memantau penyimpangan distribusi data input, dan, jika mungkin, mengumpulkan label secara berkala dan mengukur penyimpangan dalam akurasi prediksi. Catatan-catatan ini dari pelatihan dan inferensi berjalan harus dicatat dalam database untuk analisis di kemudian hari.

Langkah 6. Perluas pipa

Panduan ini menjelaskan bagaimana Anda dapat mulai membangun jaringan pipa ML AWS dengan cepat, dengan arsitektur beton. Ada pertimbangan tambahan untuk pematangan pipa, seperti manajemen metadata, pelacakan eksperimen, dan pemantauan. Ini adalah topik penting yang berada di luar cakupan panduan ini. Bagian berikut membahas aspek lain dari manajemen pipa, yaitu otomatisasi pipa.

Tingkat otomatisasi yang berbeda

Meskipun Anda dapat menyiapkan pipeline pelatihan secara manual di konsol SageMaker AI, dalam praktiknya, kami menyarankan untuk meminimalkan titik kontak manual dalam penerapan pipeline pelatihan ML untuk memastikan bahwa model ML diterapkan secara konsisten dan berulang kali. Bergantung pada kebutuhan Anda dan masalah bisnis yang Anda hadapi, Anda dapat menentukan dan menerapkan strategi penerapan pada tiga tingkatan: semi-otomatis, sepenuhnya otomatis, dan dikelola sepenuhnya.

- **Semi-otomatis** — Secara default, langkah-langkah yang dibahas di bagian sebelumnya mengikuti pendekatan semi-otomatis, karena mereka menerapkan jalur pelatihan dan inferensi dengan menggunakan templat. CloudFormation ini membantu memastikan reproduktifitas pipa dan membantu Anda mengubah dan memperbaruinya dengan mudah.
- **Sepenuhnya otomatis** — Opsi yang lebih canggih adalah menggunakan integrasi berkelanjutan dan penerapan berkelanjutan (CI/CD) to the development, staging, and production environments. Incorporating CI/CD praktik untuk penyebaran pipa pelatihan dapat memastikan bahwa otomatisasi mencakup keterlacakan serta gerbang kualitas.
- **Dikelola sepenuhnya** — Pada akhirnya, Anda dapat mengembangkan sistem yang dikelola sepenuhnya sehingga Anda dapat menerapkan pipeline pelatihan ML dengan serangkaian manifes sederhana, dan sistem dapat mengkonfigurasi sendiri dan mengoordinasikan layanan yang diperlukan AWS .

Dalam panduan ini, kami memilih untuk menyajikan arsitektur beton. Namun, ada teknologi alternatif yang dapat Anda pertimbangkan. Dua bagian berikutnya membahas beberapa pilihan alternatif untuk platform dan mesin orkestrasi.

Platform yang berbeda untuk beban kerja ML

[Amazon SageMaker AI](#) adalah layanan AWS terkelola untuk pelatihan dan penyajian model ML. Banyak pengguna menghargai beragam fitur bawaannya dan banyak opsi yang ditawarkannya untuk menjalankan beban kerja ML. SageMaker AI sangat berguna jika Anda baru memulai mengimplementasikan ML di cloud. Fitur utama SageMaker AI meliputi:

- Keterlacakan bawaan (termasuk pelabelan, pelatihan, pelacakan model, pengoptimalan, dan inferensi).
- Opsi satu-klik bawaan untuk pelatihan dan inferensi dengan pengalaman Python dan HTML minimal.
- Penyetelan hyperparameter tingkat lanjut.
- Support untuk semua kerangka kerja kecerdasan buatan dan pembelajaran mesin (ML/AI) utama dan wadah Docker khusus.
- Kemampuan pemantauan bawaan.
- Pelacakan riwayat bawaan, termasuk pekerjaan pelatihan, pekerjaan pemrosesan, pekerjaan transformasi batch, model, titik akhir, dan kemampuan pencarian. Beberapa sejarah, seperti pelatihan, pemrosesan, dan transformasi batch, tidak dapat diubah dan hanya ditambahkan.

Salah satu alternatif untuk menggunakan SageMaker AI adalah [AWS Batch](#). AWS Batch memberikan tingkat kontrol yang lebih rendah atas komputasi dan orkestrasi untuk lingkungan Anda, tetapi tidak dibuat khusus untuk pembelajaran mesin. Beberapa fitur utamanya meliputi:

- Out-of-the-box penskalaan otomatis sumber daya komputasi berdasarkan beban kerja.
- Out-of-the-box dukungan untuk prioritas pekerjaan, percobaan ulang, dan ketergantungan pekerjaan.
- Pendekatan berbasis antrian yang mendukung pembangunan pekerjaan berulang dan sesuai permintaan.
- Support untuk beban kerja CPU dan GPU. Kemampuan untuk menggunakan GPU untuk membangun model ML sangat penting, karena GPU dapat mempercepat proses pelatihan secara signifikan, terutama untuk model pembelajaran mendalam.
- Kemampuan untuk menentukan [Amazon Machine Image](#) (AMI) khusus untuk lingkungan komputasi.

Mesin yang berbeda untuk orkestrasi pipa

Komponen utama kedua adalah lapisan orkestrasi pipa. AWS menyediakan [Step Functions](#) untuk pengalaman orkestrasi yang dikelola sepenuhnya. Alternatif populer untuk Step Functions adalah Apache Airflow. Saat membuat keputusan di antara keduanya, pertimbangkan hal berikut:

- **Infrastruktur yang diperlukan** - AWS Step Functions adalah layanan yang dikelola sepenuhnya dan tanpa server, sedangkan Airflow memerlukan pengelolaan infrastruktur Anda sendiri dan didasarkan pada perangkat lunak sumber terbuka. Akibatnya, Step Functions menyediakan ketersediaan tinggi di luar kotak, sedangkan administrasi Apache Airflow memerlukan langkah-langkah tambahan.
- **Kemampuan penjadwalan** - Kedua Step Functions dan Airflow menyediakan fungsionalitas yang sebanding.
- **Kemampuan visualisasi dan UI** - Kedua Step Functions dan Airflow menyediakan fungsionalitas yang sebanding.
- **Melewati variabel dalam grafik komputasi** — Step Functions menyediakan fungsionalitas terbatas untuk menggunakan AWS Lambda fungsi, sedangkan Airflow menyediakan antarmuka XCom
- **Penggunaan** - Step Functions sangat populer di kalangan AWS pelanggan, dan Airflow telah diadopsi secara luas oleh komunitas rekayasa data.

Kesimpulan

Sebagai transisi pembelajaran mesin dari disiplin penelitian ke bidang terapan, kami telah melihat pertumbuhan tahunan sebesar 25 persen dalam pengembangan, penyebaran, dan operasi pipa ML di berbagai industri. Nilai bisnis dari ML diwujudkan melalui day-to-day operasi dan jaringan pipa, yang, pada gilirannya, mendorong penelitian dan pengembangan model dan algoritma ML. Meskipun demikian, penerapan ML dalam produksi menghadirkan banyak tantangan, karena menjalin aktivitas dan artefak yang berbeda secara signifikan, seperti manajemen data, pemrosesan, analisis, pemodelan, verifikasi, dan keamanan. Melalui berbagai keterlibatan AI/ML dengan AWS pelanggan, tim Ilmu Data kami telah mengamati bahwa tantangan utama adalah kurangnya end-to-end alur kerja yang akan menyediakan serangkaian templat untuk menggabungkan atau memisahkan aktivitas dan artefak yang berbeda secara optimal. DevOps Dalam panduan ini, kami mempresentasikan [alur kerja ML Max](#) untuk mengatasi masalah mendesak ini. ML Max menyediakan step-by-step pedoman dan satu set template pemrograman. Tujuannya adalah untuk memungkinkan transisi yang cepat dan hemat biaya dari fase pengembangan model interaktif ke konfigurasi pipa ML yang lengkap dan dapat diskalakan yang siap untuk produksi.

Sumber daya tambahan

Panduan dan pola terkait

- [Mengukur ketidakpastian dalam sistem pembelajaran mendalam](#)
- [Migrasikan beban kerja Build, Train, dan Deploy HTML ke Amazon SageMaker AI menggunakan Alat Pengembang AWS](#)
- [AWS Situs web Prescriptive Guidance](#)

AWS layanan

- [AWS Batch](#)
- [AWS CloudFormation](#)
- [IAM](#)
- [Amazon SageMaker AI](#)
- [AWS Step Functions](#)

AWS sumber daya umum

- [AWS dokumentasi](#)
- [AWS referensi umum](#)
- [AWS glosarium](#)

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	—	29 Januari 2021

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