Amazon SageMaker: Developer Guide

Contents

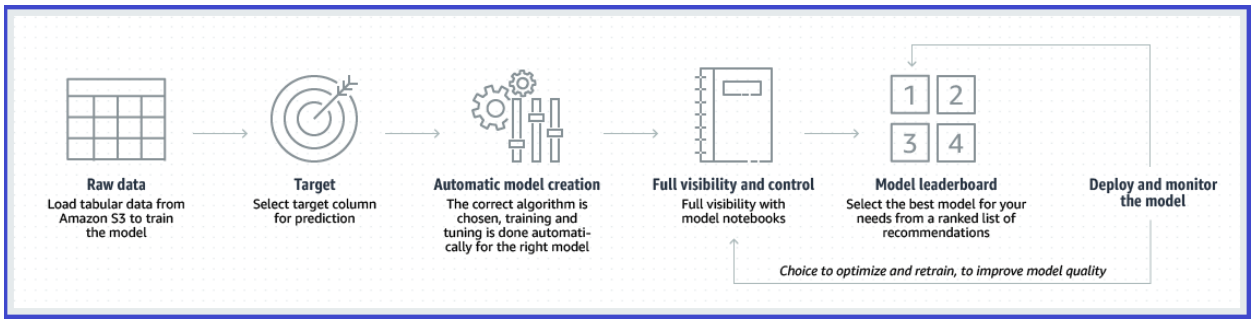
Use automated ML, no-code, or low-code

AutoML

Tasks performed by AutoML process

Autopilot performs the following key tasks that you can use on autopilot or with various degrees of human guidance:

- **Data analysis and preprocessing:** handles missing values, normalizes your data, selects features, and overall prepares the data for model training.
- **Model selection:** Explores a variety of algorithms and uses a cross-validation resampling technique to generate metrics that evaluate the predictive quality of the algorithms based on predefined objective metrics.
- **Hyperparameter optimization: A**utomates the search for optimal hyperparameter configurations.
- **Model training and evaluation:** Autopilot automates the process of training and evaluating various model candidates.
    - It splits the data into training and validation sets, trains the selected model candidates using the training data, and evaluates their performance on the unseen validation set.
    - Ranks the optimized model candidates based on their performance and identifies the best performing model.
- **Model deployment:** Once Autopilot has identified the best performing model, it provides the option to deploy the model automatically by generating the model artifacts and the endpoint exposing an API. External applications can send data to the endpoint and receive the corresponding predictions or inferences.



AutoML supported the following problem types

- **Regression, binary, and multiclass classification** (with tabular data as CSV or Parquet)
- **Text classification** with data formatted as CSV or Parquet
- **Time-series forecasting** with time-series data formatted as CSV or Parquet
- **Fine-tuning of large language models (LLMs)** for **text generation** in CSV or Parquet
- **Image classification** with image formats such as PNG, JPEG

Create a regression or classification job for tabular data using the AutoML API

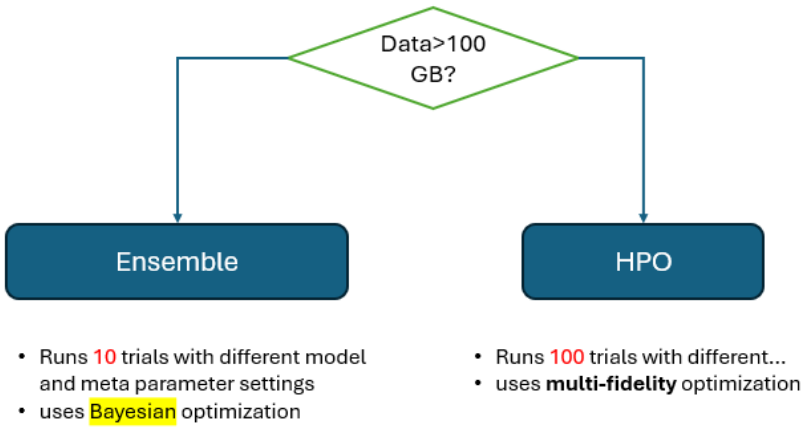
You can create an Autopilot experiment for tabular data programmatically by calling theCreateAutoMLJobV2 API action

Required parameters

- An **AutoMLJobName** to specify the name of your job.
- At least one AutoMLJobChannel in **AutoMLJobInputDataConfig** to specify your data source.
- Both an **AutoMLJobObjective** metric and your chosen type of supervised learning problem (binary classification, multiclass classification, regression) in **AutoMLProblemTypeConfig**, or none at all. For tabular data, you must choose TabularJobConfig as the type ofAutoMLProblemTypeConfig. You set the supervised learning problem in the ProblemTypeattribute of TabularJobConfig.
- An **OutputDataConfig** to specify the Amazon S3 output path to store the artifacts of your AutoML job.
- A **RoleArn** to specify the ARN of the role used to access your data.

Training modes and algorithm support

Training modes



Ensemble mode algorithms

Algorithm	Desc	Optimized for
LenarLearner	Supervised learning algorithm	either classification or regression
XGBoost	Supervised learning algorithm	combining an ensemble of estimates
Deep learning algorithm	Multilayer perceptron (MLP) and feedforward artificial neural network	Neural Network

Ensemble mode algorithms

Algorithm	Desc	Optimized for
LightGBM	Decision trees that grow in breadth, rather than depth	Speed (Light –> Speed)
XGBoost	gradient boosting that grows in depth, rather than breadth	
CatBoost	uses tree-based algorithms with gradient boosting.	handling **categorical** variables
Random Forest	several decision trees , trees are split into optimal nodes at each level	Improved predictions
Extra Trees	Similar to Random Forest, but trees split at random	- Same as above -
Linear Models	uses a linear equation to model the relationship between two variables	
Neural network torch	implemented using Pytorch	Neural Network
Neural network fast.ai	implemented using fast.ai.	Neural Network

Metrics

HPO Mode	Ensemble Mode
K-fold cross-validation	Most metrics (Lienar/Regression ones)

Model performance report Metrics

Metric Name	Value	Standard Deviation
weighted_recall	0.597104	0.005410
weighted_precision	0.591693	0.005729
accuracy	0.597104	0.005410
weighted_f0_5	0.592155	0.005659
weighted_f1	0.593423	0.005554
weighted_f2	0.595392	0.005456
accuracy_best_constant_classifier	0.200699	0.004422
weighted_recall_best_constant_classifier	0.200699	0.004422
weighted_precision_best_constant_classifier	0.040280	0.001753
weighted_f0_5_best_constant_classifier	0.047944	0.002039
weighted_f1_best_constant_classifier	0.067094	0.002684
weighted_f2_best_constant_classifier	0.111716	0.003808

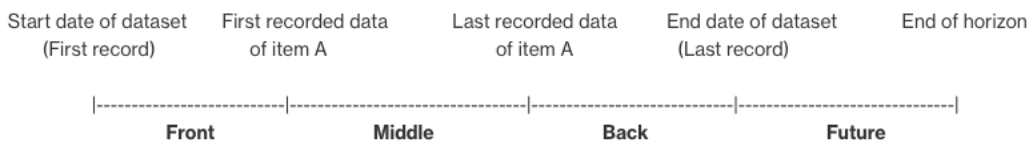
Time Series Models

Handle missing values

Autopilot supports the following filling methods:

- **Front filling:** Fills any missing values between the earliest recorded data point among all items and the starting point of each item (each item can start at a different time). This ensures that the data for each item is complete and spans from the earliest recorded data point to its respective starting point.
- **Middle filling:** Fills any missing values between the start and end dates of the items in the dataset.
- **Back filling:** Fills any missing values between the last data point of each item (each item can stop at a different time) and the last recorded data point among all items.
- **Future filling:** Fills any missing values between the last recorded data point among all items and the end of the forecast horizon.

The following image provides a visual representation of the different filling methods.



Optimize the learning process of your text generation models with hyperparameters

You can optimize the learning process of your base model by adjusting any combination of the following hyperparameters. These parameters are available for all models.

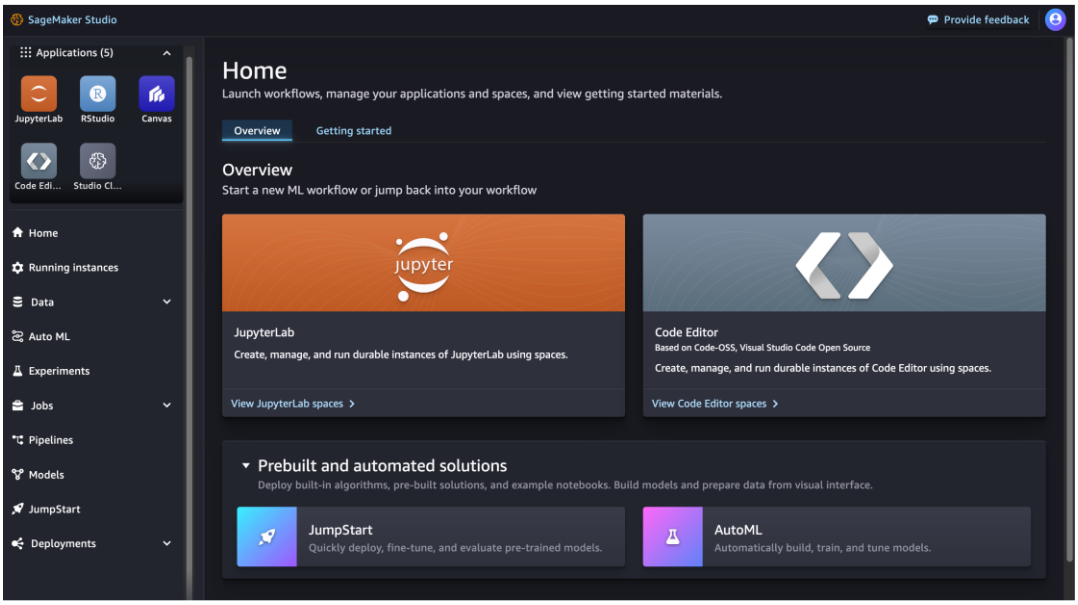
- **Epoch Count:** Determines how many times the model goes through the entire training dataset. It influences the training duration and can prevent overfitting when set appropriately.
- **Batch Size**: Defines the number of data samples used in each iteration of training. We recommend starting with a batch size of 1, then incrementally increase it until an out of memory error occurs. As a reference, 10 epochs typically takes up to 72h to complete.
- **Learning Rate:** Controls the step size at which a model's parameters are updated (how quickly or slowly the model's parameters are updated) during training. It determines during training.
- **Learning Rate Warmup Steps:** Specifies the number of training steps during which the learning rate gradually increases before reaching its target or maximum value. This helps the model converge more effectively and avoid issues like divergence or slow convergence that can occur with an initially high learning rate.

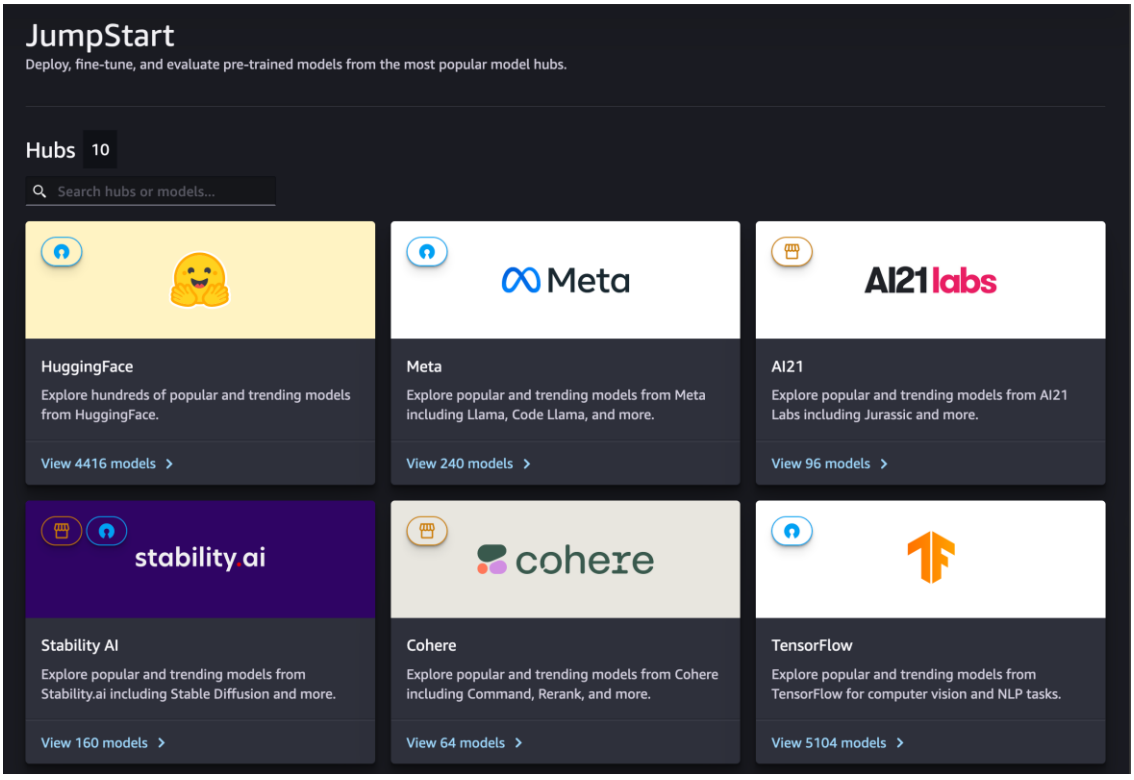
Metrics for fine-tuning large language models in Autopilot

After fine-tuning an LLM you can evaluate the quality of its generated text using a range of ROUGE scores. Additionally, you can analyze the perplexity and cross-entropy training and validation losses as part of the evaluation process.

- **Perplexity loss** measures how well the model can predict the next word in a sequence of text, with lower values indicating a better understanding of the language and context.
- **Recall-Oriented Understudy for Gisting Evaluation (ROUGE)** is a set of metrics used in the NLP and machine learning to evaluate the quality of machine-generated text, such as text summarization or text generation. It primarily assesses the similarities between the generated text and the ground truth reference (human-written) text of a validation dataset.
- The following list contains the name and description of the ROUGE metrics available after the fine-tuning of large language models in Autopilot
    - ROUGE-N **(ROUGE-1, ROUGE-2):** Measures the overlap of n-grams between the system-generated and reference texts..
    - **ROUGE-L:** (ROUGE-Longest Common Subsequence) calculates the longest common subsequence between the system-generated text and the reference text. This variant considers word order in addition to content overlap.

Train, deploy, and evaluate pretrained models with SageMaker JumpStart





Curate pretrained JumpStart foundation models for organization with private hubs.

Curate pretrained JumpStart foundation models for your organization with private hubs.

Use private model hubs to share, within your organization, models and notebooks, centralize model artifacts, improve model discoverability, and streamline model use.

Task-Specific Models with JumpStart

JumpStart supports task-specific models across fifteen of the most popular problem types

Supported frameworks	Problem types
PyTorch	• Image classification • Question answering
TensorFlow	• Text classification • Text Embedding • Sentence pair classification • Image embedding • Image classification
MXNet	• Semantic segmentation • Instance segmentation • Text Embedding
All 3 above	Object detection
Hugging Face	• Sentence pair classification • Question answering • NER • Text Summarization • Text Generation

Use ML environments offered by SageMaker

Refer to the following FAQ topics for answers to commonly asked questions about Amazon SageMaker Model Dashboard.

SageMaker Environments

- ***Amazon SageMaker Studio*** *(Recommended): The latest web-based experience for running ML workflows with a suite of IDEs. Studio supports the following applications:*
    - Amazon SageMaker Studio Classic
    - Code Editor, based on Code-OSS, Visual Studio Code - Open Source
    - JupyterLab
    - Amazon SageMaker Canvas
    - RStudio
- ***Amazon SageMaker Studio Classic****: Lets you build, train, debug, deploy, and monitor your machine learning models.*
- ***Amazon SageMaker Notebook Instances****:* Lets you prepare and process data, and train and deploy machine learning models from a compute instance running the Jupyter Notebook application.
- ***Amazon SageMaker Studio Lab****:* Studio Lab is a free service that gives you access to AWS compute resources, in an environment based on open-source JupyterLab, without requiring an AWS account.
- ***Amazon SageMaker Canvas****: Gives you the ability to use machine learning to generate predictions without needing to code.*
- ***Amazon SageMaker geospatial****: Gives you the ability to build, train, and deploy geospatial models.*
- ***RStudio on Amazon SageMaker****: RStudio is an IDE for R, with a console, syntax-highlighting editor that supports direct code execution, and tools for plotting, history, debugging and workspace management.*
- ***SageMaker HyperPod****: SageMaker HyperPod lets you provision resilient clusters for running machine learning (ML) workloads and developing state-of-the-art models such as large language models (LLMs), diffusion models, and foundation models (FMs).*

SageMaker Studio

Features

- **Amazon EFS auto-mounting in Studio**

  Amazon SageMaker supports automatically mounting a folder in an Amazon EFS volume for each user in a domain. Using this folder, users can share data between their own private spaces. However, users cannot share data with other users in the domain. Users only have access to their own folder.

  You can opt-out of Amazon SageMaker auto-mounting Amazon EFS

- **Idle shutdown**

  Amazon SageMaker supports shutting down idle resources to manage costs and prevent cost overruns due to cost accrued by idle, billable resources. It accomplishes this by detecting an app's idle state and performing an app shutdown when idle criteria are met.

- **Amazon SageMaker Studio spaces**

  Spaces are used to manage the storage and resource needs of some Amazon SageMaker Studio applications. Each space has a 1:1 relationship with an instance of an application. Every supported application that is created gets its own space.

  Spaces can be either private or shared:

  - Private: Private spaces are scoped to a single user in a domain. Private spaces cannot be shared with other users. All applications that support spaces also support private spaces.
  - Shared: Shared spaces are accessible by all users in the domain. For more information about shared spaces, see Collaborate with shared spaces.

- **Use NVMe stores with Amazon SageMaker Studio**

  Amazon SageMaker Studio applications and their associated notebooks run on EC2 instances. Some of the Amazon EC2 instance types, such as the ml.m5d instance family, offer non-volatile memory express (NVMe) solid state drives (SSD) instance stores.

  NVMe instance stores are local ephemeral disk stores that are physically connected to an instance for fast temporary storage. Studio applications support NVMe instance stores for supported instance types.

- **Local mode support in Amazon SageMaker Studio**

  Amazon SageMaker Studio applications support the use of local mode to create estimators, processors, and pipelines, then deploy them to a local environment. With local mode, you can test machine learning scripts before running them in SageMaker managed training or hosting environments.

- **Amazon SageMaker Studio pricing:** There is no additional charge for using the Amazon SageMaker Studio UI.

SageMaker Studio – Jupyter Lab

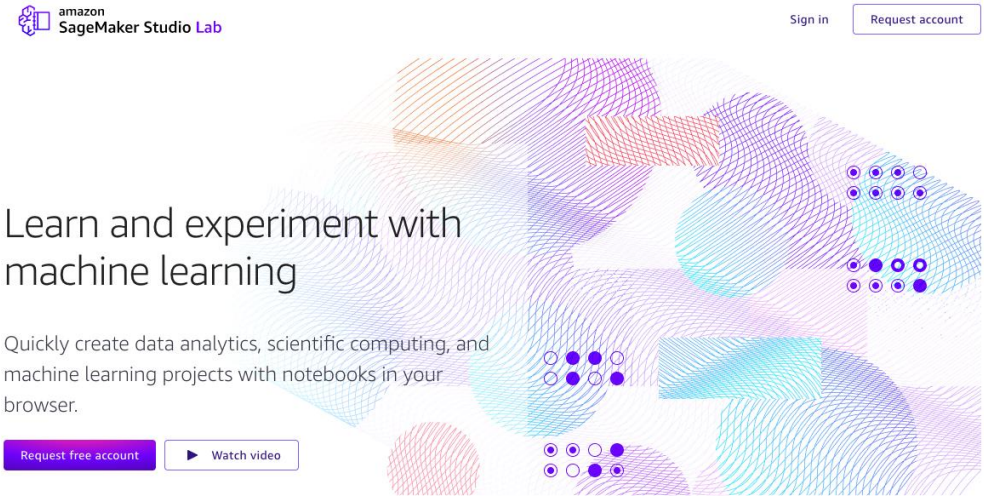
A JupyterLab space is a private or shared space within Studio that manages the storage and compute resources needed to run the JupyterLab application

Amazon SageMaker Notebook Instances

An Amazon SageMaker notebook instance is a machine learning (ML) compute instance running the Jupyter Notebook application. SageMaker creates the instance and related resources. Use Jupyter notebooks in your notebook instance to:

- prepare and process data
- write code to train models
- deploy models to SageMaker hosting
- test or validate your models

SageMaker Studio Lab



Use generative AI in SageMaker notebook environments

Features

Jupyter AI is an open-source extension of JupyterLab integrating generative AI capabilities into Jupyter notebooks. Through the Jupyter AI chat interface and magic commands, users experiment with code generated from natural language instructions, explain existing code, ask questions about their local files, generate entire notebooks, and more.

The extension connects Jupyter notebooks with large language models (LLMs) that users can use to generate text, code, or images, and to ask questions about their own data.

Jupyter AI supports generative model providers such as AI21, Anthropic, AWS (JumpStart and Amazon Bedrock), Cohere, and OpenAI.

Label Data with Human in the loop

Labeling training data using humans via SageMaker Ground Truth

Use the workforce of your choice to label your dataset. You can choose your workforce from:

- The Amazon Mechanical Turk workforce of over 500,000 independent contractors worldwide.
- A private workforce that you create from your employees or contractors for handling data within your organization.
- A vendor company that you can find in the AWS Marketplace that specializes in data labeling services.

Label Images

Options

- Bounding Box
- Image Semantic Segmentation
- Auto-Segmentation Tool
- Image Classification (Single Label)
- Image Classification (Multi-label)
- Image Label Verification

Label Text

Options

- Named Entity Recognition
- Text Classification (Single Label)
- Text Classification (Multi-label)

Label Videos and Video Frames

Labeling training data using humans via SageMaker Ground Truth Plus

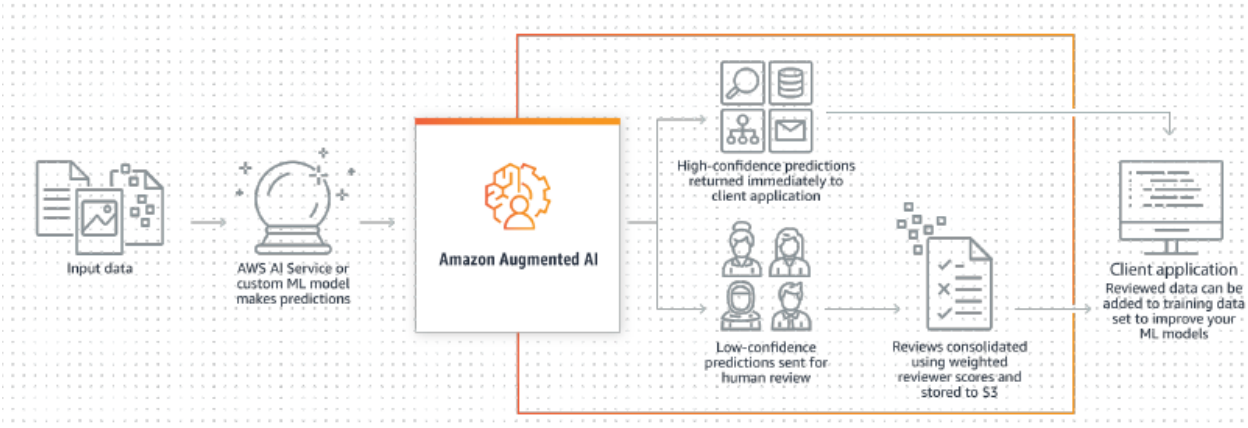
SageMaker Ground Truth Plus does not support PHI, PCI or FedRAMP certified data, and you should not provide this data to SageMaker Ground Truth Plus

Process:

- **Request a Project**
- **Create a Project Team**
- **Create a Batch**
- **Accept or Reject Batches**: If you reject a batch, you can provide feedback and explain your reasons for rejecting the batch. SageMaker Ground Truth Plus allows you to provide feedback at the data object level as well as the batch level.

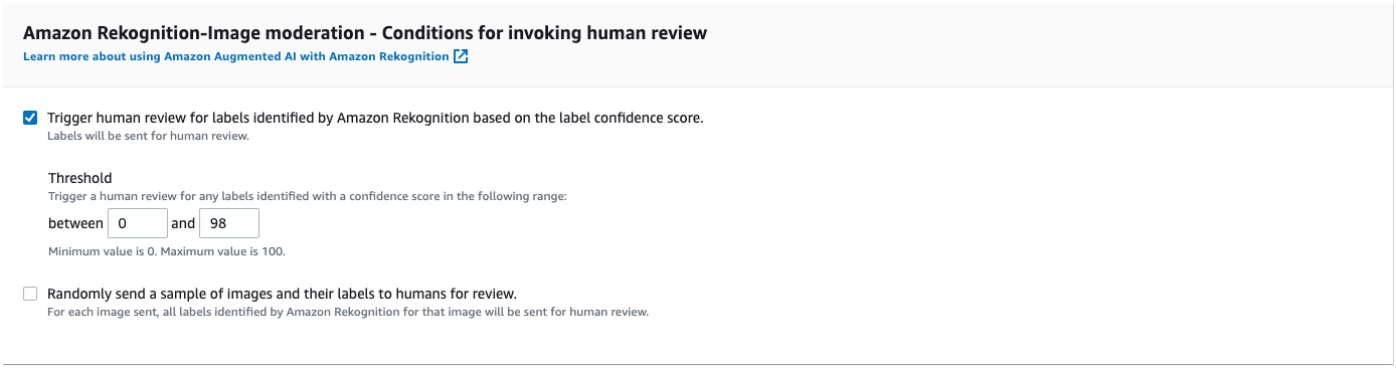
Using Amazon Augmented AI for Human Review

You can use Amazon Augmented AI to get human review of low-confidence predictions or random prediction samples



How to trigger Human review

The following image shows how you can select the Trigger human review for labels identified by Rekognition based on label confidence score option and enter a Threshold between 0 and 98 in the Amazon A2I console



Using the right Data Prep tool in SageMaker

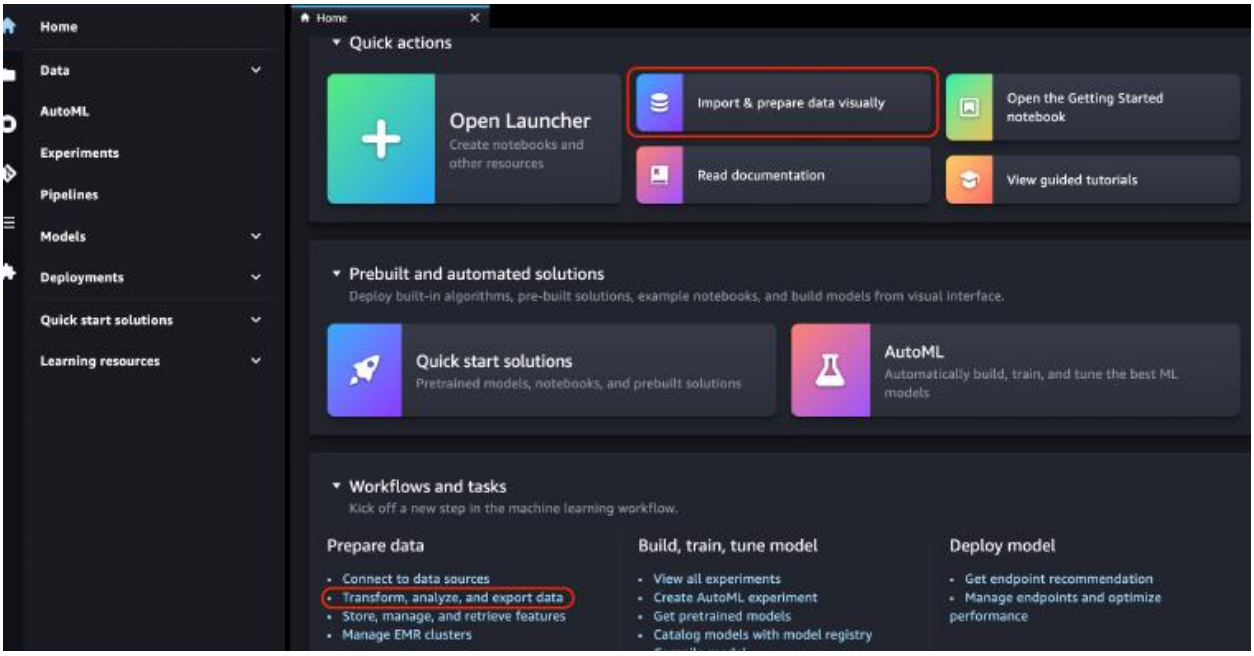
3 primary use cases for data preparation with Amazon SageMaker

	Use case 1	Use case 2	Use case 3
SageMaker feature	Data Wrangler + Canvas	SQL in Studio	**EMR** Serverless applications in Studio
Description	Canvas is a visual low-code environment for building, training, and deploying machine learning models in SageMaker. Its integrate d Data Wrangler tool allows users to combine, transform, and clean datasets through point-and-click interactions.	The SQL extension in Studio allows users to connect below to author **ad-hoc SQL queries**, and preview results in JupyterLab notebooks. • Redshift • Snowflake • Athena • S3	The integration between EMR Serverless and SageMaker Studio provides a scalable serverless environment for large-scale data preparation for machine learning using open-source frameworks such as Apache Spark and Apache Hive
Optimized for	Optimized for tabular data tasks such as handling missing values, encoding categorical variables, and applying data transform ations.	For users whose data resides in Redshift, Snowflake, Athena, or Amazon S3 and want to combine exploratory SQL without the need to learn Spark.	For users who prefer a serverless experience for scaling short-running or intermittent interactive workloads revolving around Apache Spark.
Considerations	It may not be the optimal choice if • your team already has expertise in Python, Spark, etc. • you need full flexibility to customize transformations to add complex business logic or require full control over your data processing environment.	Only for structured data residing in Redshift, Snowflake, Athena, or S3 only.	• Steep learning curve. • For long-running, computationally intensive, large-scale data processing tasks, consider using Amazon EMR clusters
Additional options			

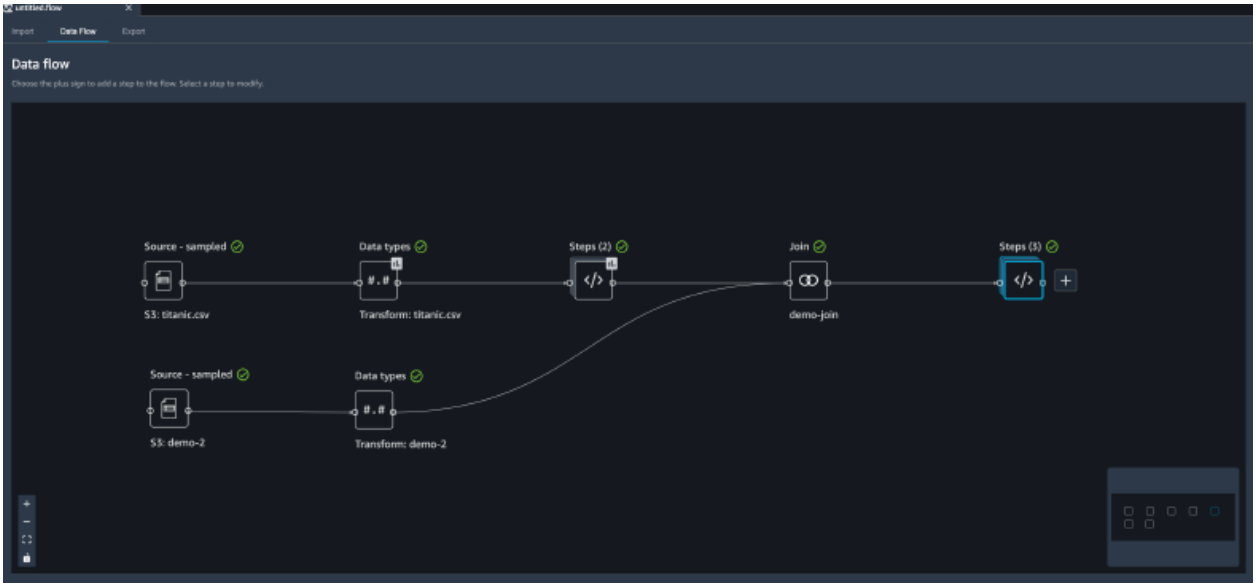
- **Using Amazon EMR clusters**: For long-running, computationally intensive, large-scale data processing tasks, consider using Amazon EMR clusters from SageMaker Studio. Amazon EMR clusters are designed to handle massive parallelization and can scale to hundreds or thousands of nodes, making them well-suited for big data workloads that require frameworks like Apache Spark, Hadoop, Hive, and Presto. The integration of Amazon EMR with SageMaker Studio allows you to leverage the scalability and performance of Amazon EMR, while keeping your full ML experimentation, model training and deployment, centralized and managed within the SageMaker Studio environment.
- Using **Glue** interactive sessions: You can use the Apache Spark-based serverless engine from AWS Glue interactive sessions to aggregate, transform, and prepare data from multiple sources in SageMaker Studio.
- Identify **bias in training data using SageMaker Clarify**
- **Feature Store** for Create, store, and share features
- **Human-in-the-loop**:
- **SageMaker Model Building Pipeline**: After performing exploratory data analysis and creating your data transformations steps, you can productionize your transformation code using SageMaker Processing jobs and automate your preparation workflow using SageMaker Model Building Pipelines.

Prepare Data with Data Wrangler

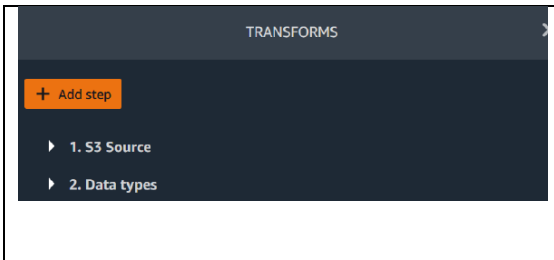
UI

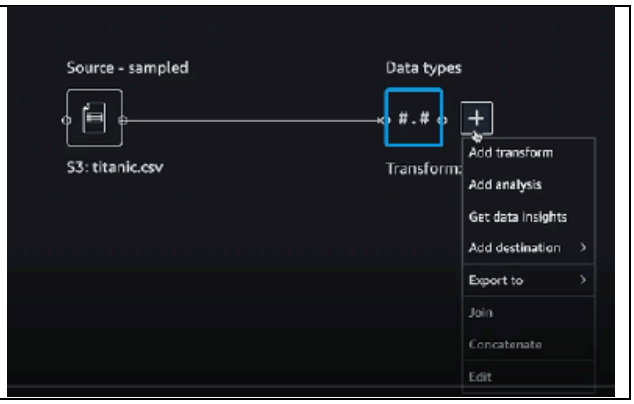


1. Create Workflow using Data Wrangler

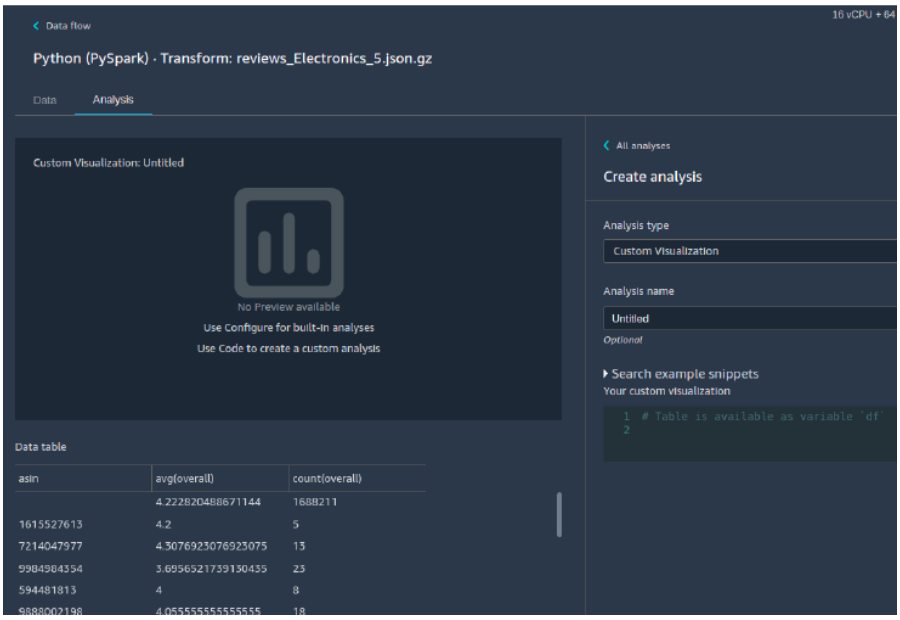


2. Transform Data





3. Analyze and Visualize



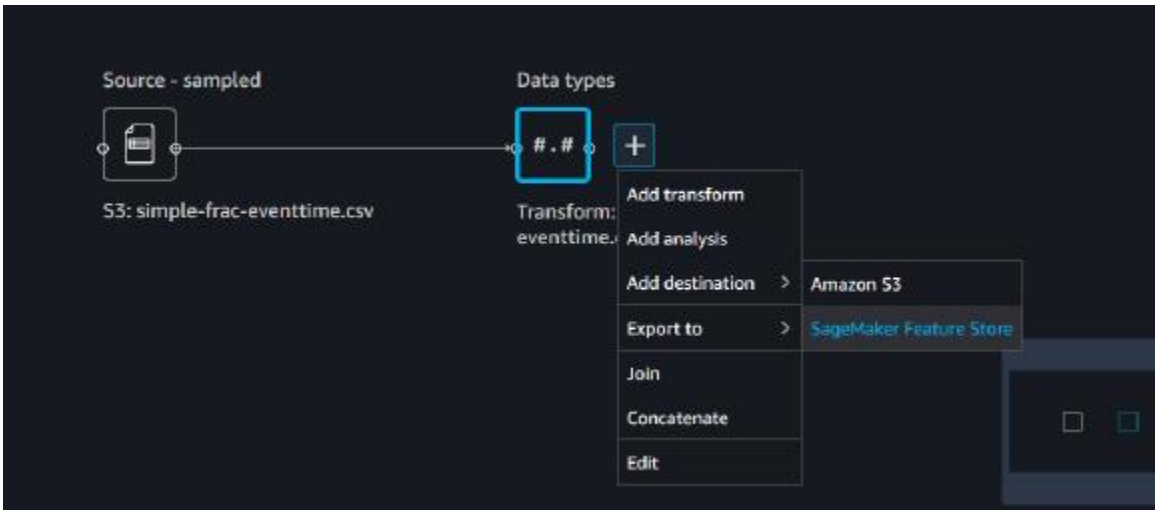
4. Export

You can export your data transformations to the following:

- Amazon S3
- Pipelines
- Amazon SageMaker Feature Store
- Python Code

Export using the UI

1. Choose the **+** symbol next to the node containing the dataset that you'd like to export.
2. Under **Add destination**, choose **SageMaker Feature Store**, S3, etc.



Create, store and share Features with Feature Store

Feature Store Basics

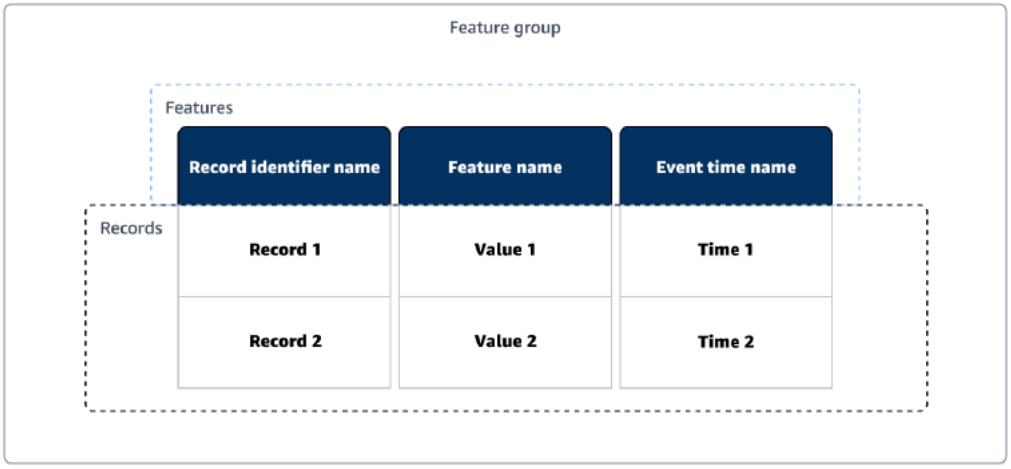
Feature Store Modes

Feature Store in the following modes:

- **Online** – In online mode, features are read with low latency (milliseconds) reads and used for high throughput predictions. This mode requires a feature group to be stored in an online store.
- **Offline** – In offline mode, large streams of data are fed to an offline store, which can be used for training and batch inference. This mode requires a feature group to be stored in an offline store. The offline store uses your S3 bucket for storage and can also fetch data using Athena queries.
- **Online and Offline** – This includes both online and offline modes.

Create feature groups

To ingest features into Feature Store, you must first define the feature group and the feature definitions (feature name and data type) for all features that belong to the feature group.

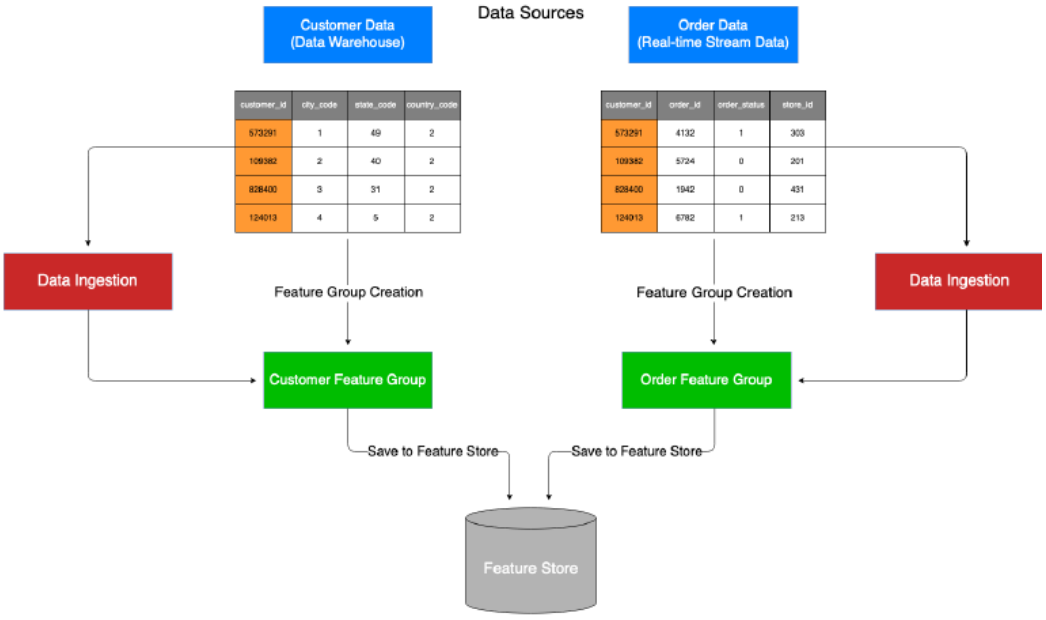


Creating a Feature Store

Step 1: Set up your SageMaker session

Step 2: Inspect your data

The following diagram illustrates the steps that data goes through before Feature Store ingests it. In this notebook, we illustrate the use case where you have data from multiple sources and want to store them independently in a Feature Store. Our example considers data from a data warehouse (customer data), and data from a real-time streaming service (order data).



Step 3: Create feature groups

Step 4: Ingest data into a feature group

Data sources and ingestion

- **Stream ingestion:** You can use streaming sources such as Kafka or Kinesis as a data source. Records can be ingested into your feature group by using the synchronous PutRecordAPI call.
- **Data Wrangler with Feature Store:** Data Wrangler is a feature of Studio Classic that provides an end-to-end solution to import, prepare, transform, featurize, and analyze data. Data Wrangler enables you to engineer your features and ingest them into your online or offline store feature groups.
- **Batch ingestion with Feature Store Spark:** SageMaker Feature Store Spark is a Spark connector that connects the Spark library to Feature Store. Feature Store supports batch data ingestion with Spark, using your existing ETL pipeline, on Amazon EMR, AWS Glue job, an SageMaker Processing job, or a SageMaker notebook

Using a Feature Store

(OPTIONAL) Step 1: Sharing a Feature Store

With AWS Resource Access Manager (AWS RAM) you can securely share the feature group catalog, which contains all of your feature group and feature resources, with other AWS accounts

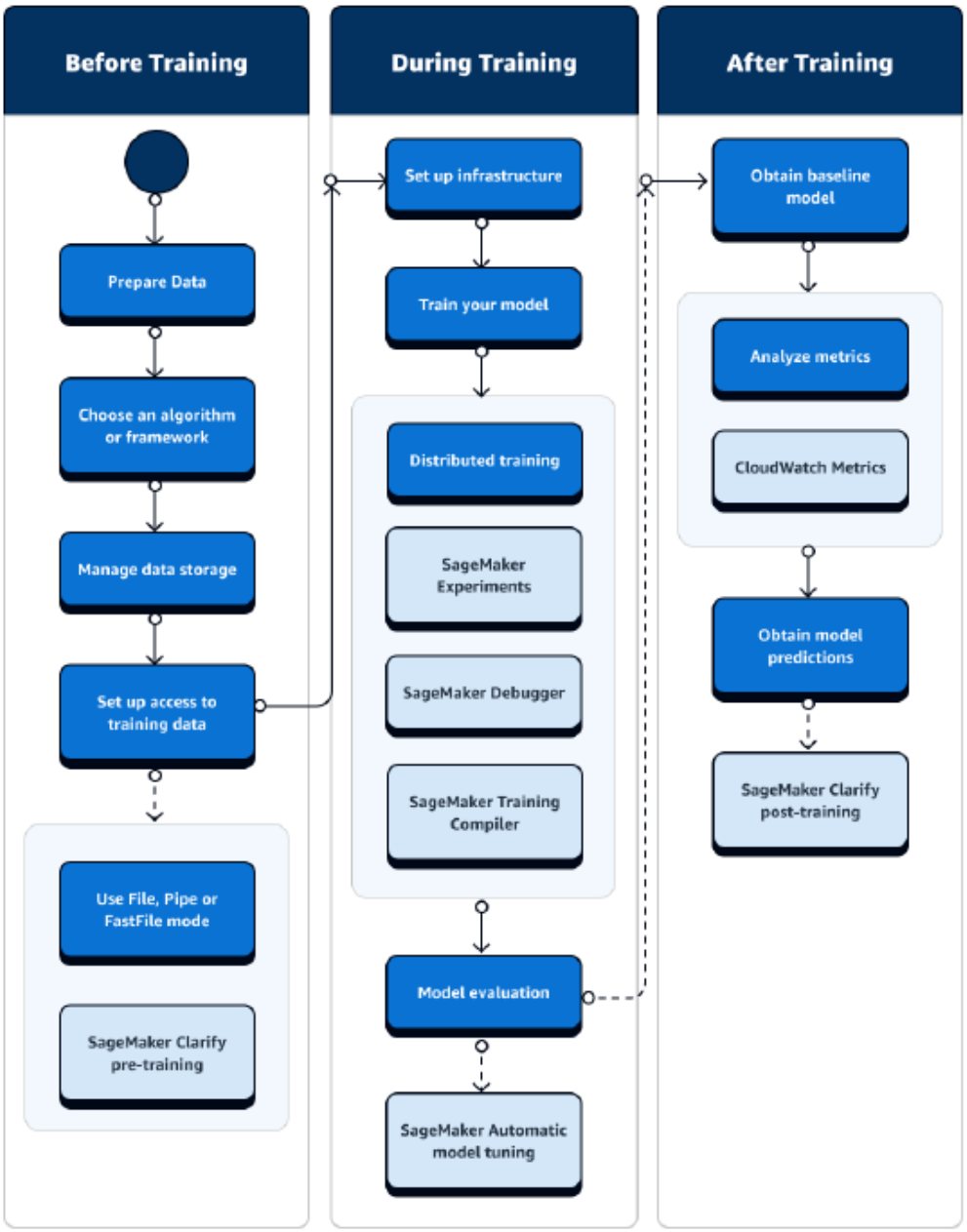
Step 2. Create a dataset from your feature groups

After you've created feature groups for your offline store and populated them with data, you can create a dataset by running queries or using the SDK to join data stored in the offline store from different feature groups. After a Feature Store feature group has been created in an offline store, you can choose to use the following methods to get your data:

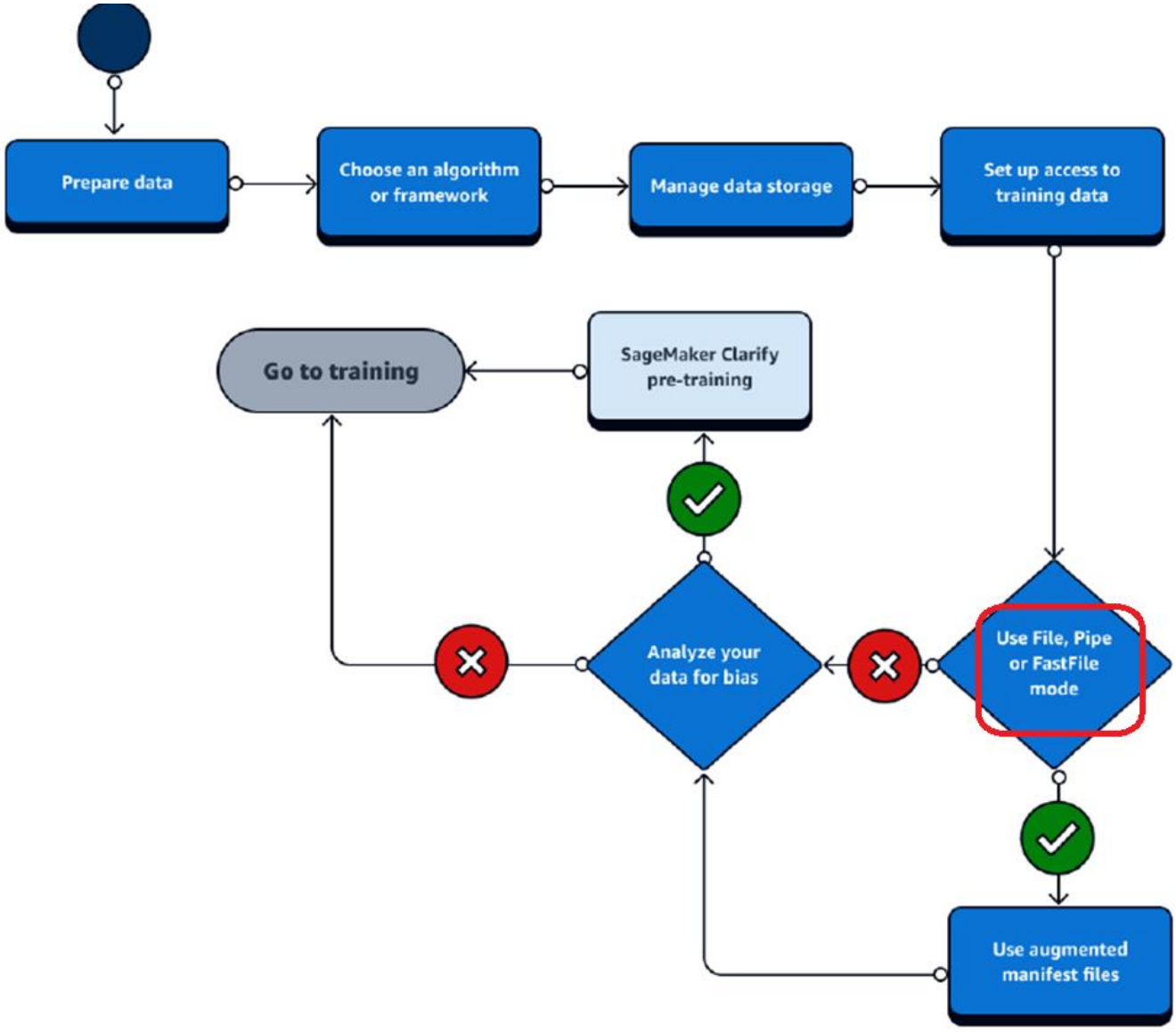
- Using the Amazon SageMaker Python SDK
- Running SQL queries in the Amazon Athena

Train ML Models

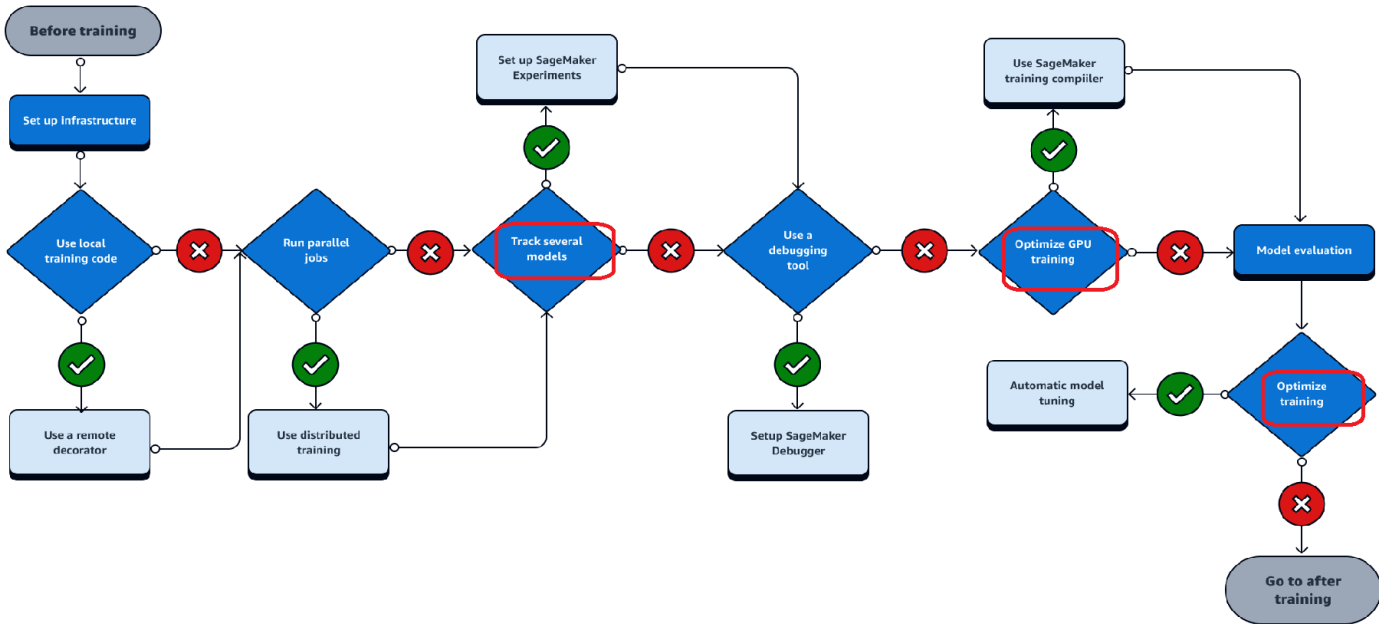
SageMaker Training workflow



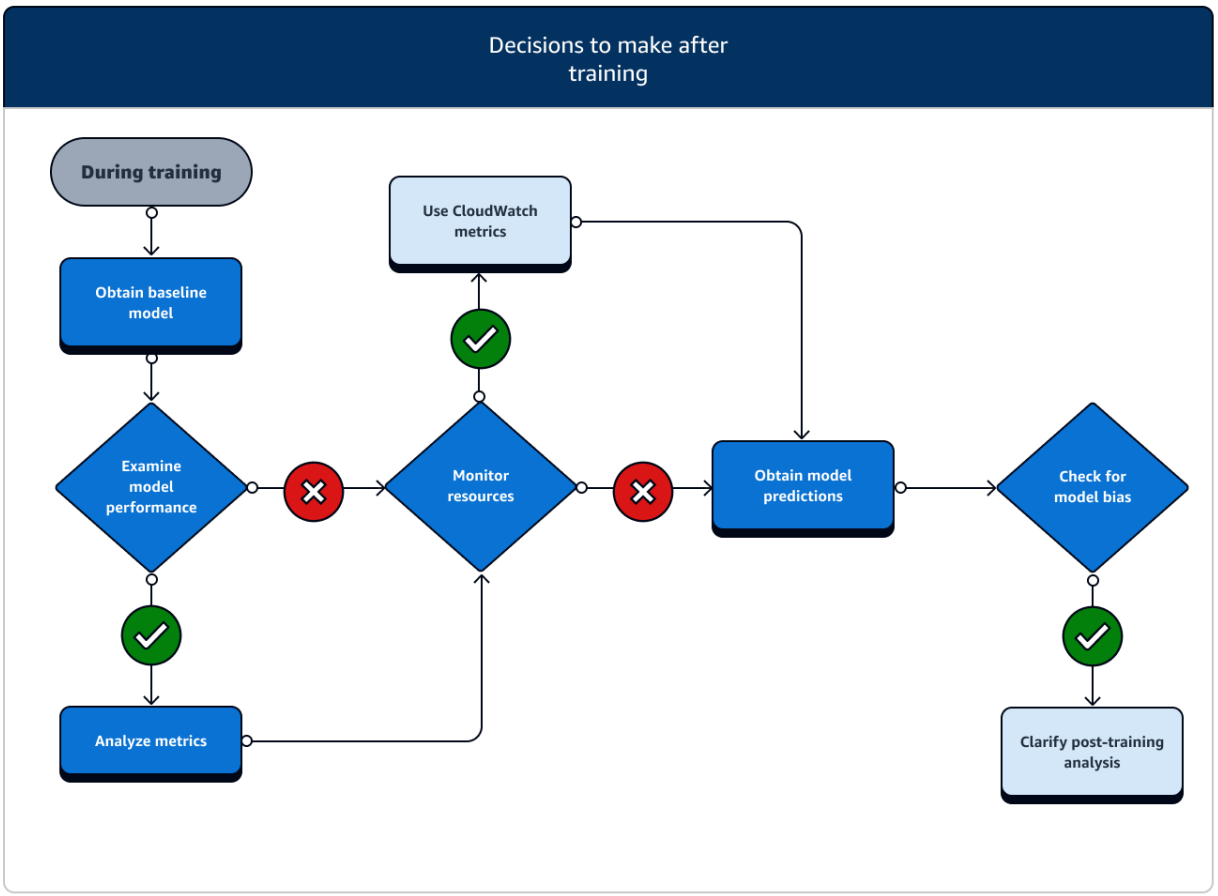
1. Before training



2. During training



3. After training



SageMaker Training features

- **Choose an algorithm**
- **Manage machine learning experiments using Amazon SageMaker with MLflow**

  Amazon SageMaker with MLflow is a capability of Amazon SageMaker that lets you create, manage, analyze, and compare your machine learning experiments

  Use MLflow while training and evaluating models to:

  - find the best candidates for your use case
  - compare model performance, parameters, and metrics across experiments in the MLflow UI
  - keep track of your best models in the MLflow Model Registry
  - automatically register them as a SageMaker model
  - deploy registered models to SageMaker endpoints.

- **Perform automatic model tuning (AMT) with SageMaker**

  Amazon SageMaker automatic model tuning (AMT) finds the best version of a model by running many training jobs on your dataset. Amazon SageMaker automatic model tuning (AMT) is also known as hyperparameter tuning. To do this, AMT uses the algorithm and ranges of hyperparameters that you specify. It then chooses the hyperparameter values that creates a model that performs the best, as measured by a metric that you choose.

- **Refine data during training with Amazon SageMaker smart sifting**

  SageMaker smart sifting operates during training as data is loaded. The SageMaker smart sifting algorithm runs loss calculation over the batches, and sifts non-improving data out before the forward and backward pass of each iteration. The refined data batch is then used for the forward and backward pass. That helps improve the efficiency of your training datasets and reduce total training time and cost.

- **Debug and improve model performance**

  Amazon SageMaker provides two debugging tools to help identify such convergence issues and gain visibility into your models.

  - **Amazon SageMaker with TensorBoard**

  To offer a greater compatibility with the open-source community tools within the SageMaker Training platform, SageMaker hosts TensorBoard as an application in SageMaker domain. You can bring your training jobs to SageMaker and keep using the TensorBoard summary writer to collect the model output tensors.

  - **Amazon SageMaker Debugger**

  Amazon SageMaker Debugger is a capability of SageMaker that provides tools to register hooks to callbacks to extract model output tensors and save them in S3. It provides built-in rules for detecting model convergence issues, such as

- Overfitting
- saturated activation functions
- vanishing gradients, and more.

You can also set up the built-in rules with Amazon CloudWatch Events and AWS Lambda for taking automated actions against detected issues.

- **Profile and optimize computational performance w/ SageMaker Profiler**

Amazon SageMaker Profiler is a profiling capability of SageMaker with which you can deep dive into compute resources provisioned while training deep learning models, and gain visibility into operation-level details. SageMaker Profiler provides Python modules for adding annotations throughout PyTorch or TensorFlow training scripts and activating SageMaker Profiler.

- **Distributed training in Amazon SageMaker**
  - The SMDDP library improves communication between nodes with implementations of AllReduceand AllGather collective communication operations that are optimized for AWS network infrastructure and Amazon SageMaker ML instance topology. You can use the SMDDP library as the backend of PyTorch-based distributed training packages: PyTorch distributed data parallel (DDP), PyTorch fully sharded data parallelism (FSDP), DeepSpeed, and Megatron-DeepSpeed .
  - **Use the SageMaker model parallelism library (SMP)**

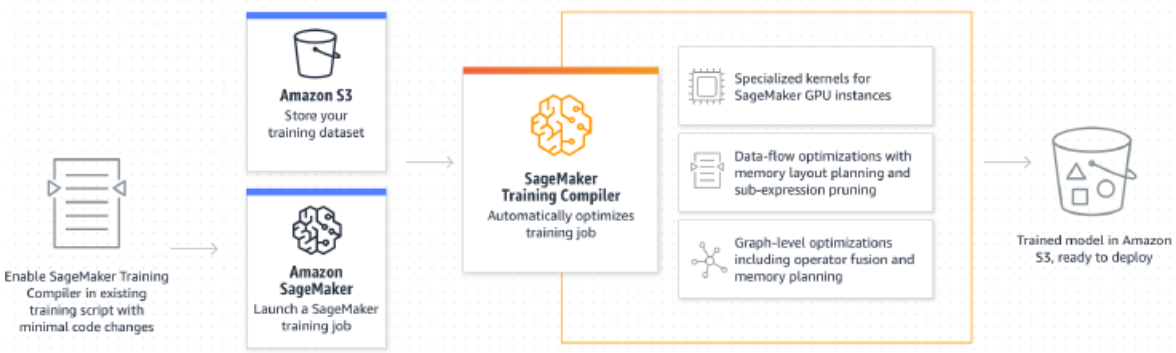
SageMaker provides the SMP library and supports various distributed training techniques, such as sharded data parallelism, pipelining, tensor parallelism, optimizer state sharding, and more

- **Use open source distributed training frameworks**

SageMaker also supports PyTorch DistributedDataParallel (DDP)

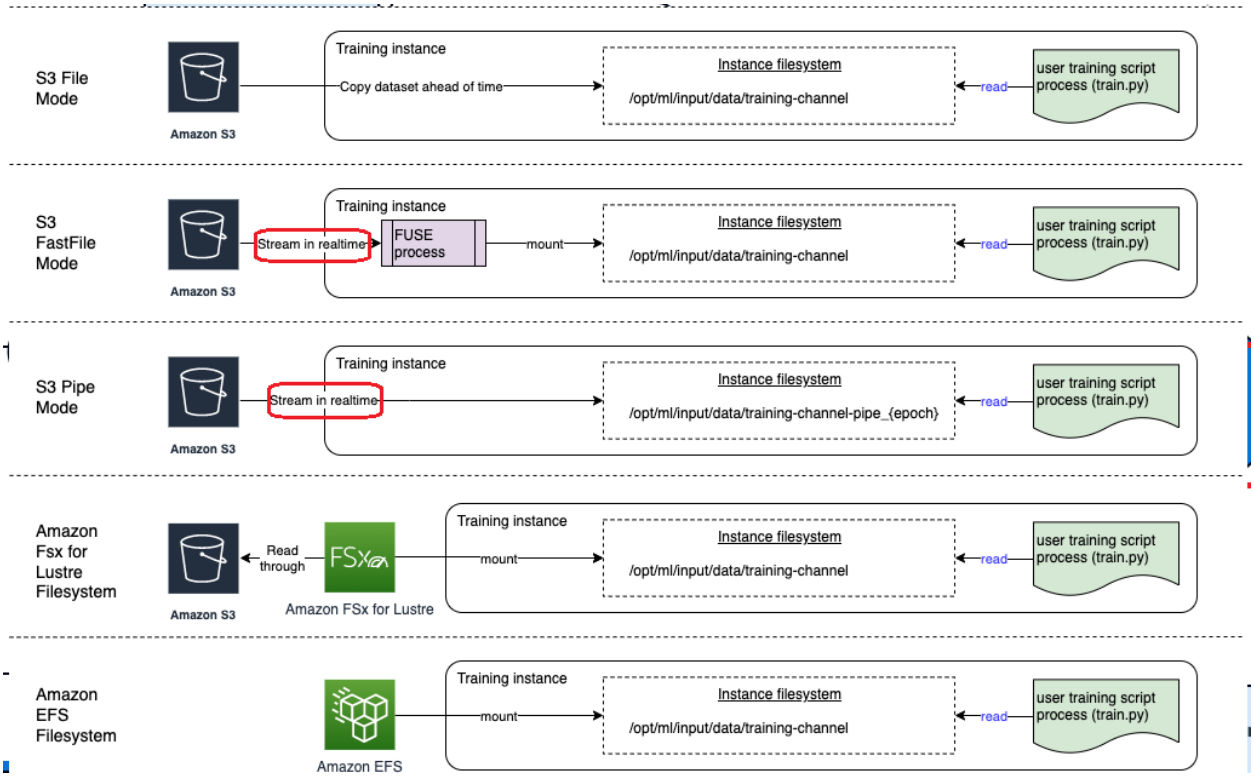
- **Amazon SageMaker Training Compiler**

SageMaker Training Compiler is a capability of SageMaker that makes these hard-to-implement optimizations to reduce training time on GPU instances. The compiler optimizes DL models to accelerate training by more efficiently using SageMaker machine learning (ML) GPU instances. SageMaker Training Compiler is available at no additional charge within SageMaker and can help reduce total billable time as it accelerates training.



SageMaker Access Training Data with SageMaker

When you create a training job, you specify the location of a training dataset and an input mode for accessing the dataset. For data location, Amazon SageMaker supports Amazon Simple Storage Service (Amazon S3), Amazon Elastic File System (Amazon EFS), and Amazon FSx for Lustre. The input modes determine whether to stream data files of the dataset in real time or download the whole dataset at the start of the training job.



Use Incremental Training in Amazon SageMaker

Why/Use incremental training to:

- Train a new model using an expanded dataset that contains an underlying pattern that was not accounted for in the previous training and which resulted in poor model performance.
- Use the model artifacts or a portion of the model artifacts from a popular publicly available model in a training job. You don't need to train a new model from scratch.
- Resume a training job that was stopped.
- Train several variants of a model, either with different hyperparameter settings or using different datasets.

Train Using SageMaker Managed Warm Pools

SageMaker managed warm pools let you retain and reuse provisioned infrastructure after the completion of a training job to reduce latency for repetitive workloads, such as iterative experimentation or running many jobs consecutively. Subsequent training jobs that match specified parameters run on the retained warm pool infrastructure, which speeds up start times by reducing the time spent provisioning resources.

How it works

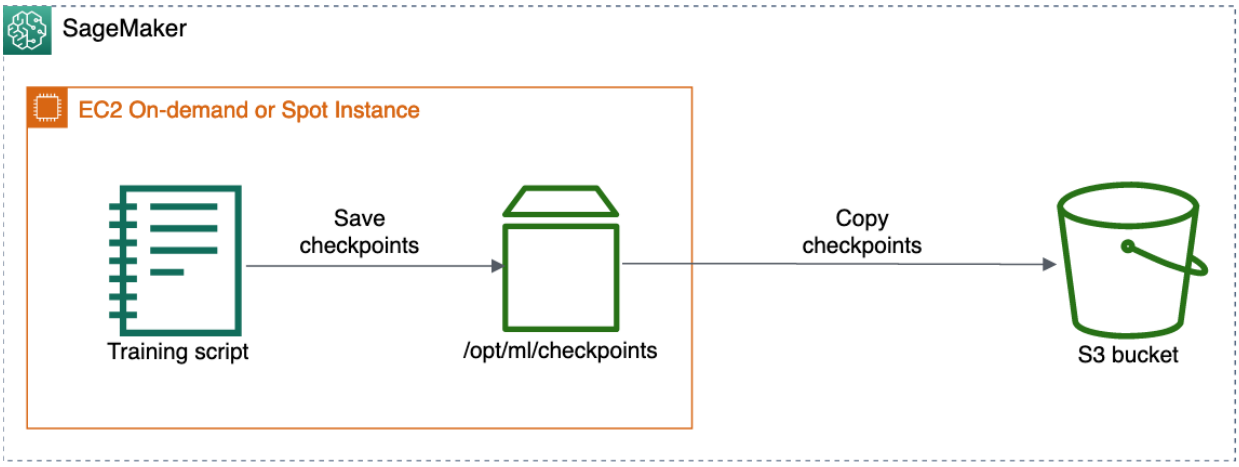
To use SageMaker managed warm pools and reduce latency between similar consecutive training jobs, create a training job that specifies a **KeepAlivePeriodInSeconds** value in its **ResourceConfig**.

Checkpoints for frameworks and algorithms in SageMaker

Use checkpoints **to save snapshots of ML models** built on your preferred frameworks within SageMaker.

SageMaker frameworks and algorithms that support checkpointing

SageMaker supports checkpointing for AWS Deep Learning Containers and a subset of built-in algorithms without requiring training script changes. SageMaker saves the checkpoints to the default local path '/opt/ml/checkpoints' and copies them to Amazon S3.



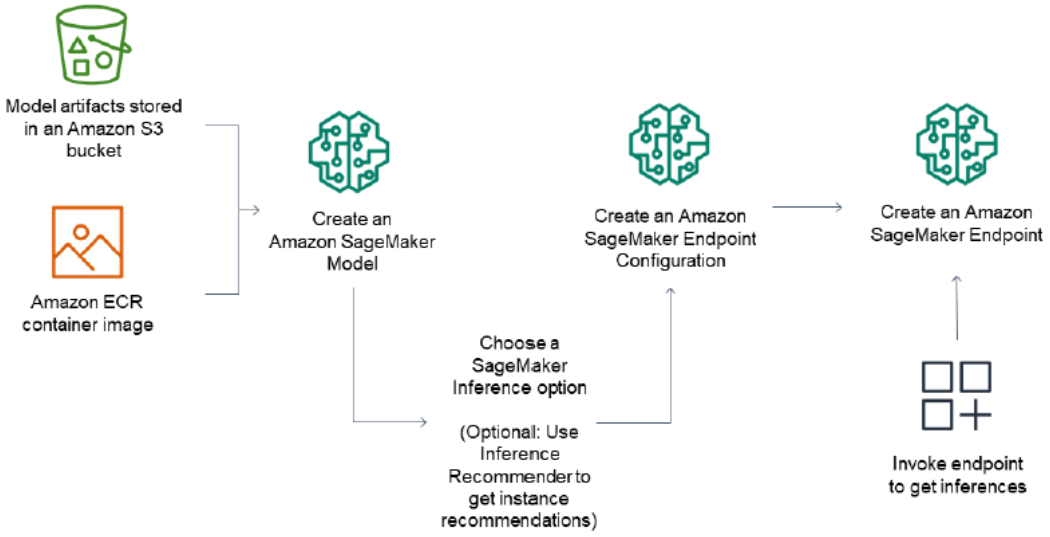
Deploy Models (for Inference)

Options

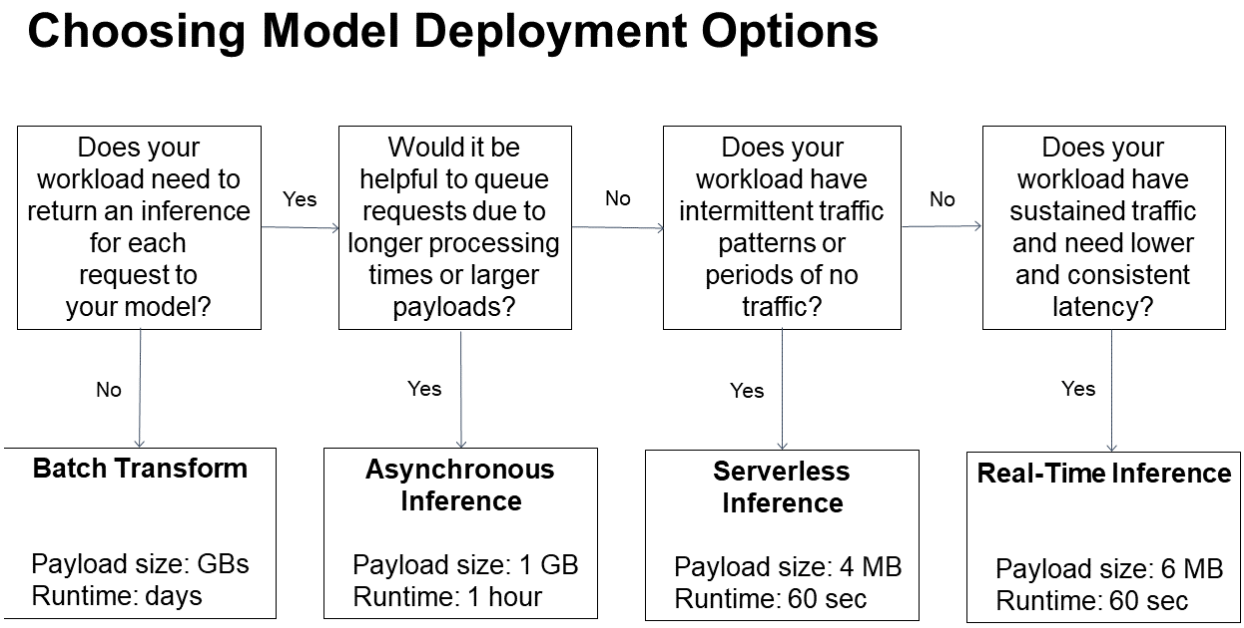
	JumpStart	ModelBuilder	CloudFormation
SageMaker feature	Use in Studio to accelerate your foundational model deployment.	Deploy models using ModelBuilder from the Python SDK.	Deploy and manage models at scale with AWS CloudFormation.
Description	Use the Studio UI to deploy pre-trained models from a catalog to pre-configured inference endpoints . This option is ideal for citizen data scientists, or for anyone who wants to deploy a model without configuring complex settings.	Use the ModelBuilder class from the Amazon SageMaker Python SDK to deploy your own model and configure deployment settings. This option is ideal for experienced data scientists, or for anyone who has their own model to deploy and requires fine-grained control.	Use AWS CloudFormation and IaC for deploying and managing SageMaker models. This option is ideal for advanced users who require consistent and repeatable deployments.
Optimized for	Fast and streamlined deployments of popular open source models	Deploying your own models	Ongoing management of models in production
Considerations	Lack of customization for container settings and specific application needs	No UI, requires that you're comfortable developing and maintaining Python code	Requires infrastructure management and organizational resources, and also requires familiarity with the AWS SDK for Python (Boto3) or with AWS CloudFormation templates.

Steps

1. Steps for model deployment



2. Inference options



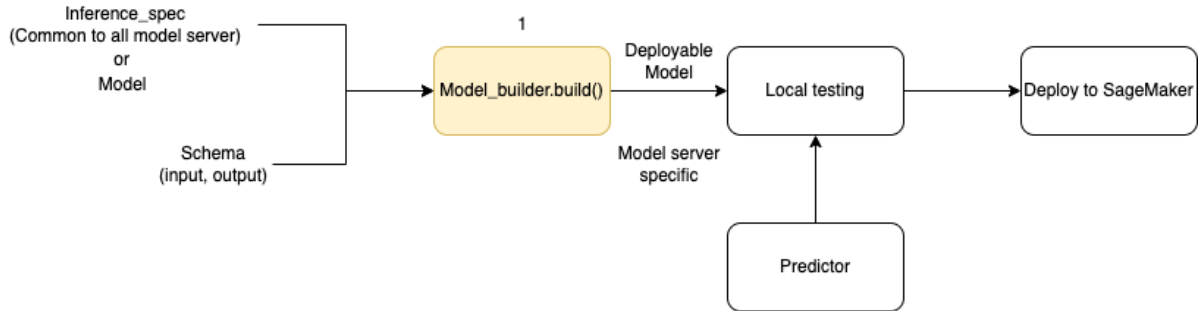
3. Advanced endpoint options

- **Host multiple models in one container behind one endpoint –** Use this option if you have multiple models that use the same framework and can share a container. This option helps you optimize costs by improving endpoint utilization and reducing deployment overhead.
- **Host multiple models which use different containers behind one endpoint –** Use this option if you have multiple models that use different frameworks and require their own containers. You get many of the benefits of Multi-Model Endpoints and can deploy a variety of frameworks and models.
- **Serial Inference Pipelines –** Use this option if you want to host models with pre-processing and post-processing logic behind an endpoint. Inference pipelines are fully managed by SageMaker and provide lower latency because all of the containers are hosted on the same Amazon EC2 instances.

4. Bring your own model

Create a model in Amazon SageMaker with ModelBuilder

Steps for model



Set up Online Explainability with SageMaker Clarify

1. EnableExplanations parameter: It is evaluated for **each record** in the explainability request. If this parameter is evaluated to be **true**, then the record will be explained. If this parameter is evaluated to be **false**, then explanations are not be generated.
2. **Synthetic dataset:** SageMaker Clarify uses the Kernel SHAP algorithm. Given a record and the SHAP configuration, the explainer first generates a synthetic dataset. SageMaker Clarify then queries the model container for the predictions of the dataset, and then computes and returns the feature attributions.

Deployment Options

Amazon SageMaker offers the following four options to deploy models for inference.

Feature	Real-time inference	Batch transform	Asynchronous inference	Serverless inference
Autoscaling support	✓	N/A	✓	✓
Multi-model endpoint	✓			
Multi-container endpoint	✓			
Serial inference pipeline	✓	✓		
Inference Recommender	✓			
Shadow testing	✓			
Virtual private cloud support	✓	✓	✓	

Implement MLOps

Refer to the following FAQ topics for answers to commonly asked questions about Amazon SageMaker Model Dashboard.

Q. What is Model Dashboard?

Amazon SageMaker Model Dashboard is a centralized reposit

Options

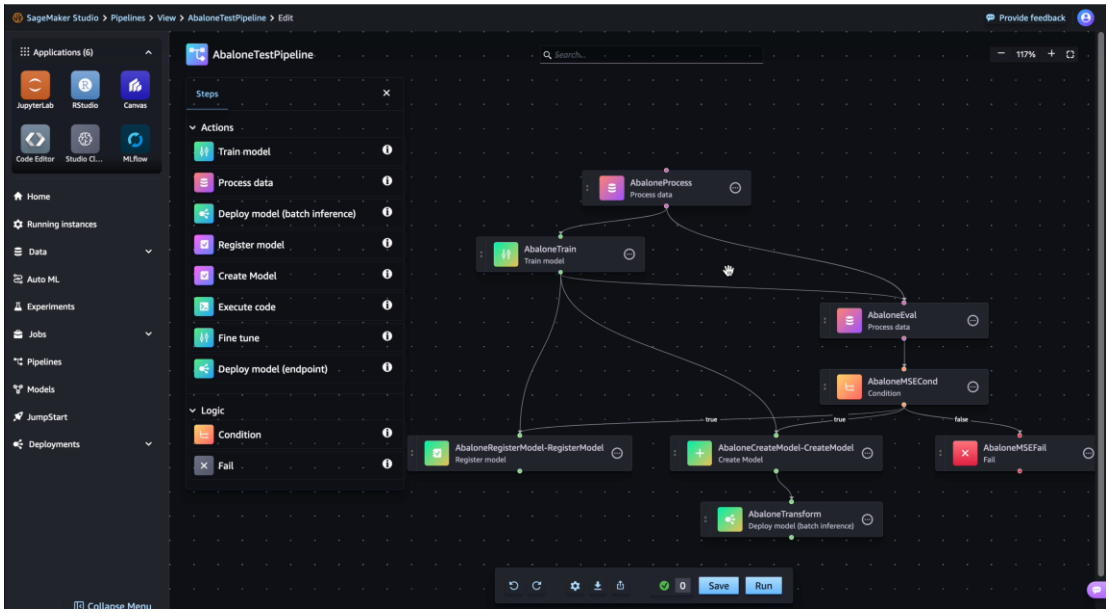
SageMaker offers the following workflow technologies:

1. **SageMaker Pipelines**: Tool for building and managing ML pipelines.
2. **Kubeflow Pipelines (Kubernetes Orchestration)**: SageMaker custom operators for your Kubernetes cluster and components for Kubeflow Pipelines.
3. **SageMaker Notebook Jobs**: On demand or scheduled non-interactive batch runs of your Jupyter notebook.

You can also leverage other services that integrate with SageMaker to build your workflow. Options include the following services:

4. **Airflow Workflows**: SageMaker APIs to export configurations for creating and managing Airflow workflows.
5. **AWS Step Functions**: Multi-step ML workflows in Python that orchestrate SageMaker infrastructure without having to provision your resources separately.

5. SageMaker Pipelines



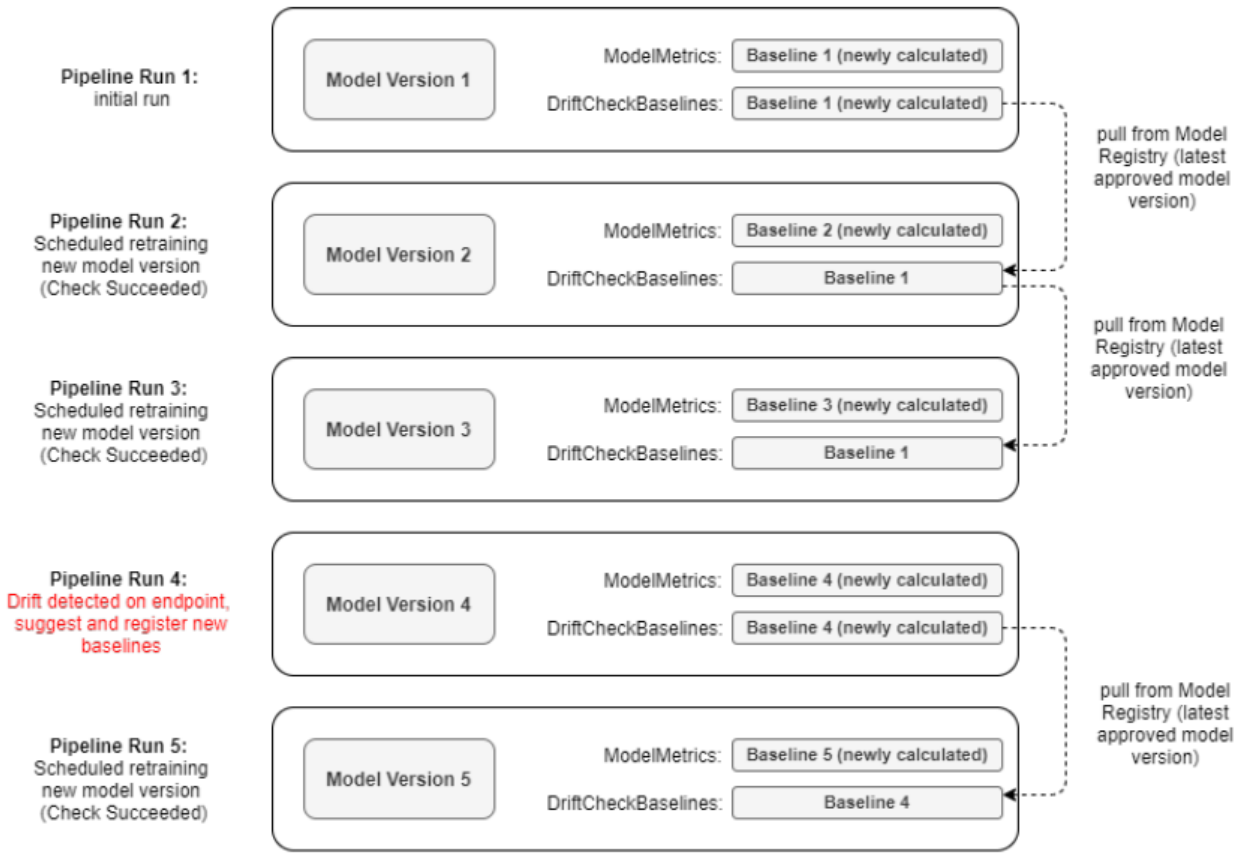
Amazon SageMaker Pipelines steps

- **Step types**

  (Processing, Training, Tuning, AutoML, Model, Create model, Register model, Deploy model (endpoint), Transform, Condition, Callback, Lambda, ClarifyCheck, QualityCheck, EMR, Notebook Job, Fail**)**

- **Step properties**
- **Step parallelism:** When a step does not depend on any other step, it runs immediately upon pipeline execution. However, executing too many pipeline steps in parallel can quickly exhaust available resources. Control the number of concurrent steps for a pipeline execution with **ParallelismConfiguration**.
- **Data dependency between steps**
- **Custom dependency between steps**
- **Use a custom image in a step**

Pipelines RUN WI Drift Rollback

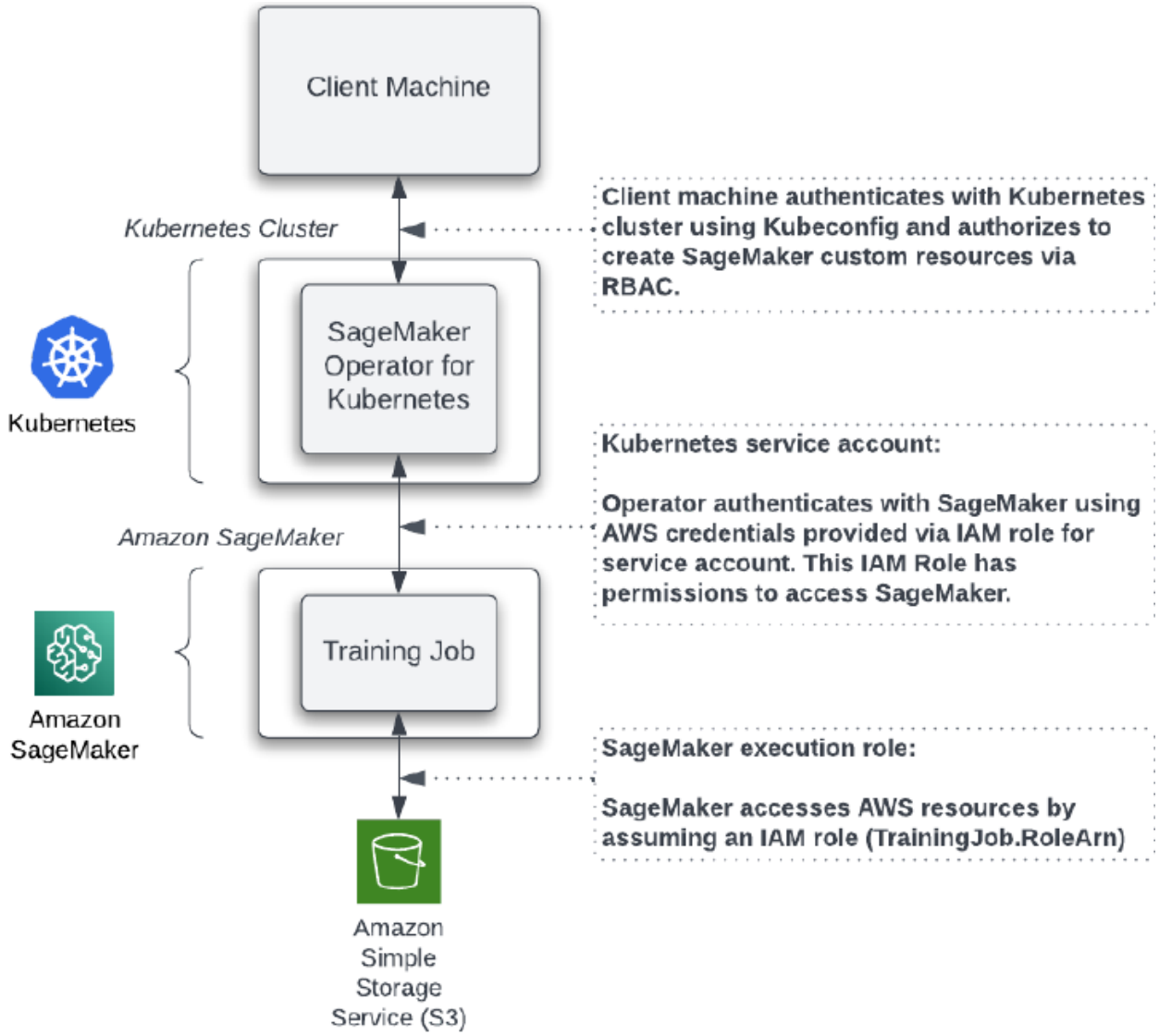


6. Sagemaker Notebook Jobs

Why use

- scenarios in which you might want to run your notebook as a noninteractive, scheduled job
- Scheduling jobs for model drift monitoring
- Exploring the parameter space for better models

7. Kubernetes Pipeline (KubeFlow)



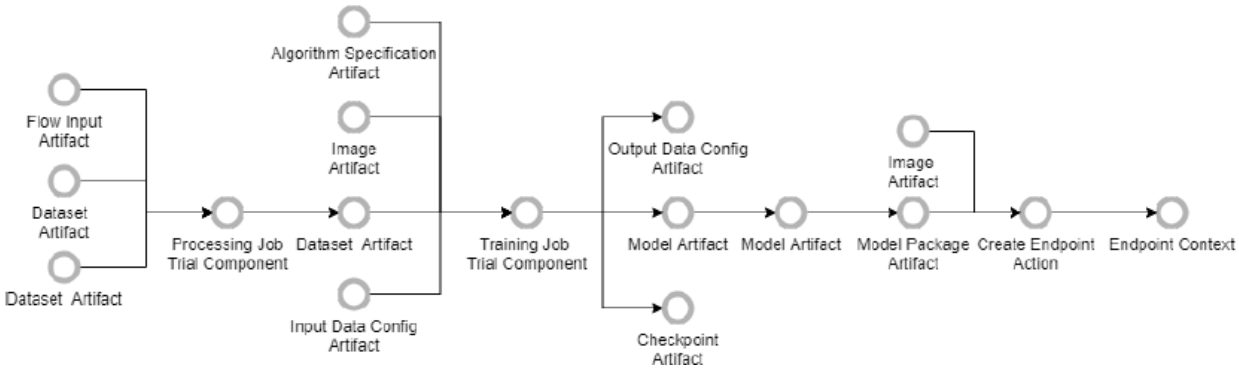
Converting Existing Kubernetes pipelines to use SageMaker (i.e. Bring it to AWS)

a. Install Kubeflow Pipelines on EKS
b. Configure your pipeline permissions to access SageMaker
c. Access the KFP UI (Kubeflow Dashboard) (http://localhost:8080)

Amazon SageMaker ML Lineage Tracking

Amazon SageMaker ML Lineage Tracking

- creates and stores information about the steps of a machine learning (ML) workflow from data preparation to model deployment
- automatically creates a connected graph of lineage



Register and Deploy Models with Model Registry

Topics

1. **Create/Delete a Model Group:** A Model Group contains a group of versioned models.

2. **Register a Model Version**

   You can register a SageMaker model by creating a model version that specifies the model group to which it belongs.

3. **View Model Groups and Versions**

4. **View and Update the Details of a Model Version**

   (list_model_packages, describe_model_package, etc.)

5. **Compare Model Versions**

6. **View and Manage Model Group and Model Version Tags**

7. **Share Models with SageMaker Canvas Users**

8. **Delete a Model Version**

9. **Update the Approval Status of a Model**

10. **Deploy a Model from the Registry**

    After you register a model version and approve it for deployment, deploy it to a SageMaker endpoint for real-time inference.

11. **Cross-account discoverability**

12. **View the Deployment History of a Model**

Model Collections

You can use Collections to group registered models that are related to each other and organize

Automate MLOps with Project templates

SageMaker project templates offer you the following choice of code repositories, workflow automation tools, and pipeline stages:

a. **Code repository**: AWS CodeCommit or third-party Git repositories such as GitHub and Bitbucket

b. **CI/CD workflow automation**: AWS CodePipeline or Jenkins
c. **Pipeline stages**: Model building and training, model deployment, or both

them in hierarchies to improve model discoverability at scale

Monitor data and model quality with Amazon SageMaker Model Monitor

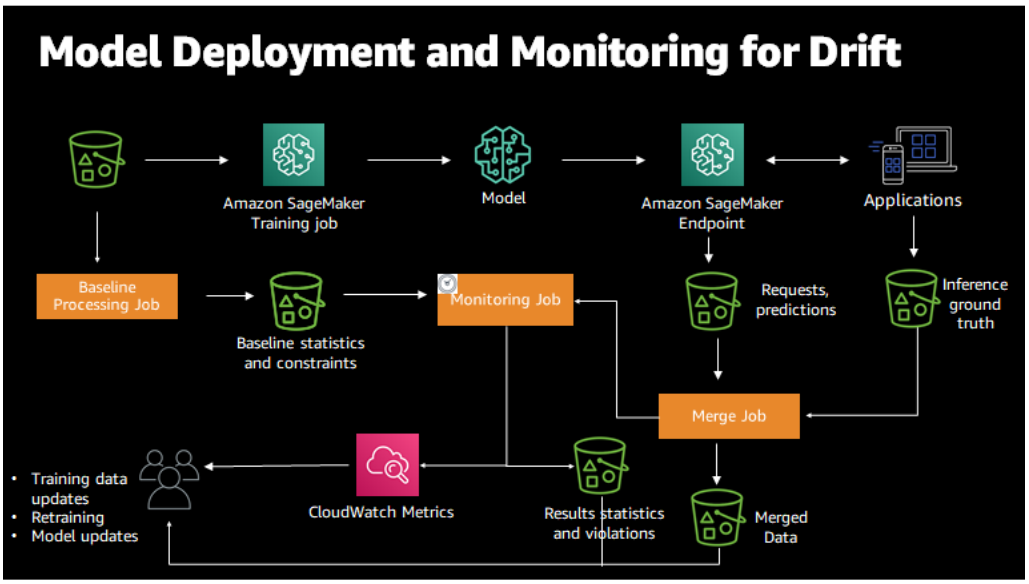
Options

With Model Monitor, you can set up:

1. Continuous monitoring with a real-time endpoint.
2. Continuous monitoring with a batch transform job that runs regularly.
3. On-schedule monitoring for asynchronous batch transform jobs.

1. Continuous monitoring with a real-time endpoint

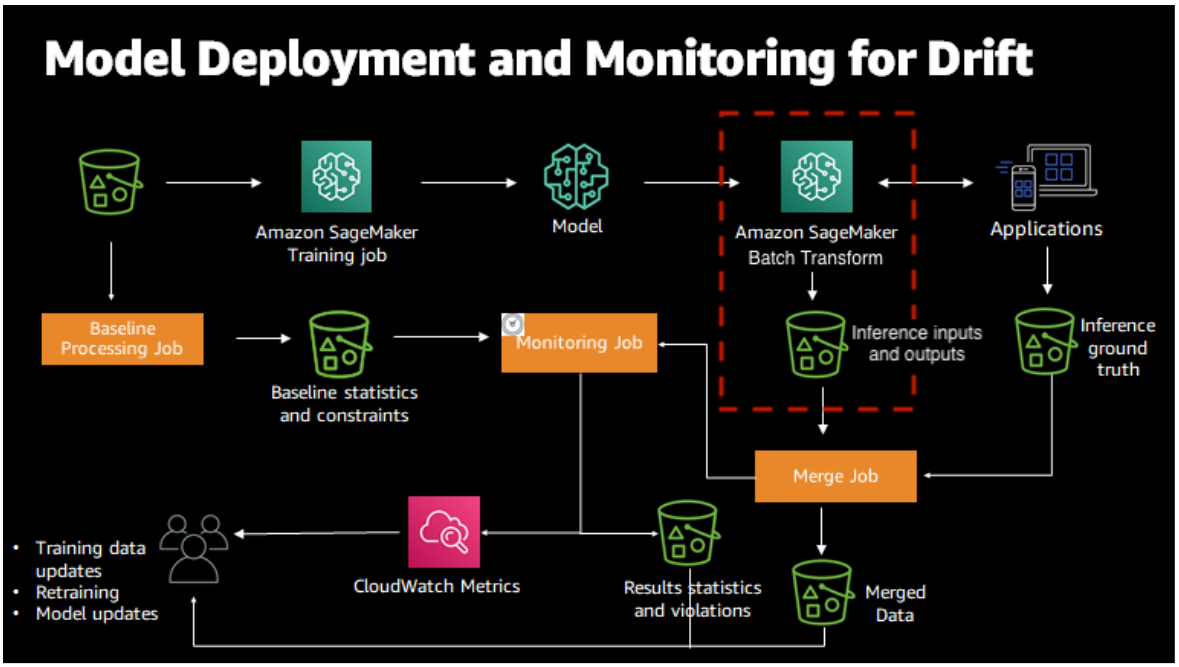
Amazon SageMaker Model Monitor automatically monitors ML models in production and notifies you by using rules to detect drift in your models and alerts you when it happens.



Capture data from real-time endpoint: To prevent impact to inference requests, Data Capture stops capturing requests at high levels of disk usage. It is recommended you keep your disk utilization below 75% in order to ensure data capture continues capturing requests.

2. Continuous monitoring with batch transform job that runs regularly

In this case, instead of receiving requests to an endpoint and tracking the predictions, Model Monitor monitors inference inputs and outputs. The following figure diagrams the process of monitoring a batch transform job.



Files out in Model Monitoring (Violations Report)

File Name	Description
statistics.json	Contains columnar statistics for each feature in the dataset that is analyzed. See the schema of this file in the next topic. **Note** This file is created only for data quality monitoring.
constraint_violations.json	Contains a list of violations found in this current set of data as compared to the baseline statistics and constraints file specified in the baseline_constaints and baseline_statistics paths.

Model Monitor Role

Model Monitor provides the following types of monitoring:

1. **Monitor data quality** - Monitor drift in data quality.
2. **Monitor model quality** - Monitor drift in model quality metrics, such as accuracy.
3. **Monitor Bias Drift for Models in Production** - Monitor bias in your model's predictions
4. **Monitor Feature Attribution Drift for Models in Production** - Monitor drift in feature attribution

1. Monitor data quality

Steps

a. Create a Baseline
b. Schedule data quality monitoring jobs
c. Schema for Statistics (statistics.json file)
d. CloudWatch Metrics
e. Schema for Violations (constraint_violations.json file)

a. Create a Baseline

- Start baseline processing job with DefaultModelMonitor.suggest_baseline(..)
- The baseline statistics for the dataset are contained in the statistics.json file and the suggested baseline constraints are contained in the constraints.json file in the location you specify withoutput_s3_uri.

b. Schedule data quality monitoring jobs

- call the create_monitoring_schedule() method of DefaultModelMonitor class instance
  - For real-time data -> Pass <<EndpointInput>> instance to the **endpoint_input** argument of your DefaultModelMonitor instance
  - For batch data -> Pass <<BatchTransformInput>> instance to the **batch_transform_input** argument of your **DefaultModelMonitor** instance

c. Schema for Statistics (statistics.json file)

Amazon SageMaker Model Monitor prebuilt container computes per column/feature statistics. The statistics are calculated for the baseline dataset and also for the current dataset that is being analyzed.

d. CloudWatch Metrics

You can use the built-in SageMaker Model Monitor container for CloudWatch metrics. When the emit_metrics option is Enabled in the baseline constraints file, SageMaker emits these metrics for each feature/column observed in the dataset in the following namespace

e. Schema for Violations (constraint_violations.json file)

The violations file is generated as the output of a MonitoringExecution, which lists the results of evaluating the constraints (specified in the constraints.json file) against the current dataset that was analyzed.

Violation Check Type	Description
data_type_check	If the data types in the current execution are not the same as in the baseline dataset, this violation is flagged.
completeness_check	If the completeness (% of non-null items) observed in the current execution exceeds the threshold specified in completeness threshold specified per feature, this violation is flagged.
baseline_drift_check	If the calculated distribution distance between the current and the baseline datasets is more than the threshold specified in monitorin g_config.comparison_threshold, this violation is flagged.
missing_column_check	If the # of columns in the current dataset is less than the # in the baseline dataset, this violation is flagged.
extra_column_check	..
categorical_values_check	If there are more unknown values in the current dataset than in the baseline dataset..

2. Monitor data quality

Steps

a. **Create a model quality baseline**
b. **Schedule model quality monitoring jobs**
c. **Ingest Ground Truth labels and merge them with predictions**
d. **Model quality metrics and Amazon CloudWatch monitoring**

a. Create a model quality baseline

You need to have a dataset that contains predictions from your model along with labels that represent the Ground Truth for your data.

- **Create a baseline job** using ModelQualityMonitor class provided by the SageMaker Python SDK and invoke suggest_baseline method.

b. Create Monitoring schedule

- **Batch Job** -> Call the create_monitoring_schedule() method of your ModelQualityMonitor class instance to schedule an hourly model quality monitor
- **real-time endpoint** -> pass your <<EndpointInput>> instance to the endpoint_input argument of your ModelQualityMonitor instance

c. Ingest Ground Truth labels and merge them with predictions

- You must upload Ground Truth data to an Amazon S3 bucket that has the same path format as captured data, which is of the following form:s3://*bucket/prefix/yyyy/mm/dd/hh*
- After you create and upload the Ground Truth labels, include the location of the labels as a parameter when you create the monitoring job.

d. Model quality metrics and Amazon CloudWatch monitoring

The specific metrics calculated depend on the type of ML problem:

- regression,
- binary classification
- multiclass classification

Monitoring these metrics is crucial for detecting model drift over time.

Monitoring model quality metrics with CloudWatch

If you set the value of the enable_cloudwatch_metrics to True when you create the monitoring schedule, model quality monitoring jobs send all metrics to CloudWatch.

3. Monitor Bias Drift in Models

Steps

a. Create a Bias Drift Baseline
b. Bias Drift Violations
c. Configure Parameters to Monitor Bias Drift
d. Schedule Bias Drift Monitoring Jobs
e. Inspect Reports for Data Bias Drift
f. CloudWatch Metrics for Bias Drift Analysis

a. Create a Bias Drift Baseline

model_bias_monitor.suggest_baseline(..) ***(Similar to other monitoring types)***

b. Bias Drift Violations

Here is the schema of the bias drift violations file.

- **facet** – The name of the facet, provided by configuration facetname_or_index.
- **facet_value** – The value of the facet, provided by configuration facet value_or_threshold.
- **metric_name** – The short name of the bias metric. For example, "CI" for class imbalance.
- **constraint_check_type** – The type of violation monitored. Currently only bias_drift_check is supported
- **description** – A descriptive message to explain the violation.

c. Configure Parameters to Monitor Bias Drift

d. Schedule Bias Drift Monitoring Jobs

e. Inspect Reports for Data Bias Drift

f. CloudWatch Metrics for Bias Drift Analysis

4. Monitor Feature Attribution Drift in Models

Steps

a. Create a SHAP Baseline for models in production..
b. ...

Evaluate, explain, and detect bias in models

- **Understand options for evaluating large language models with SageMaker Clarify**

The following

  - **Automatic model evaluation jobs**

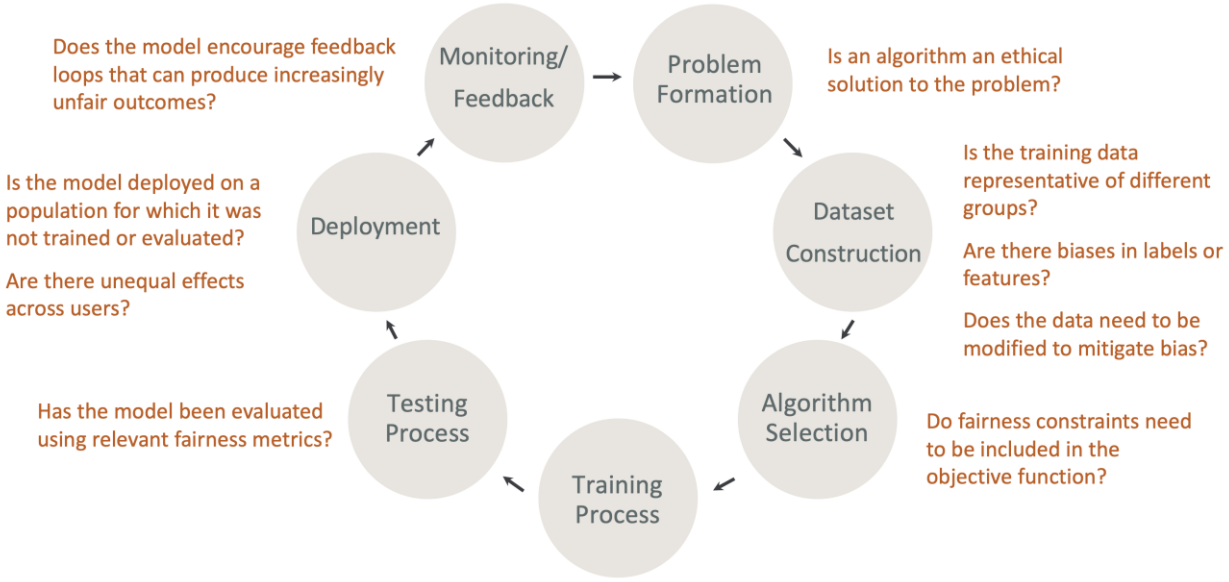
    Automatic model evaluation jobs support the use of a single model, using a JumpStart model.

    Alternatively, you can deploy the **fmeval** library into your own code base, and customize the model evaluation job for your own use cases.

  - **Model evaluation jobs that use humans workers**

    You can also employ **human workers** to manually evaluate your model responses for more subjective dimensions, such as helpfulness or style. To create a model evaluation job that uses human workers, you must use Studio.

- **Fairness, model explainability and bias detection with SageMaker Clarify**



Configure the Analysis

To analyze your data and models for explainability and bias using SageMaker Clarify, you must configure a processing job. Part of the configuration for this processing job includes the **Analysis configuration files, which includes** showing feature importance for a dataset in CSV format:

- pre-training and post-training bias metrics
- SHAP values
- partial dependence plots (PDPs)

Pre-training Bias Metrics

Bias metric	Description	Example question	Interpreting metric values
Class Imbalance (CI)	Measures the imbalance in the # of members for each facet values.	Could there be age-based biases due to not having enough data for the demographic outside a middle-aged facet?	Normalized range: [-1,+1] +ve -> more of favorable samples in the dataset.
Difference in Proportions of Labels (DPL)	Measures the imbalance of positive outcomes between different facet values.	Could there be age-based biases in ML predictions due to biased labeling of facet values in the data?	Range for normalize d binary & multicate gory facet labels: [-1,
Kullback-Leibler Divergence (KL) **&** **Jensen-Shannon Divergence (JS)**	Measures the divergence of outcome distributions of different facets.	How different are the distributions for loan application outcomes for different demographic groups?	Range for binary, multicategory, continuous: [0, +∞)
Lp-norm (LP)	Measures a p-norm difference ...	How different are the distributions for loan application outcomes for different demographics?	Same as above
Total Variation Distance (TVD)	Measures half of the L1-norm difference ..	How different are the distributions for loan application outcomes for different demographics?	Same as above
Kolmogorov-Smirnov (KS)	Measures maximum divergence between outcomes in distributions for different facets in a dataset	Which college application outcomes manifest the greatest disparities by demographic group?	Same as above

Post-training Bias Metrics

Bias metric	Description	Example question
Difference in Positive Proportions in Predicted Labels (DPPL)	Measures the difference in the proportion of positive predictions between the different facets.	Has there been an imbalance across demographic groups..
Disparate Impact (DI)	Measures the ratio of proportions of the predicted labels for the diff facet	Has there been an imbalance across demographic groups in the predicted positive outcomes?
Conditional Demographic Disparity in Predicted Labels (CDDPL)	Measures the disparity of predicted labels as a whole, but also by subgroups	Do some demographic groups have a larger proportion of rejections for loan?
Counterfactual Fliptest (FT)	Examines and assesses whether similar members of facet have different model predictions	Is one group of a age matched closely on all features yet paid more on average?
Accuracy Difference (AD)	.. difference between the prediction accuracy.	Does the model predict labels as accurately for applications across all demographic groups
Recall Difference (RD)	Compares the recall ...	... having higher recall for one age group ..?
Difference in Condiional Rejection (DCR)		
Difference in Rejection Rates (DRR)		
Difference in Acceptance Rates (DAR)		
Specificity difference (SD)	specificity of the model between favored and disfavored facets	
Treatment Equality (TE)	Measures the difference in the ratio of FP to FN between facets.	Is the ratio of FP to FN the same across all age demographics?
Generalized entropy (GE)	Measures the inequality in benefits b assigned to each input by the model predictions	Does one model candidate lead to a more uneven distribution of desired outcomes than other?

Model governance to manage permissions and track model performance

Tools and purpose

- **Amazon SageMaker Role Manager** - Manage least-privilege permissions for ML practitioners
- **Amazon SageMaker Assets -** streamlines ML governance by easily publish, share, and subscribe to ML assets and data assets.
- **Amazon SageMaker Model Cards** - create detailed model documentation
- **Amazon SageMaker Model Dashboard** - Gain visibility into your models with centralized dashboards

Amazon SageMaker Model Dashboard

SageMaker Model Dashboard integrates valuable information from

- Amazon SageMaker Model Monitor
- Transform Jobs
- Endpoints
- ML Lineage Tracking
- Amazon CloudWatch

Model Card – STEPS to create

1. **Enter model details and intended use**
2. **Enter training details**
   - Optimization direction
   - Metric
   - Description
3. **Enter evaluation details**
   If you have existing evaluation reports generated by SageMaker Clarify or Model Monitor, either provide an S3 URI for those reports or upload them manually to add them to the model card
4. **Enter any additional details (only structure in the card that can be modified)**

Model Dashboard

- **Risk rating**: A user-specified parameter from the model card (**low**, **medium**, or **high** value).
- **Model Monitor alerts:** Model Monitor alerts are a primary focus of the Model Dashboard, and reviewing the existing documentation on the various monitors provided by SageMaker is a helpful way to get started. For an in-depth explanation of the SageMaker Model Monitor feature and sample notebooks, see Data and model quality monitoring with Amazon SageMaker Model Monitor.
- **Model Monitor status** values by the following monitor types **None**, **Inactive**, **OK**)
    - ***Data Quality*:** Compares live data to training data
    - ***Model Quality***: Compares the predictions that the model makes with the actual Ground Truth labels that the model attempts to predict.
    - ***Bias Drift*:** Compares the distribution of live data to training data, which can also cause inaccurate predictions.
    - ***Feature Attribution Drift*:** Also known as explainability drift. Compares the relative rankings of your features in training data versus live data, which could also be a result of bias drift.

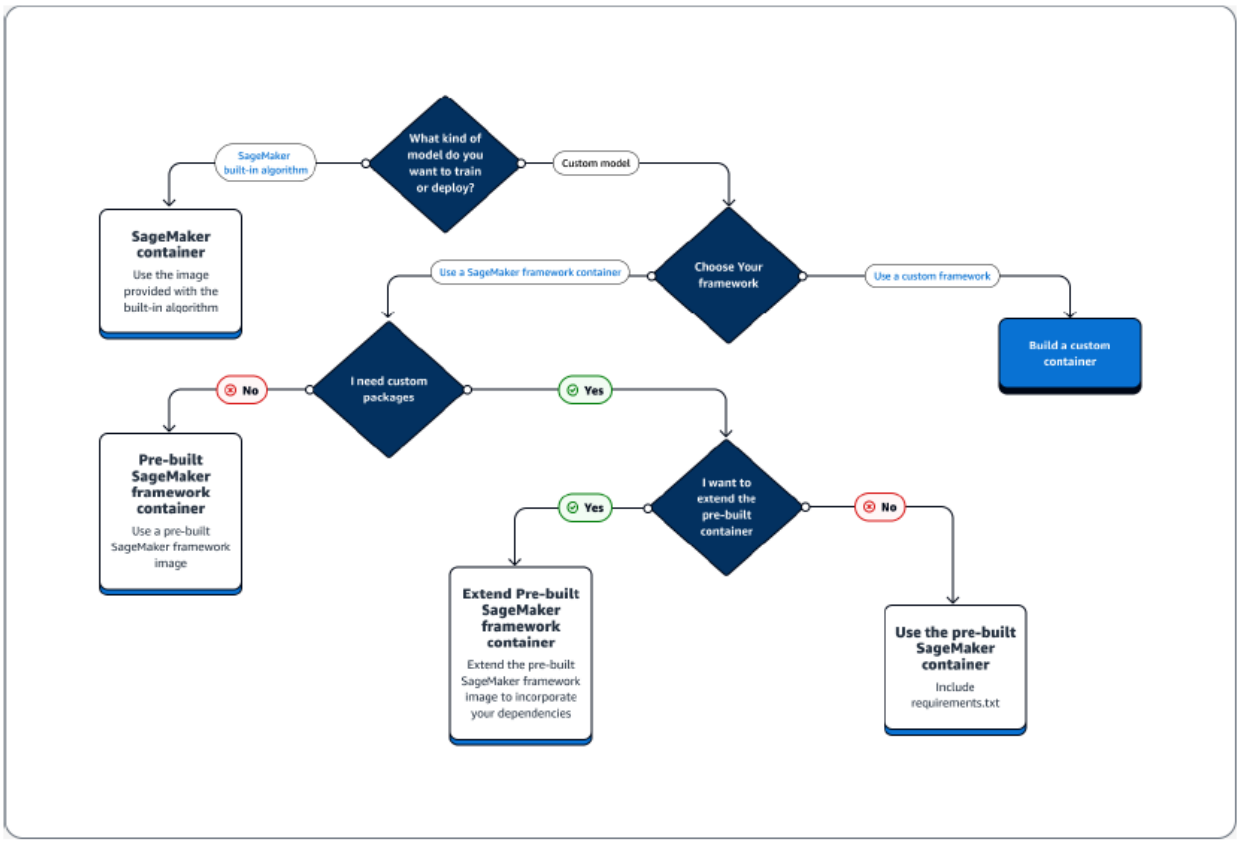
- **Endpoint**: The endpoints which host your model for real-time inference.
- **Batch transform job**: The most recent batch transform job that ran using this model.
- **Model details:** Each entry in the dashboard links to a model details page where you can dive deeper into an individual model.

Docker containers for training and deploying models

- **Scenarios for Running Scripts, Training Algorithms, or Deploying Models with SageMaker**

  The following decision tree illustrates three main scenarios:

  - **Use cases for using pre-built Docker containers with SageMaker**
  - **Use cases for extending a pre-built Docker container**
  - **Use case for building your own container**



Extend a Pre-built Container

To extend a pre-built SageMaker image, you need to set the following environment variables within your Dockerfile.

- SAGEMAKER_SUBMIT_DIRECTORY: The directory within the container in which the Python script for training is located.

- SAGEMAKER_PROGRAM: The Python script that should be invoked and used as the entry point for training.

Building your own container with your own algorithms and models

- Containers with custom training algorithms
- Containers with custom inference code

- **Troubleshooting your Docker containers and deployments**
  - **Error: SageMaker has lost the Docker daemon.**

    To fix this error, restart Docker using the following command.

    ```
    sudo service docker restart
    ```

  - **Error: The /tmp** directory of your Docker container has run out of space.

    Docker containers use the / and /tmp partitions to store code. These partitions can fill up easily when using large code modules in local mode. The SageMaker Python SDK supports specifying a custom temp directory for your local mode root directory to avoid this issue.

    To specify the custom temp directory in the Amazon Elastic Block Store volume storage, create a file at the following path ~/.sagemaker/config.yaml and add the following configuration. The directory that you specify as container_root must already exist. The SageMaker Python SDK will not try to create it.

    ```
    local:
      container_root: /home/ec2-user/SageMaker/temp
    ```

    With this configuration, local mode uses the /temp directory and not the default /tmp directory.

  - **Low space errors on SageMaker notebook instances**

    A Docker container that runs on SageMaker notebook instances uses the root Amazon EBS volume of the notebook instance by default. To resolve low space errors, provide the path of the Amazon EBS volume attached to the notebook instance as part of the volume parameter of Docker commands.

    ```
    docker run -v EBS-volume-path:container-path
    ```

Configure security in Amazon SageMaker

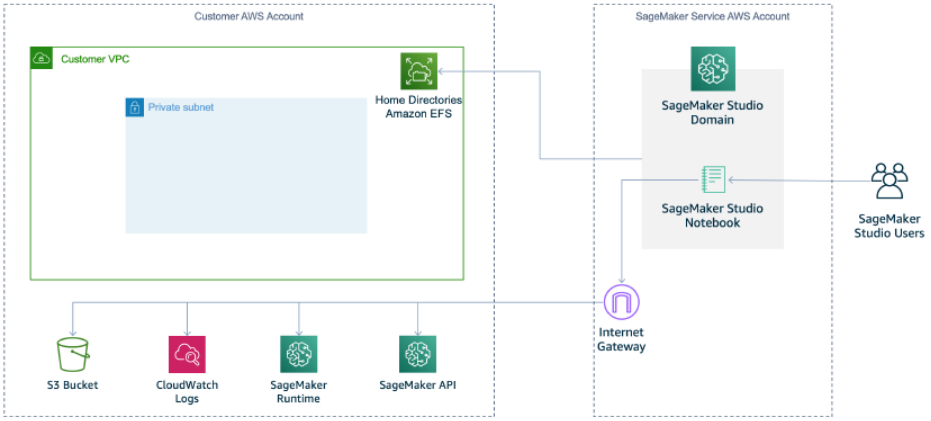
- **Data Privacy in Amazon SageMaker**

  You can opt out of sharing aggregated metadata with SageMaker training when creating a training job using the CreateTrainingJob API.

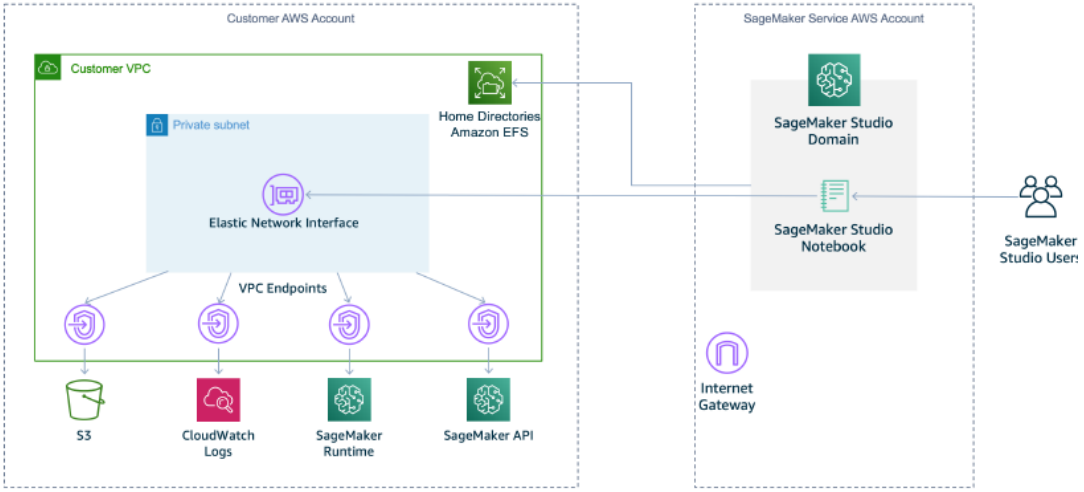
- **Data Protection in Amazon SageMaker**
- **Identity and Access Management for Amazon SageMaker**
- **Logging and Monitoring**
- **Compliance validation for Amazon SageMaker**
- **Resilience in Amazon SageMaker**
- **Infrastructure Security in Amazon SageMaker**
    - SageMaker Scans all pre-built SageMaker images, AWS Marketplace Training and Inference Containers, SageMaker built-in algorithm containers, for Security Vulnerabilities
    - **Connect Amazon SageMaker Studio in a VPC to External Resources**
        - By **default**, Amazon SageMaker Studio provides a network interface that allows communication with the internet through a VPC managed by SageMaker
        - **VPC only** - To prevent SageMaker from providing internet access to Studio.
            - You must use private subnets only. You cannot use public subnets in VpcOnly mode.
            - If you want to allow internet access, you must use a NAT gateway with access to the internet, for example through an internet gateway.
            - If you don't want to allow internet access, create interface VPC endpoints (AWS PrivateLink) to allow Studio to access the following services with the corresponding service names. You must also associate the security groups for your VPC with these endpoints.
                - SageMaker API : com.amazonaws.*region*.sagemaker.api.
                - SageMaker runtime: com.amazonaws.*region*.sagemaker.runtime. This is required to run Studio notebooks and to train and host models.
                - Amazon S3: com.amazonaws.*region*.s3.
                - SageMaker Projects: com.amazonaws.*region*.servicecatalog.
                - SageMaker Studio: aws.sagemaker.region.studio.
                - Any other AWS services you require.

Connect Amazon SageMaker Studio in a VPC to External Resources

a. Default communication with the internet



b. VPC only communication with the internet



Algorithms and packages in the AWS Marketplace

OPTIONS

- **SageMaker Algorithms**

  Algorithms enable you to perform end-to-end machine learning. It has two logical components: training and inference. Buyers can

  - use the training component to create training jobs in SageMaker and build a machine learning model.
  - use the inference component with the model artifacts generated during a training job to create a deployable model in their SageMaker account.
  - use the deployable model for real-time inference by using SageMaker hosting services.

- **SageMaker Model Packages**

  Buyers use a model package to build a deployable model in SageMaker. They can use the deployable model for real-time inference by using SageMaker hosting services.

- **Listings for your own algorithms and models with the AWS Marketplace**
- **Find and Subscribe to Algorithms and Model Packages on AWS Marketplace**
- **Usage of Algorithm and Model Package Resources**

Tools for monitoring AWS resources provisioned while using SageMaker

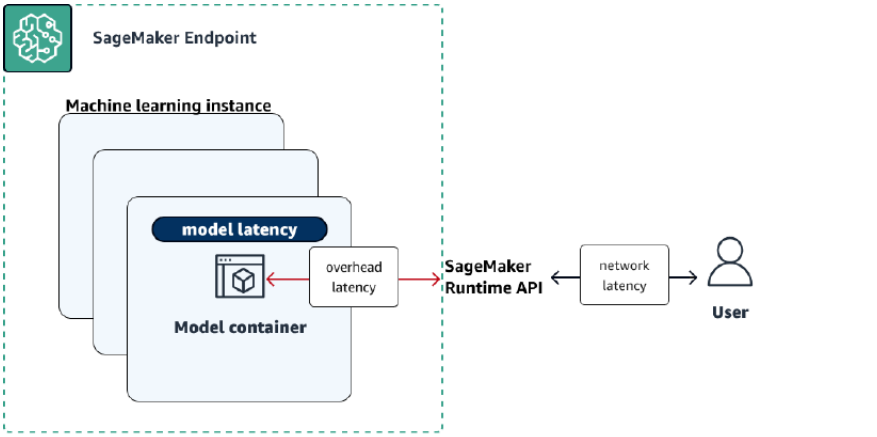
Metrics for monitoring Amazon SageMaker with Amazon CloudWatch

Metrics are available at a 1-minute frequency.

SageMaker Metrics and Dimensions

- **SageMaker endpoint invocation metrics**
  The following illustration shows how a SageMaker endpoint interacts with the Amazon SageMaker Runtime API.

  The overall time between sending a request to an endpoint and receiving a response depends on the following three components.



Total time (end-to-end) from request to response = network latency + overhead latency + model latency

- SageMaker inference component metrics
- SageMaker multi-model endpoint metrics
- SageMaker jobs and endpoint metrics
- SageMaker Inference Recommender jobs metrics
- SageMaker Ground Truth metrics
- Amazon SageMaker Feature Store metrics
- SageMaker pipelines metrics

Events that Amazon SageMaker sends to Amazon EventBridge

Examples of the actions that can be automatically triggered include the following

- Invoking an AWS Lambda function

- Invoking Amazon EC2 Run Command
- Relaying the event to Amazon Kinesis Data Streams
- Activating an AWS Step Functions state machine
- Notifying an Amazon SNS topic or an AWS SMS queue

SageMaker events monitored by EventBridge:

- SageMaker model state change
- Training job state change
- Hyperparameter tuning job state change
- Transform job state change
- Endpoint state change
- Feature group state change
- Model package state change
- Pipeline execution state change
- Pipeline step state change
- Processing job state change
- SageMaker image state change
- SageMaker image version state change