

# Amazon SageMaker

## End to End Machine Learning



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Today's Agenda

- Amazon SageMaker Studio
- Data Preparation for ML
- Model Training
- Model Deployment
- MLOPs Introduction

# SageMaker Studio

- Integrated development environment (IDE) for machine learning
- New features – multiple domains, real-time collaboration, scheduling notebooks
- RStudio Integration
- Demo

# SageMaker Studio

Provides end to end machine learning platform, increasing ML productivity by up to 10x

Computing Anomaly Scores

```
[1]: results = rcf.inference.predict(taxi_data_numpy)
scores = [datum['score'] for datum in results['scores']]
# add scores to taxi_data frame and print first few values
taxi_data['score']= pd.Series(scores, index=taxi_data.index)
taxi_data.head()

[2]: fig, ax1 = plt.subplots()
ax2 = ax1.twinx()

# Try this out - change 'start' and 'end' to zoom in on the anomalies found earlier in this notebook
# start, end = 0, len(taxi_data)
# start, end = 5500, 5500
taxi_data_subset = taxi_data[start:end]

ax1.plot(taxi_data_subset['value'], color='C0', alpha=0.8)
ax2.plot(taxi_data_subset['score'], color='C1')

ax1.set_xlabel('Taxi Ridethash', color='C0')
ax2.set_xlabel('Anomaly Score', color='C1')

ax1.tick_params('y', colors='C0')
ax2.tick_params('y', colors='C1')

ax1.set_yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2])
ax2.set_yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2])

fig.tight_layout()
fig
```

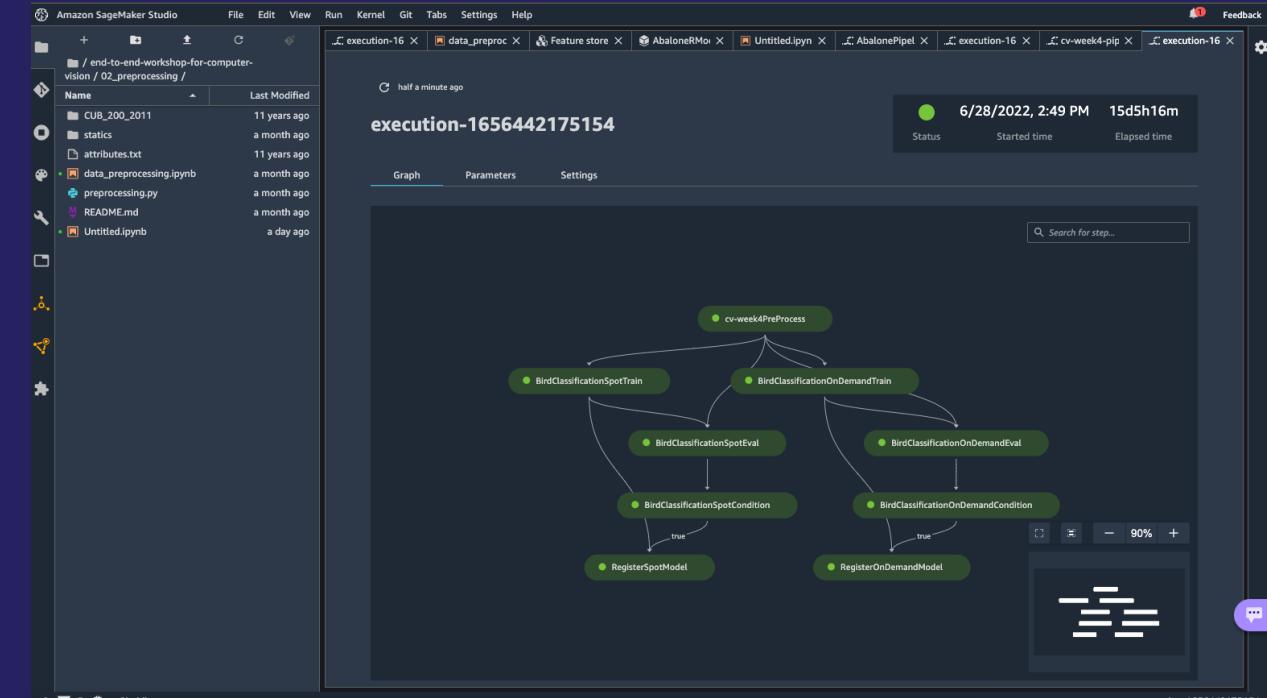
Note that the anomaly score spikes where our eyeball-norm method suggests there is an anomalous data point as well as in some places where our eyeballs are not as accurate.

Below we print and plot any data points with scores greater than 3 standard deviations (approx 99.9th percentile) from the mean score.

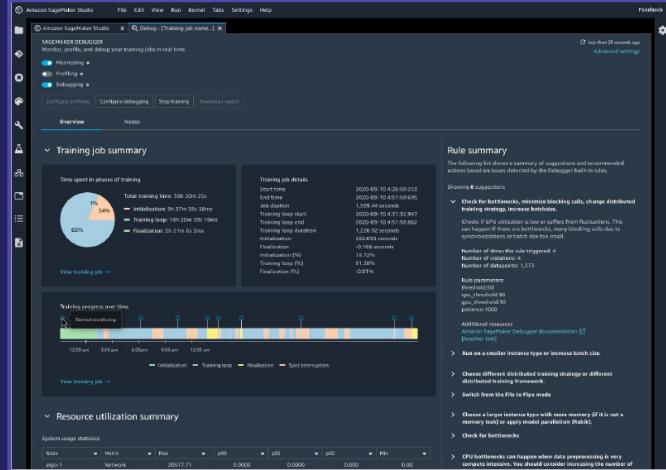
```
[3]: score_mean = taxi_data['score'].mean()
score_std = taxi_data['score'].std()
score_cutoff = score_mean + 3*score_std
anomalies = taxi_data_subset[taxi_data_subset['score'] > score_cutoff]
anomalies
```

The following is a list of known anomalous events which occurred in New York City within this timeframe:

```
[4]: anomalies
```

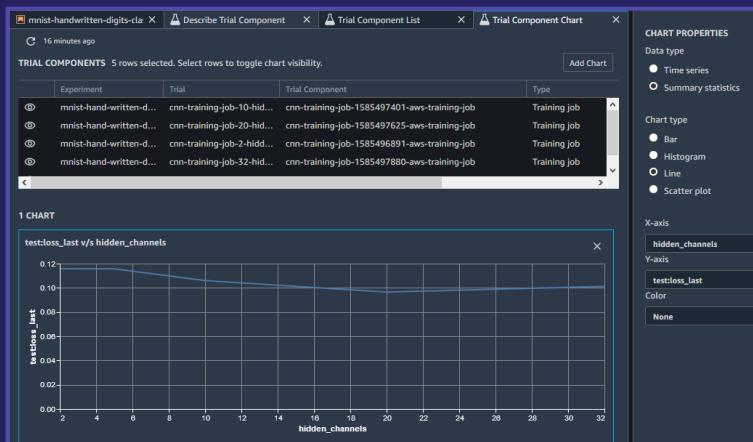


# Train Models Cost Effectively in SageMaker Studio Notebooks



**Experiment management and model tuning**  
Save weeks of effort by automatically tracking training runs and tuning hyperparameters

**Debug and profile training runs**  
Use real-time metrics to correct performance problems

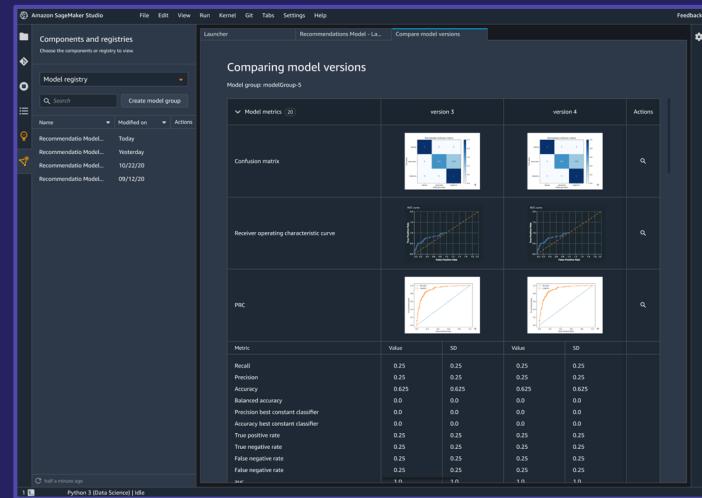


**Launch distributed training jobs from Studio Notebooks**  
Easily scale compute using SageMaker Training jobs

# Deploy Models at Scale using Sagemaker Studio

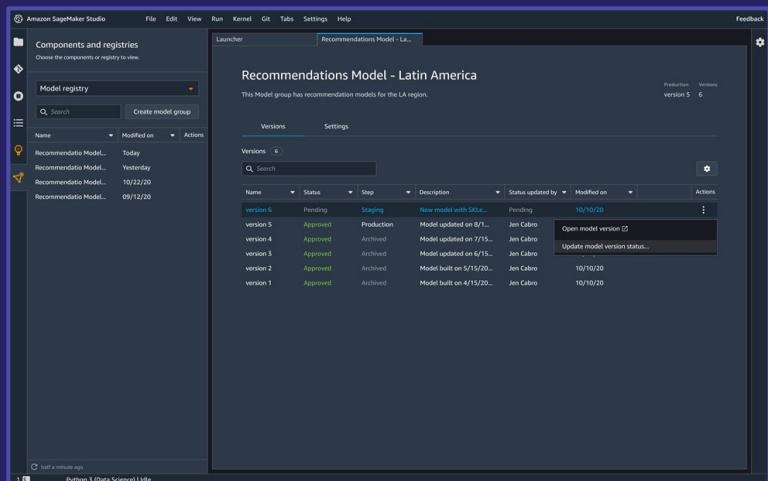
## Quickly create models or use prebuilt models

Decrease the operational overhead of deploying, scaling, and managing ML models with built in MLOps tooling including workflow automation, feature stores, and model registry interfaces



## Monitor hosted endpoints directly from Studio UI

Review model quality, data quality and endpoint configuration settings



## Deploy and test hosted endpoints from SageMaker Studio

70+ instance types with varying levels of compute and memory, including CPU and GPU instances

# SageMaker JumpStart in Studio

Easily access ML assets and quickly bring ML applications to market

The screenshot shows the Amazon SageMaker Studio interface. On the left, there's a sidebar with icons for AutoML, Experiments, Notebook jobs, Pipelines, Models, Deployments, and SageMaker JumpStart. The SageMaker JumpStart section is expanded, showing options like Models, notebooks, solutions, Pretrained models, example notebooks, & prebuilt solutions, and learning. The main content area has a dark background with a light gray header bar. The header includes the 'Amazon SageMaker Studio' logo, a navigation bar with File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help, and a user profile section for 'studio-user / Personal Studio'. Below the header, the title 'SageMaker JumpStart' is displayed, along with 'Show introduction' and 'Browse Shared Models' buttons. A search bar with the placeholder 'Search' is also present. The main content area features a section titled 'Foundation Models' with a sub-section 'Deploy foundation models trained on broad dataset and usable in wide range of use cases.' It lists several models:

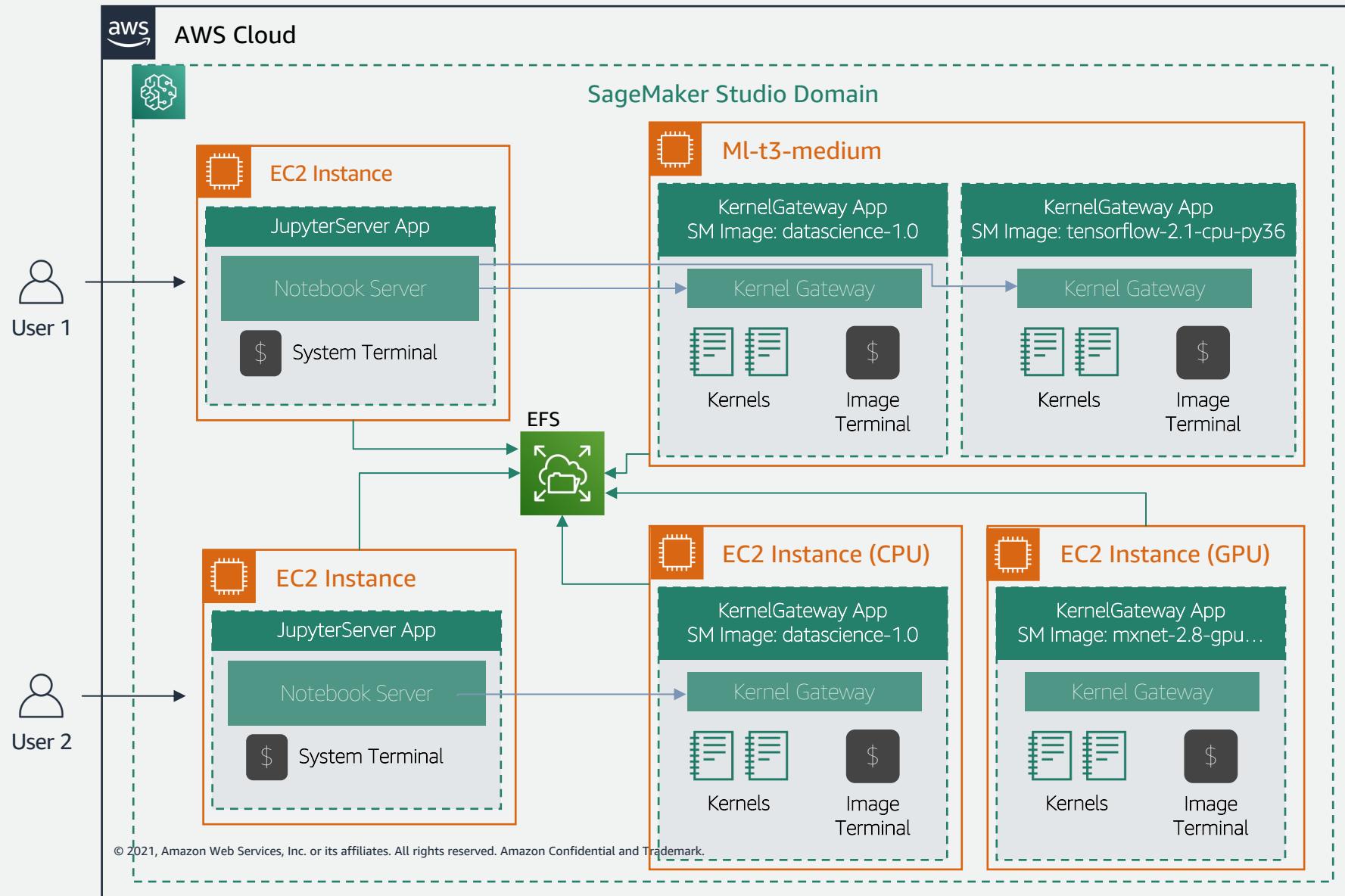
- Stable Diffusion 2.1 base**  
Featured | Text To Image  
Fine-tunable: Yes  
Source: Stability AI  
Pre-training Dataset: LAION-5B  
[View model >](#)
- FLAN-T5 XXL**  
Text2text Generation  
Pre-training Dataset: English Text  
Fine-tunable: No  
Source: Hugging Face  
[View model >](#)
- Alexa TM 20B**  
Featured | Text Generation  
Pre-training Dataset: Common Crawl (mC4) and...  
Fine-tunable: No  
Source: Alexa  
[View model >](#)
- Bloom**  
Featured | Te...  
Pre-training Dat...  
Fine-tunable: No  
Source: Hugging...  
[View model >](#)

# SageMaker Studio Architecture

SageMaker Studio is based on a Kernel Gateway architecture. It enables you to right-size compute resources for your applications that supports the reading and execution experience of the user's notebooks, terminals, and consoles . Each Studio user profile can run two types of Apps:

The **JupyterServer** - which is free of charge. Tip: you can work on Notebooks and UI plugins without selecting a kernel (and thus no cell execution!).

The **Kernel Gateway Apps** run on the selected compute resource of your Kernel. Compute instances can run up to four Kernel Apps (if CPU/RAM/GPU requirements are met).

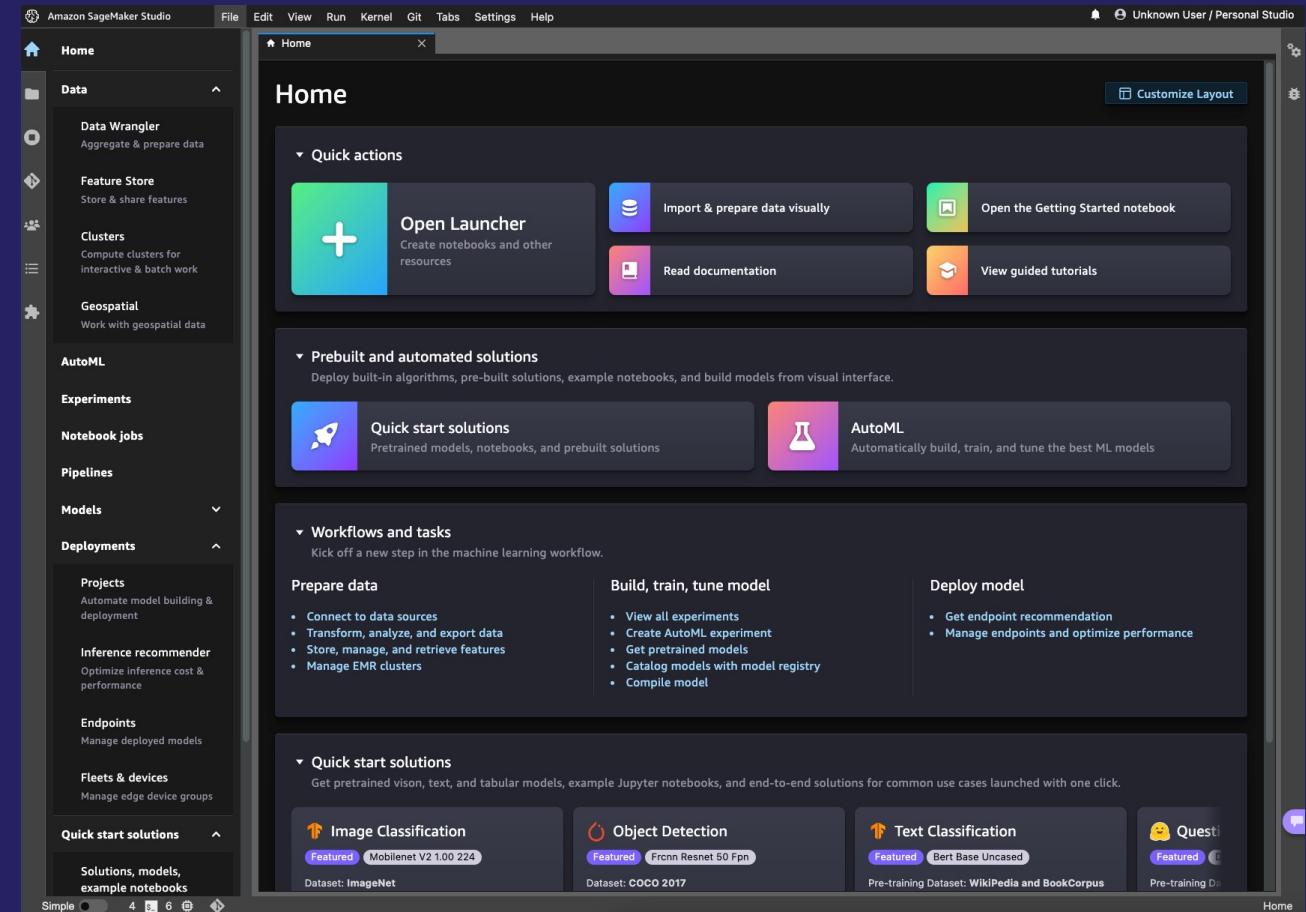


# Updated SageMaker Studio User Interface

Revamped interface enables better feature discoverability

## Updated SageMaker Studio UI:

- Customizable "Home"
- Embedded introduction documentation
- New "Learning resources"

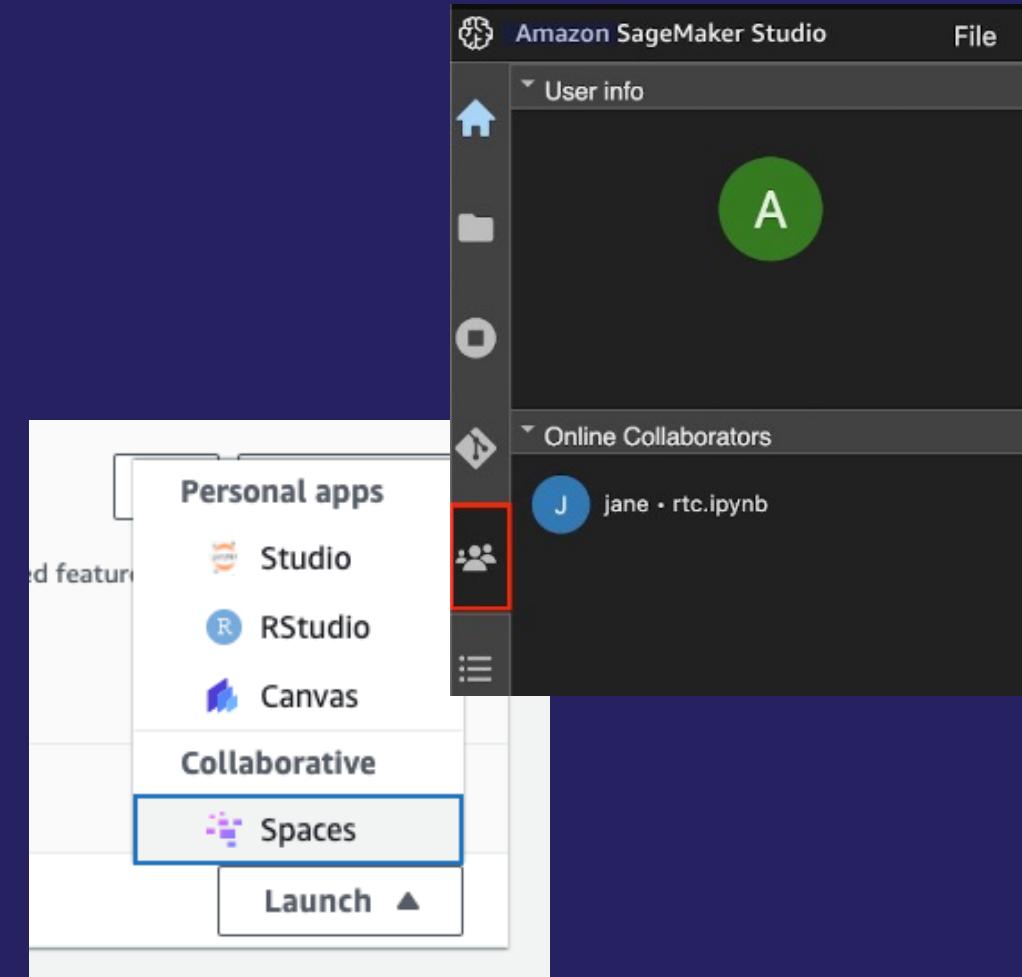


# SageMaker Studio now has shared spaces

Users can organize collaborative ML projects and initiatives by creating a shared IDE application that users utilize with their own Amazon SageMaker user profile

SageMaker Studio shared spaces provide logical separation of ML projects in a collaborative environment:

- Shared EFS directory in a space
- Collaborative editing capabilities
- Filtered SageMaker resources
  - Experiments
  - Model Registry
  - Pipelines, etc.



# SageMaker Studio now supports automated tagging

SageMaker automatically tags resources at domain, user and space level supporting detailed cost allocation

## SageMaker Studio Automated Tagging

- Domain, user or space level tagging
- All ARNable SageMaker resources
  - Training jobs
  - Processing jobs
  - Kernel Gateways
  - Endpoints
  - Etc.
- Supports more detailed cost allocation for administrators

Tags		Edit
Key	Value	
sagemaker:user-profile-arn	arn:aws:sagemaker:us-east-2:308466261674:user-profile/d-7rlqvske47cg/sean-user	
sagemaker:domain-arn	arn:aws:sagemaker:us-east-2:308466261674:domain/d-7rlqvske47cg	

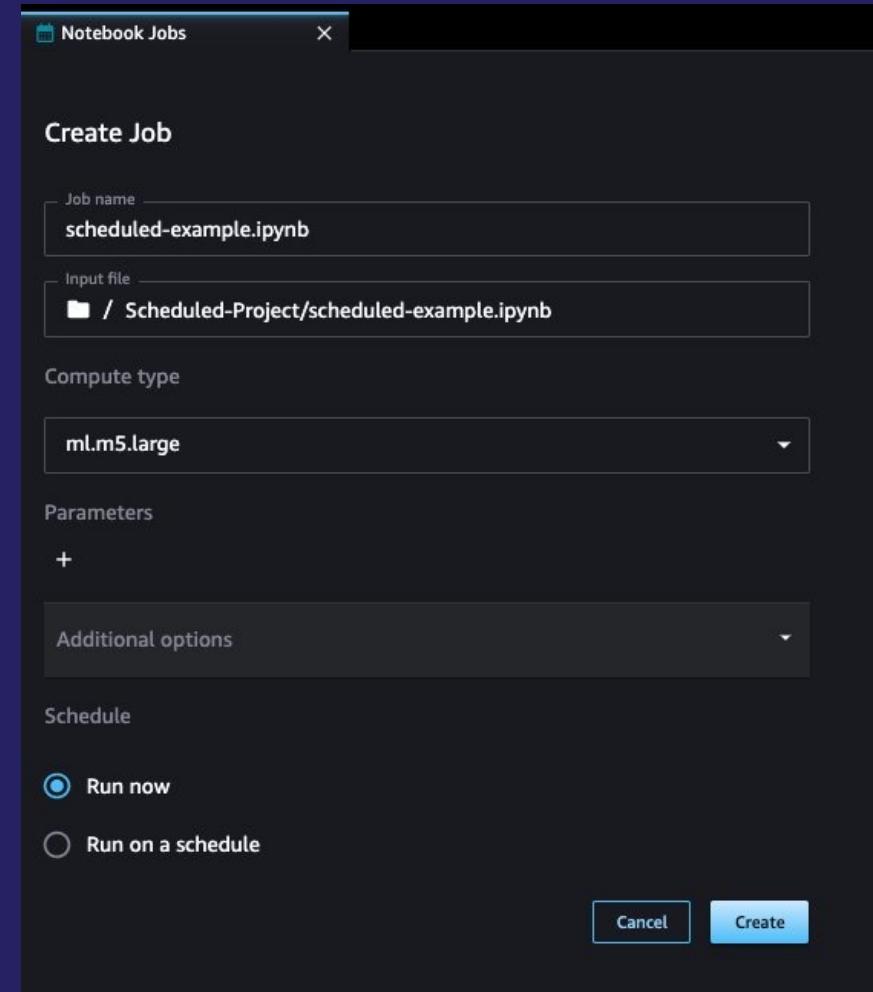


# SageMaker Studio now offers Notebook Jobs

Operationalize notebooks in two or three clicks

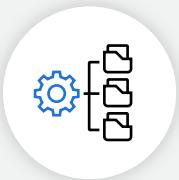
## With SageMaker notebook jobs:

- Run your notebooks as is or in a parameterized fashion with just a few simple clicks from the SageMaker Studio
- Run notebooks on a schedule or immediately
- No need for the end-user to modify their existing notebook code
- View the populated notebook cells after the job is complete, including any visualizations



# RStudio on SageMaker

Fully managed  
cloud-based RStudio IDE



## Fully managed cloud-based RStudio Workbench

Pre-configured R packages and publish using RStudio Connect.



## Broad selection of elastic compute

Easily dial up or down compute on the fly with compute-optimized and GPU-accelerated instances.



## Easy lift-and-shift migration

Bring current workbench license using AWS License Manager or own RStudio development environment in a custom docker image.



## Built-in security and compliance

Quick start login with AWS IAM Identity Center (Successor to AWS Single Sign-On), IAM access, VPC restriction, KMS encryption, and CloudWatch monitoring



## Unify your Python and R data science teams on SageMaker

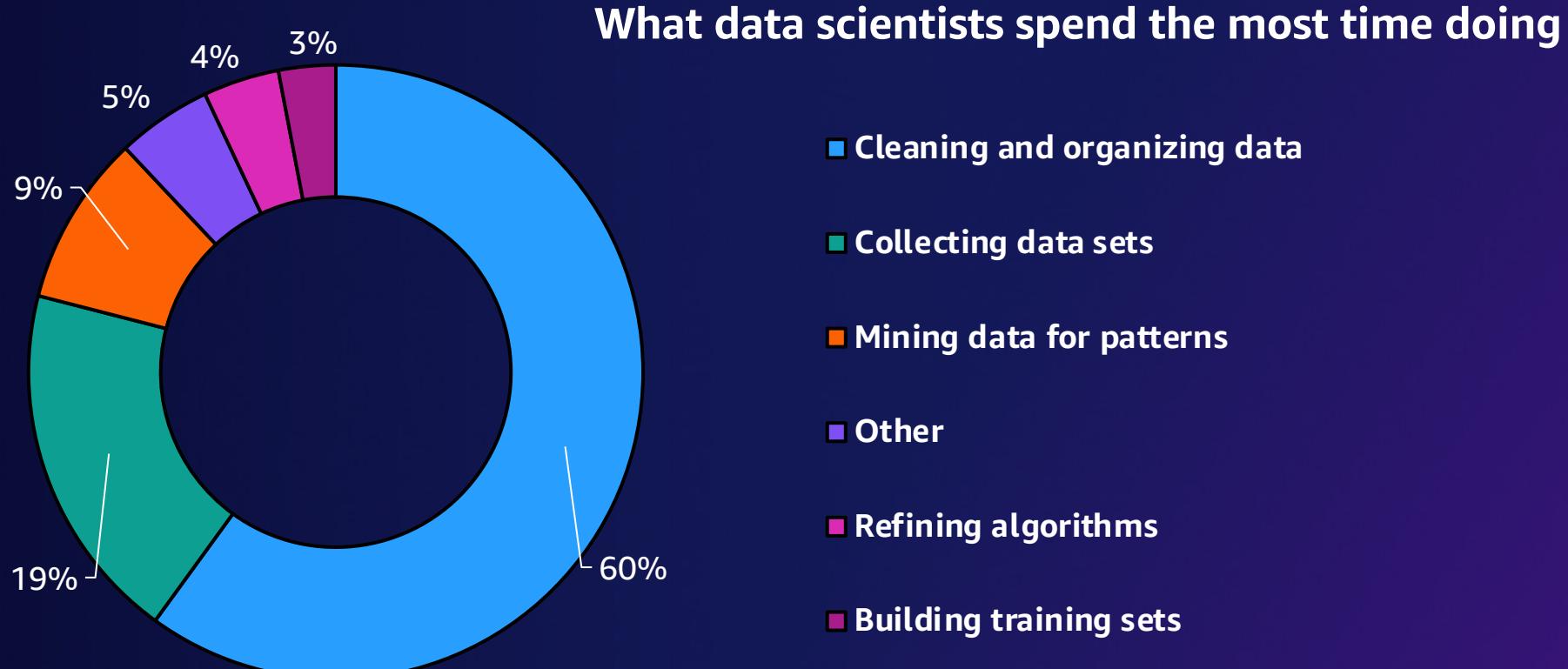
Seamlessly switch between RStudio IDE and Sagemaker Studio. Code, datasets, repositories, and other artifacts are automatically synchronized between the two environments

# Demo + Q&A

# Data preparation for ML

- SageMaker Studio Notebooks
- SageMaker Processing
- SageMaker Data Wrangler
- Glue interactive sessions
- EMR integration
- SageMaker Feature Store
- Demo

# 80% of time spent on data prep



Source: [Forbes survey of 80 data scientists, March 2016](#)

# SageMaker Studio Notebooks

The JupyterLab logo, featuring the word "jupyter" in a dark grey sans-serif font and "lab" in a larger orange sans-serif font.

SageMaker Studio Notebook Instances now come with JupyterLab 3 notebooks to boost developer productivity

Integrated debugger with support for breakpoints and variable inspection

Table of contents panel to more easily navigate notebooks

Filter bar for the file browser

Support for multiple display languages

Install extensions through pip, Conda, and Mamba

# SageMaker Studio Notebooks

Managed Environments and Flexible Compute

Set up notebook environment

Set up environment for "Untitled.ipynb".

Image

Data Science 3.0

Kernel

Python 3

Instance type

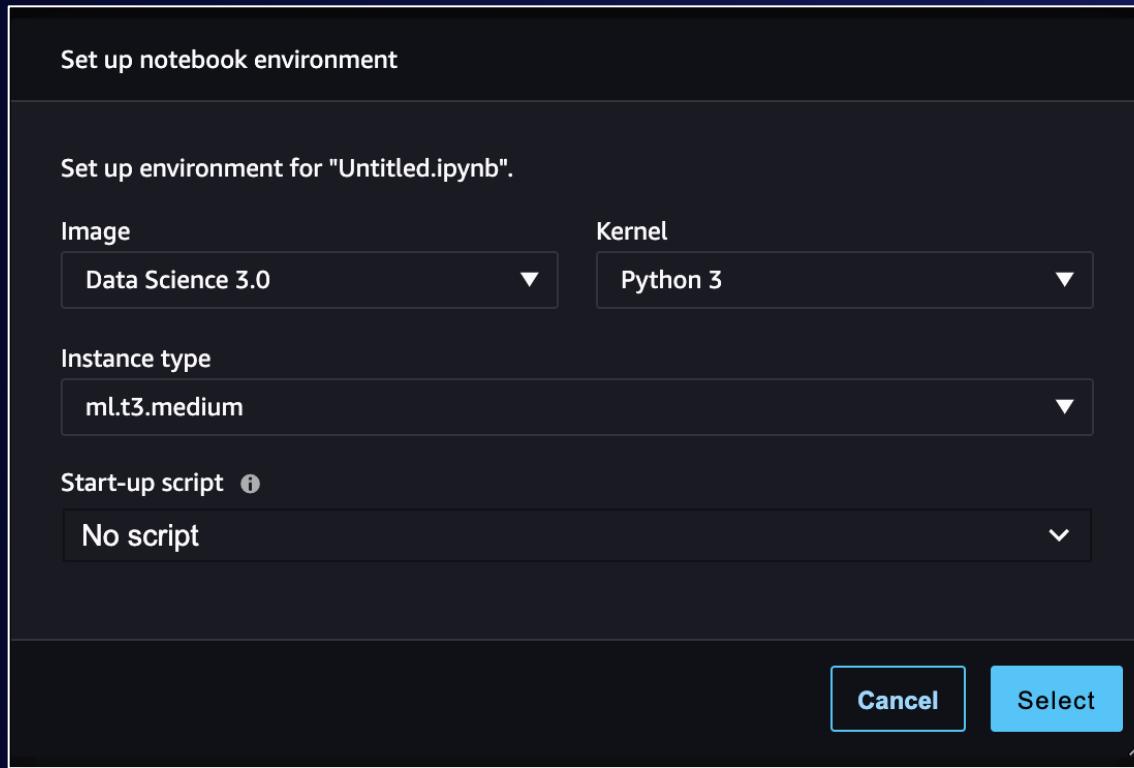
ml.t3.medium

Start-up script ⓘ

No script

Cancel

Select



Select Instance

Running notebook

ExampleNotebook.ipynb

Current instance type

2 vCPU + 4 GiB ml.t3.medium Fast Launch

If you change your instance, existing settings for this notebook will be lost, and installed packages will not be carried over.

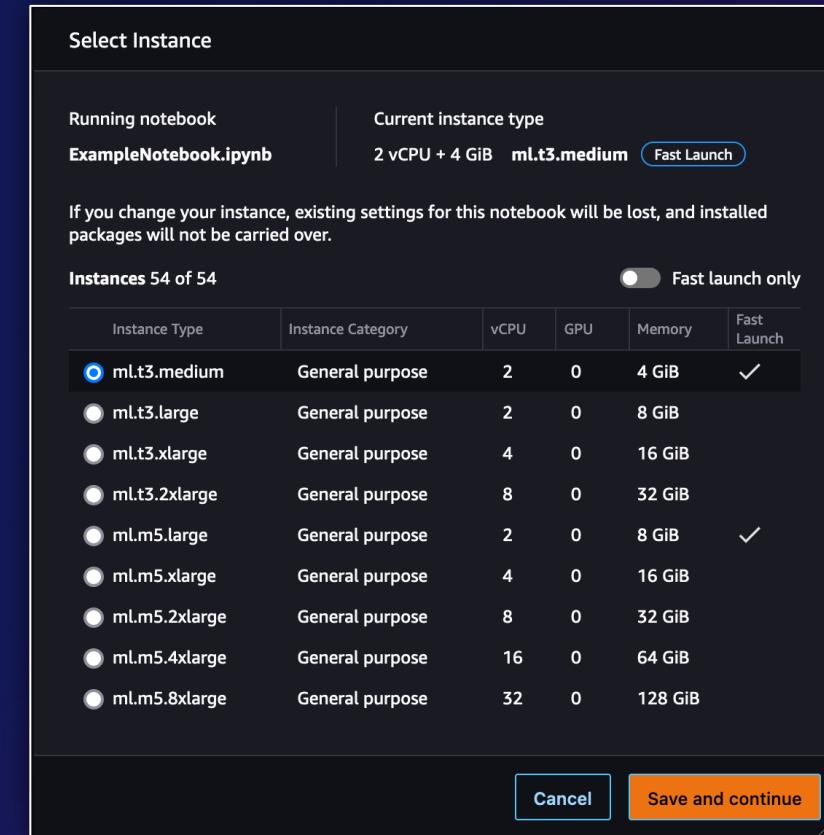
Instances 54 of 54

Fast launch only

Instance Type	Instance Category	vCPU	GPU	Memory	Fast Launch
ml.t3.medium	General purpose	2	0	4 GiB	✓
ml.t3.large	General purpose	2	0	8 GiB	
ml.t3.xlarge	General purpose	4	0	16 GiB	
ml.t3.2xlarge	General purpose	8	0	32 GiB	
ml.m5.large	General purpose	2	0	8 GiB	✓
ml.m5.xlarge	General purpose	4	0	16 GiB	
ml.m5.2xlarge	General purpose	8	0	32 GiB	
ml.m5.4xlarge	General purpose	16	0	64 GiB	
ml.m5.8xlarge	General purpose	32	0	128 GiB	

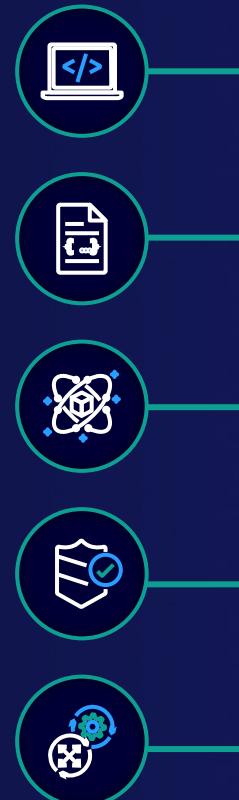
Cancel

Save and continue



# Amazon SageMaker Processing Job

**Managed solution for  
data processing and  
model evaluation jobs**



Achieve distributed processing for clusters

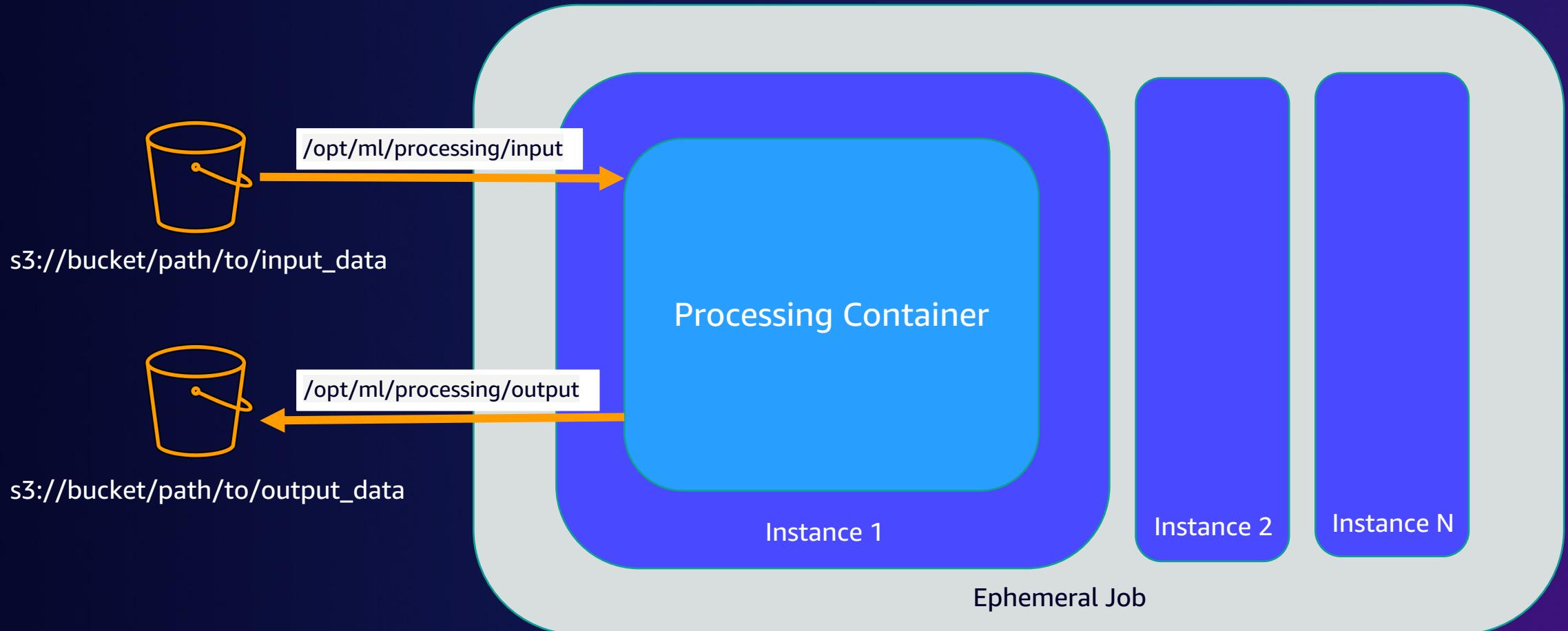
Bring your own script for feature engineering

Use SageMaker's built-in containers or bring your own

Leverage SageMaker's security and compliance features

Your resources are created, configured, and terminated automatically

# SageMaker Processing Job



# SageMaker Processing Job

```
import boto3
import sagemaker
from sagemaker import get_execution_role
from sagemaker.sklearn.preprocessing import SKLearnProcessor

region = boto3.session.Session().region_name

role = get_execution_role()
sklearn_processor = SKLearnProcessor(
    framework_version="0.20.0", role=role, instance_type="ml.m5.xlarge", instance_count=1
)
```

```
from sagemaker.processing import ProcessingInput, ProcessingOutput

sklearn_processor.run(
    code="preprocessing.py",
    # arguments = ['arg1', 'arg2'],
    inputs=[ProcessingInput(source="dataset.csv", destination="/opt/ml/processing/input")],
    outputs=[
        ProcessingOutput(source="/opt/ml/processing/output/train"),
        ProcessingOutput(source="/opt/ml/processing/output/validation"),
        ProcessingOutput(source="/opt/ml/processing/output/test"),
    ],
)
```

# Amazon SageMaker Data Wrangler

**No-code data preparation**



**Single visual interface for common data prep techniques**

**Select data from multiple sources**

**300+ built-in transformations to prepare data without writing code**

# Visual Workflow Interface

Import Data Flow

Create job

1 Select destination nodes

Job name: dataprep-flow-remapped-2022-10-04T10-13-11

1. S3: Test15

Unselect all

Output KMS key ⓘ

Optional

Trained parameters 0 View all

Refit ⓘ

Validation complete 0 errors Done

S3: shipment.csv

Source - Sampled → Data types → Transform → Join → Transform

Transform: shipment.csv

WB\_SHP

S3: waybill.csv

Source - Sampled → Data types → Transform → Join → Transform

Transform: waybill.csv

1st Transform: shipment.csv

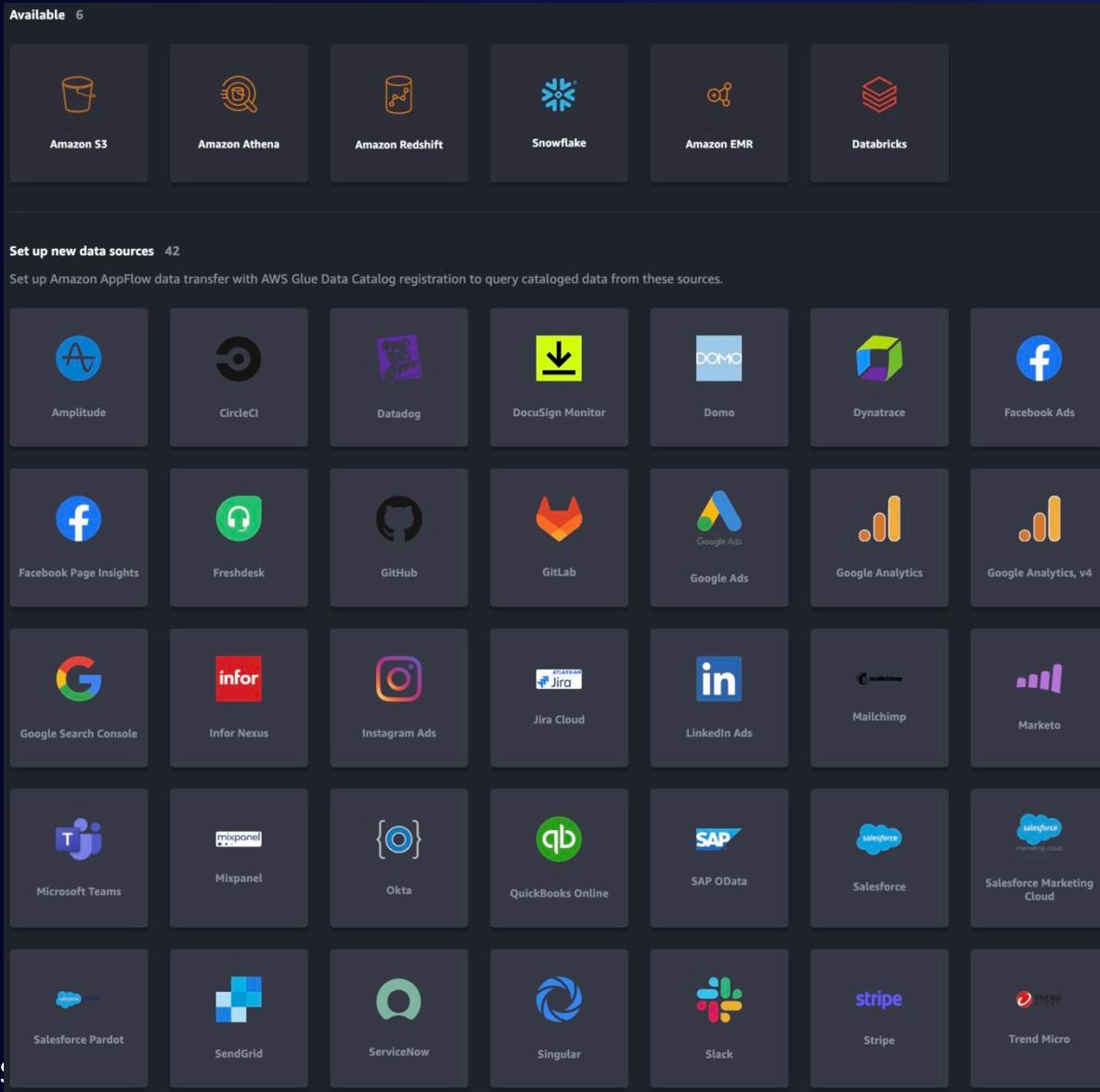
Join SHP\_WB\_CMS

Transform: 1st J

Cancel Next, 2. Configure job

```
graph LR; S1[Source - Sampled] --> DT1[Data types]; S2[Source - Sampled] --> DT2[Data types]; DT1 --> T1[Transform]; DT2 --> T2[Transform]; T1 --> J1{Join}; T2 --> J1; J1 --> T3[Transform]; J2{Join} --> T4[Transform]; T3 --> T4;
```

# Quickly connect and import data



- **Connect to multiple data sources**
  - Amazon S3, Amazon Athena, Amazon Redshift, Snowflake, Databricks
  - Salesforce Customer Data Platform (Preview)
  - Amazon EMR
  - Support 40+ SaaS data sources
- **Support for common file formats:**
  - CSV, Parquet, JSON and JSONL, ORC, Database tables
- **Explore data with S3 browser or SQL explorer, and sample data.**
  - Sampling strategies: top-k, random, stratified

# Easily prepare data for ML with 300+ built-in transforms

- Common and ML transforms
- Time Series, Train/Test split, Dimensionality Reduction, Multi-collinearity, PCA...

The screenshot displays the AWS Data Wrangler interface for adding data transforms. It consists of two main panels: a left panel titled 'ADD TRANSFORM' and a right panel titled 'RESULTS'.

**Left Panel (ADD TRANSFORM):**

- Search:** A search bar at the top labeled 'Search transforms'.
- CUSTOM Section:**
  - Custom formula:** Define a new column using a Spark SQL expression to query data in the current dataframe.
  - Custom transform:** Use Pyspark, Pandas, or Pyspark (SQL) to define custom transformations.
- STANDARD Section:**
  - Balance data:** Balance the data for binary classification problems using random oversampling, random undersampling or SMOTE.
  - Dimensionality Reduction:** For the top K principal components, trains a model to project vectors to a lower dimensional space.
  - Encode categorical:** Convert categorical variables to numeric or vector representations.
  - Featurize date/time:** Encode date/time values to numeric and vector representations.
  - Featurize text:** Generate vector representations from natural language text.
  - Format string:** Clean and prepare strings using standard string formatting operations.

**Right Panel (RESULTS):**

- Search Bar:** A search bar at the top with the text 'time'.
- RESULTS Section:**
  - Time Series:** Transformers to preprocess and manipulate time series.
  - Featurize date/time:** Encode date/time values to numeric and vector representations.
  - Encode categorical:** Convert categorical variables to numeric or vector representations.
  - Related: time**
  - Featurize text:** Generate vector representations from natural language text.
  - Related: time**
  - Handle missing:** Replace, drop, or add indicators for missing values.
  - Related: time**
  - Handle outliers:** Remove or replace outlier numeric and categorical values.
  - Related: time**
  - Parse column as type:** Cast a column to a new data type.
  - Related: time**
  - Search and edit:** Find, replace, split, and otherwise transform input string values using search and edit functions.
  - Related: time**

# Custom transforms & code snippets

## Panda, Spark SQL, PySpark

CUSTOM TRANSFORM

Use Pyspark, Pandas, or Pyspark (SQL) to define custom transformations. [Learn more.](#)

Python (PySpark)

```
1 from pyspark.sql.functions import col, round
2 df = df.withColumn('duration',
3     round((col("tpep_dropoff_datetime").cast("long")-
4         col("tpep_pickup_datetime").cast("long"))/60,
5     2))
6 df = df.drop("tpep_dropoff_datetime")
```

Clear

Preview Add

## Common code snippet library

CUSTOM TRANSFORM

Use Pyspark, Pandas, or Pyspark (SQL) to define custom transformations. [Learn more.](#)

Name

Optional

Python (PySpark)

▼ Search example snippets

Search for a sample snippet [i](#)

Select...

- Uniform sampling
- Stratified sampling
- Get top k rows
- Get bottom k rows
- Rename a column
- Rename multiple columns
- Rename columns by the pattern
- Delete columns by name
- Delete several columns by prefix

CUSTOM TRANSFORM

Use Pyspark, Pandas, or Pyspark (SQL) to define custom transformations. [Learn more.](#)

Name

Optional

Python (PySpark)

▼ Search example snippets

Search for a sample snippet [i](#)

Stratified sampling

Description

Extract samples from each group (defined by the column value). Each group has its own sampling rate or the absolute number of rows to sample.

Code snippet

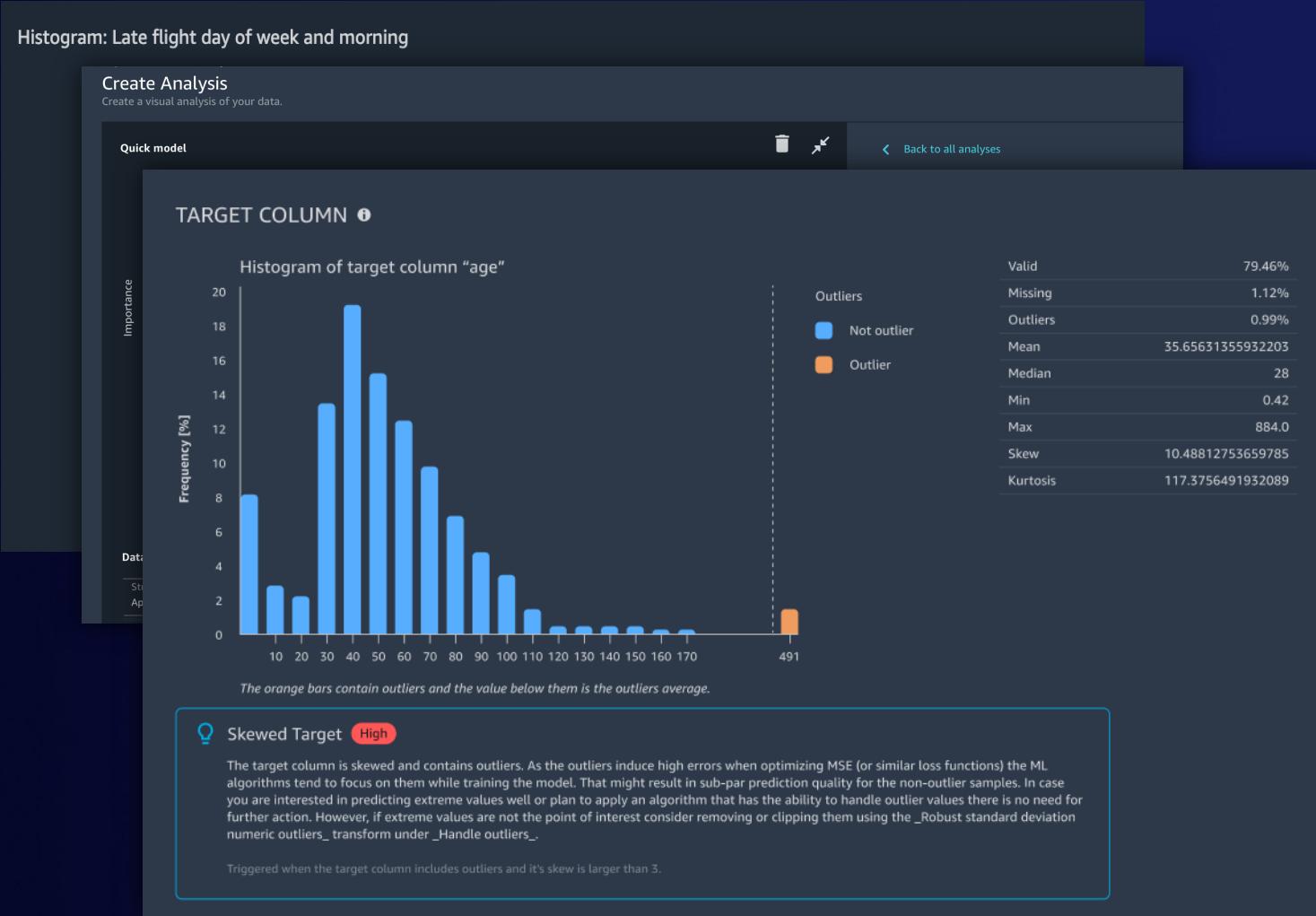
```
1 # Specify the column that defines the stratas (groups)
2 strata_column = "col_1"
3
4 # Specify the fraction of data to sample from each strata (group)
5 # in the example below class1 will be sampled at 0.1 rate and
6 # class2 will be sampled at 0.2 rate
7 # if class3 is present in the column, it will not be sampled
8 fractions = {"class1": 0.1, "class2": 0.2}
9
10 # Fix the randomness seed to ensure you pull the same samples
11 # everytime you re-run the transformer
12 df = df.sampleBy(strata_column, fractions, seed=0)
13
```

Your custom transform

```
1 # Table is available as variable `df`
2
```

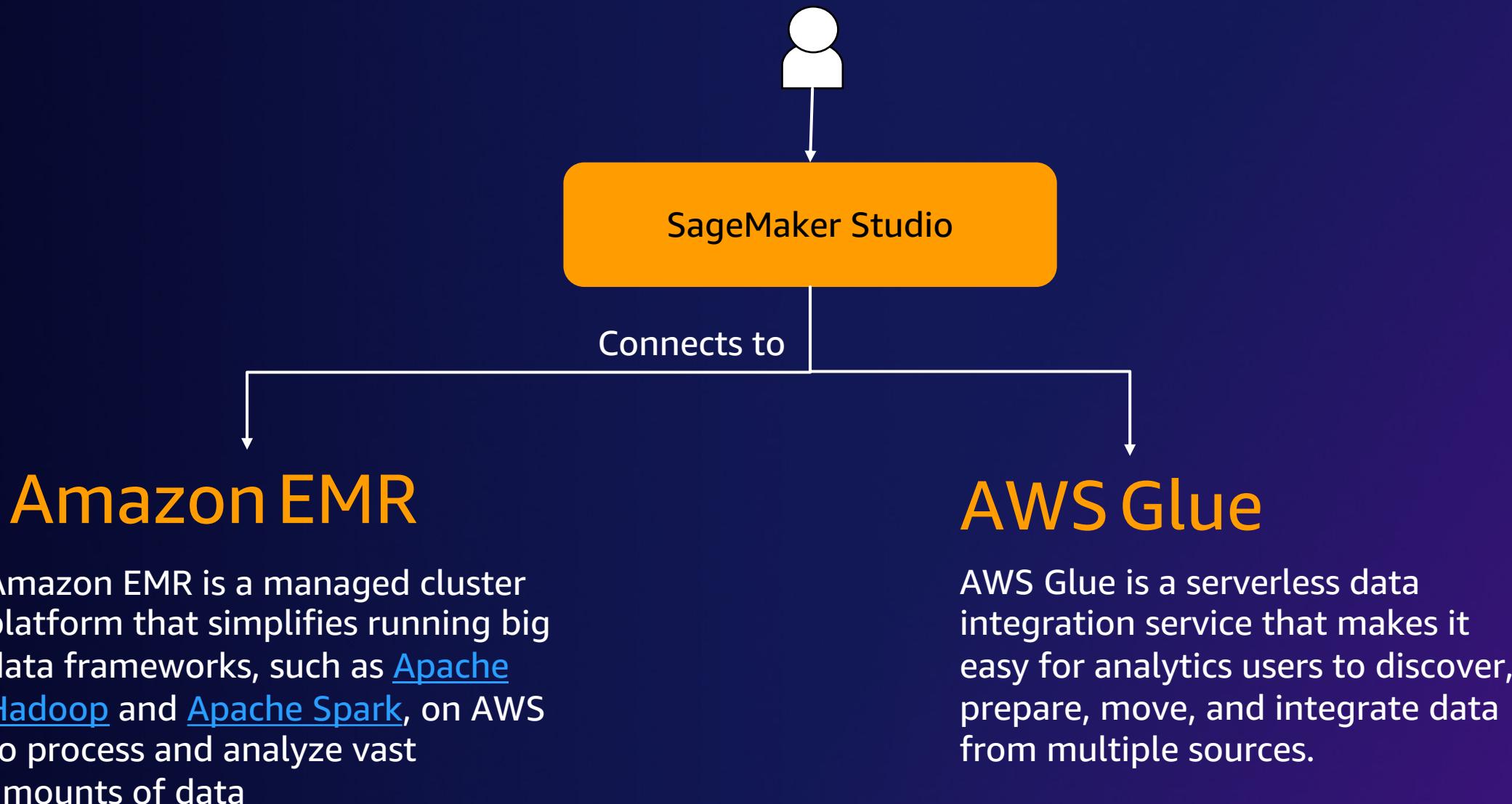


# Understand data visually: identify quality issues



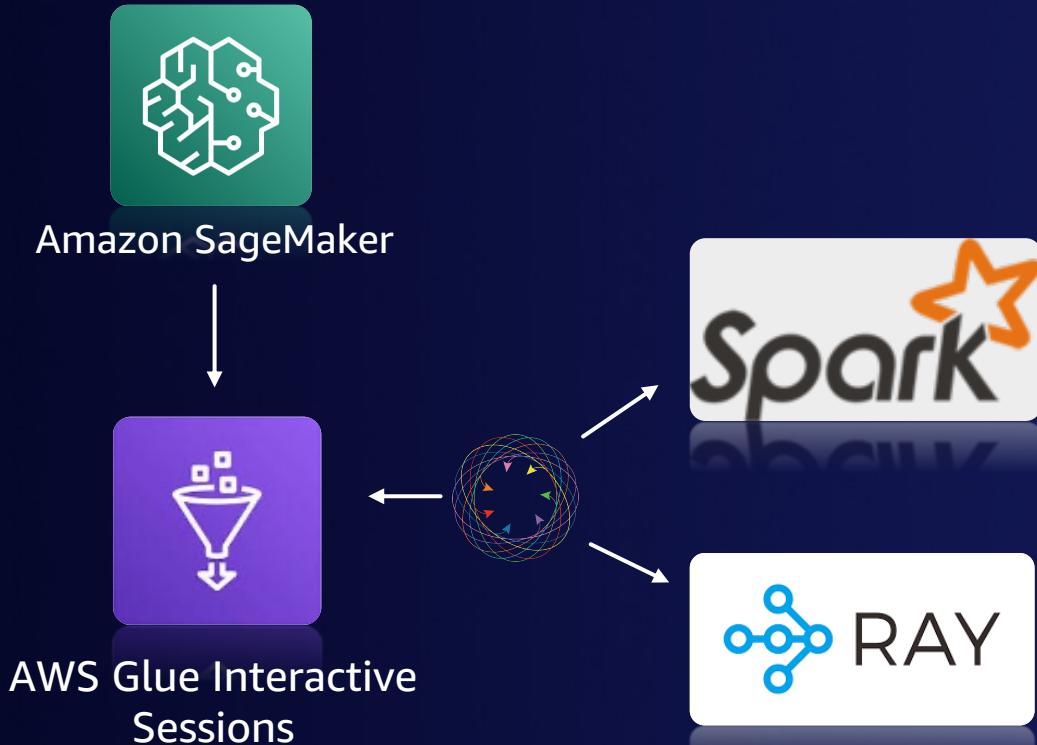
- Popular visualizations & ML powered analyses
- **Quick model** to predict model performance with prepared data
- Data insight report
  - Identifies issues: outliers and potential bias in features and target
  - Recommend transform to fix issue
  - Snapshot before and after

# Unifying Machine Learning and Data Analytics



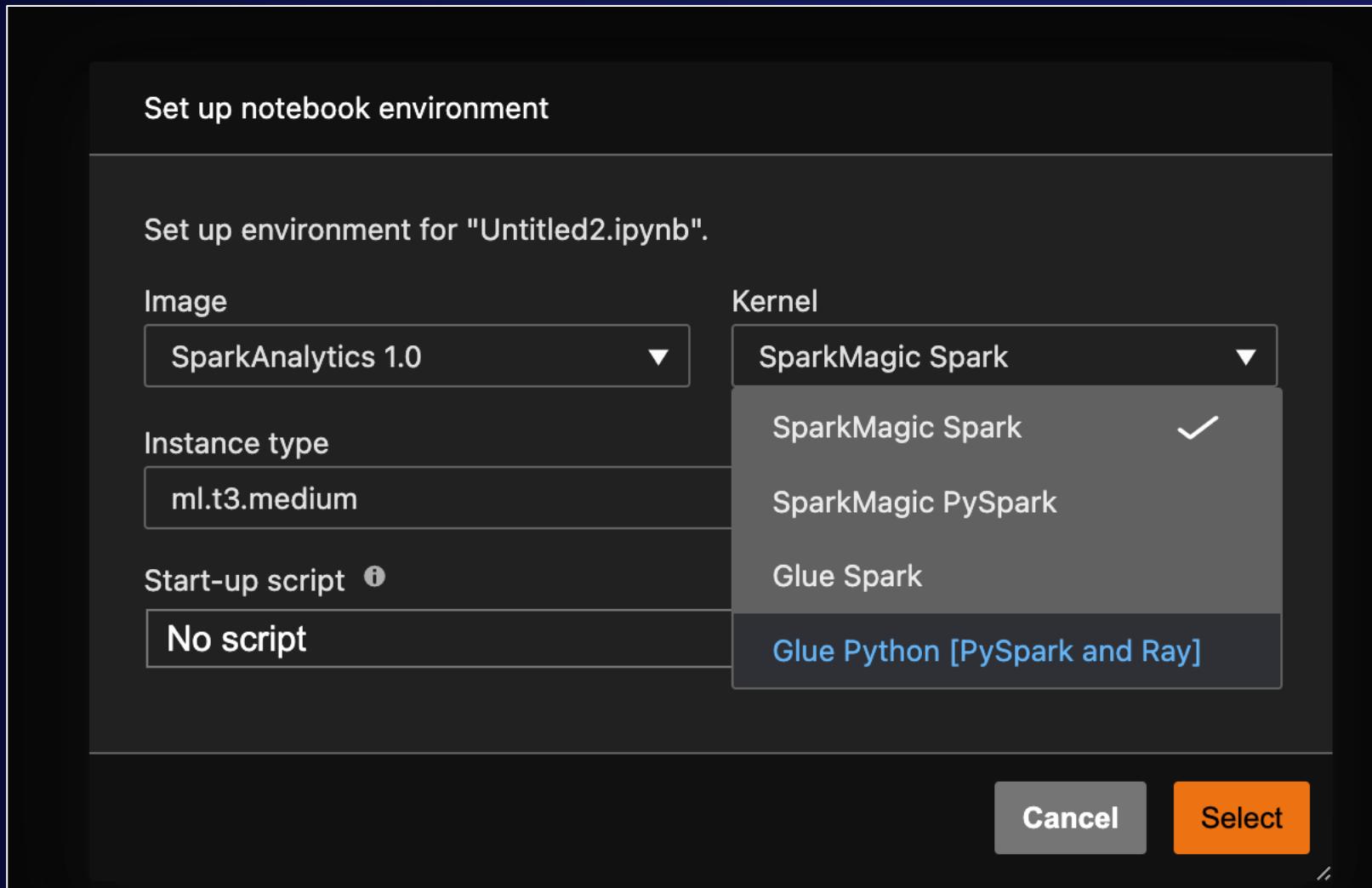
# SageMaker Studio + Glue Interactive Sessions

Scaling your data integration workloads using interactive Apache Spark and Ray

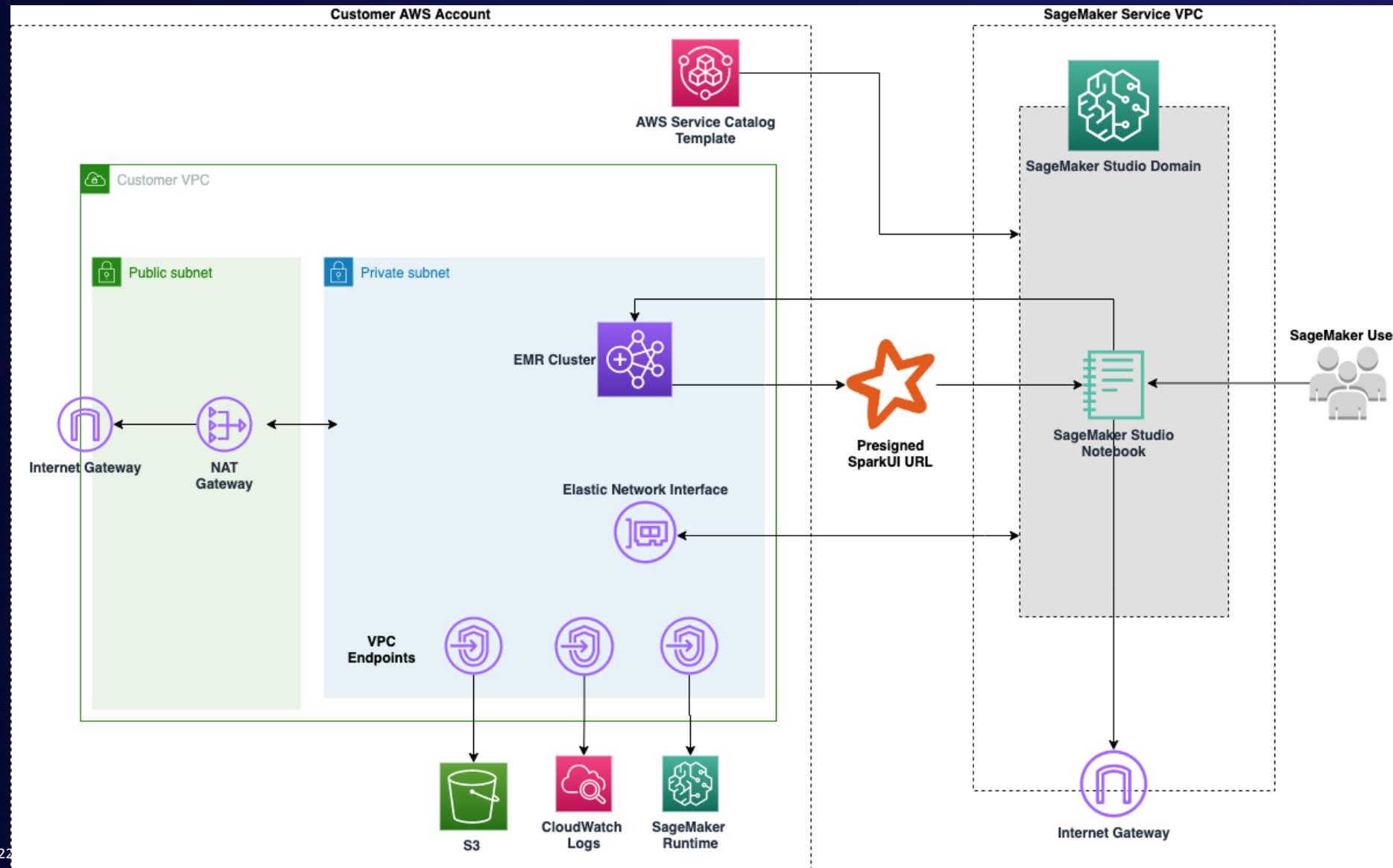


- From SageMaker Studio, access AWS Glue Interactive Sessions to access large, serverless, distributed clusters for Apache Spark and Ray
- Data engineers and ML practitioners can use Apache Spark or Ray to process large datasets with Python and popular Python libraries.
- Train distributed ML models on large datasets using algorithms from Apache Spark or Ray

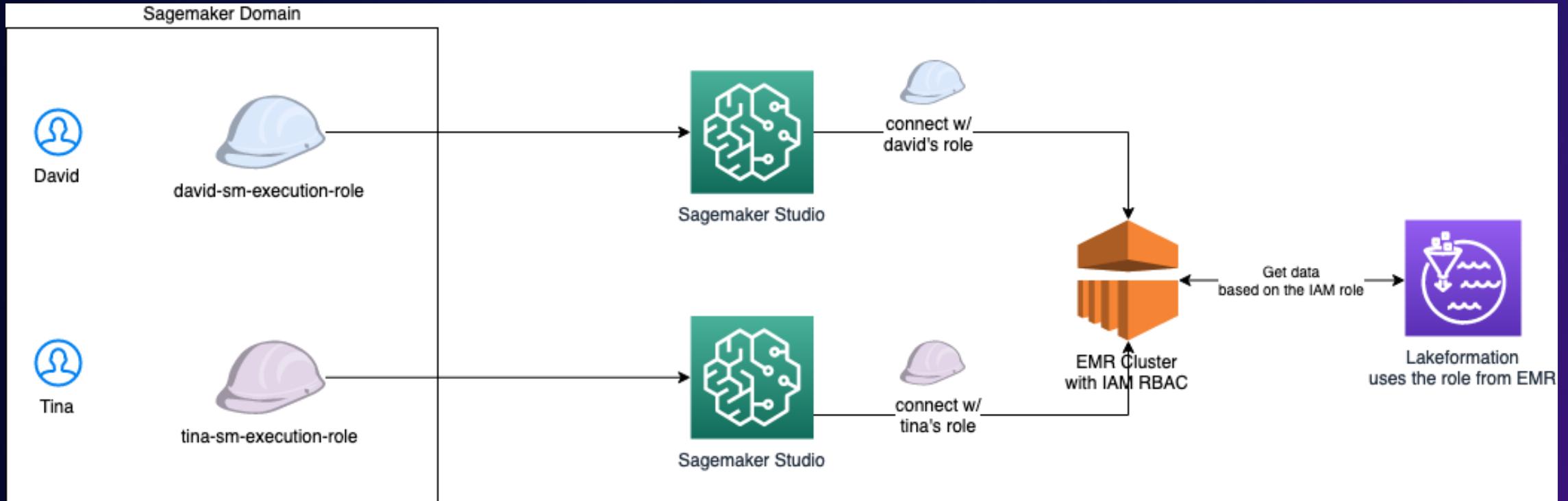
# Glue Kernels Built into SageMaker Studio



# SageMaker Studio + EMR Integration



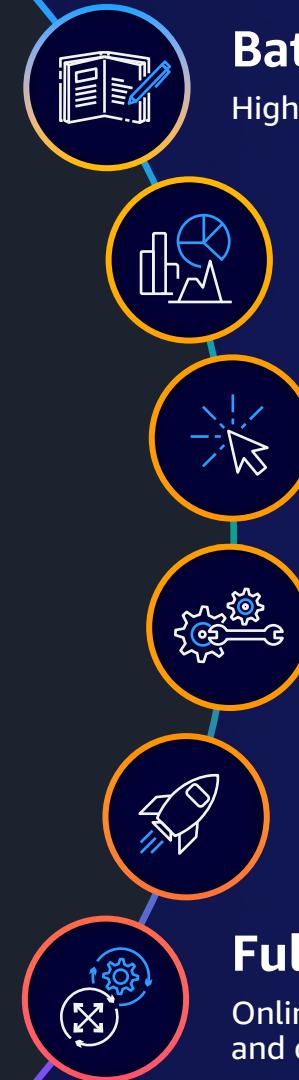
# Credential Passthrough to EMR Cluster



David and Tina only get access to data as defined by LakeFormation permissions. They connect to the same EMR cluster to access data from SageMaker Studio

# SageMaker Feature Store

**Accelerate machine learning with a feature store**



## Batch and streaming ingestion

High throughput writes for ingesting features

## Online and offline features

Online features for real-time prediction, and offline features for historical data for model training and batch prediction

## Feature metadata and data cataloging

Store metadata for features and leverage automatic data cataloging to easily query and extract feature data

## Feature discovery and reuse

Search for features to reuse, before starting new development

## Security and access control

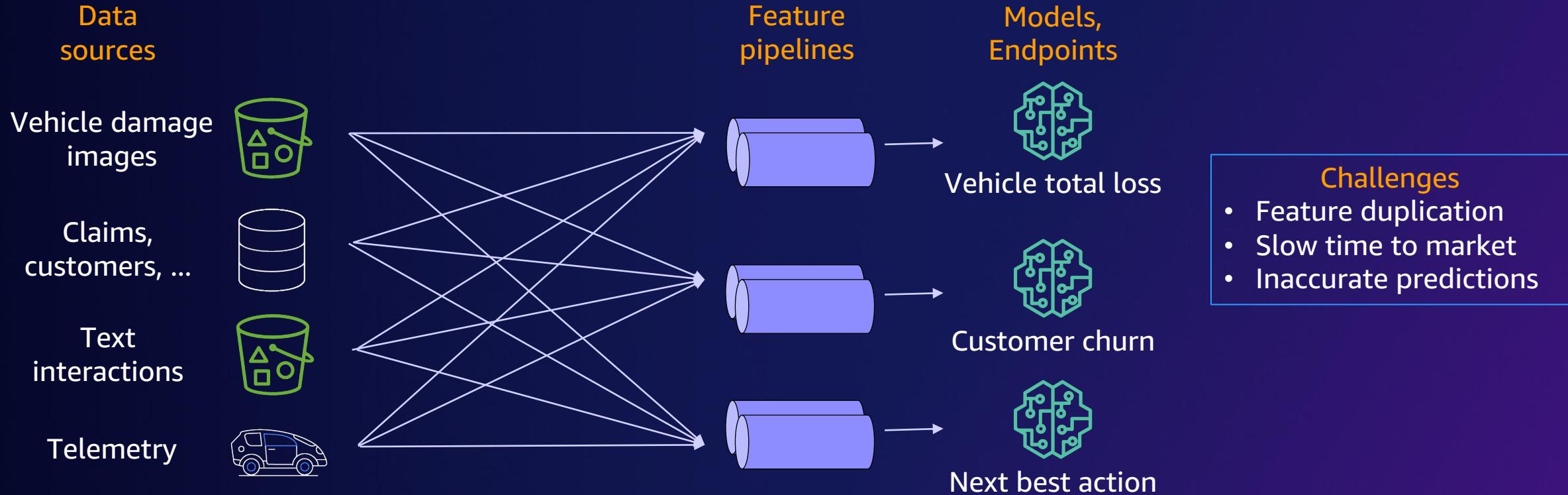
Access control for feature data and feature metadata, and support for encryption at rest, Amazon VPC, and AWS PrivateLink

## Fully managed

Online features cached in low-latency store; maintain consistency between online and offline store to avoid train-infer skew

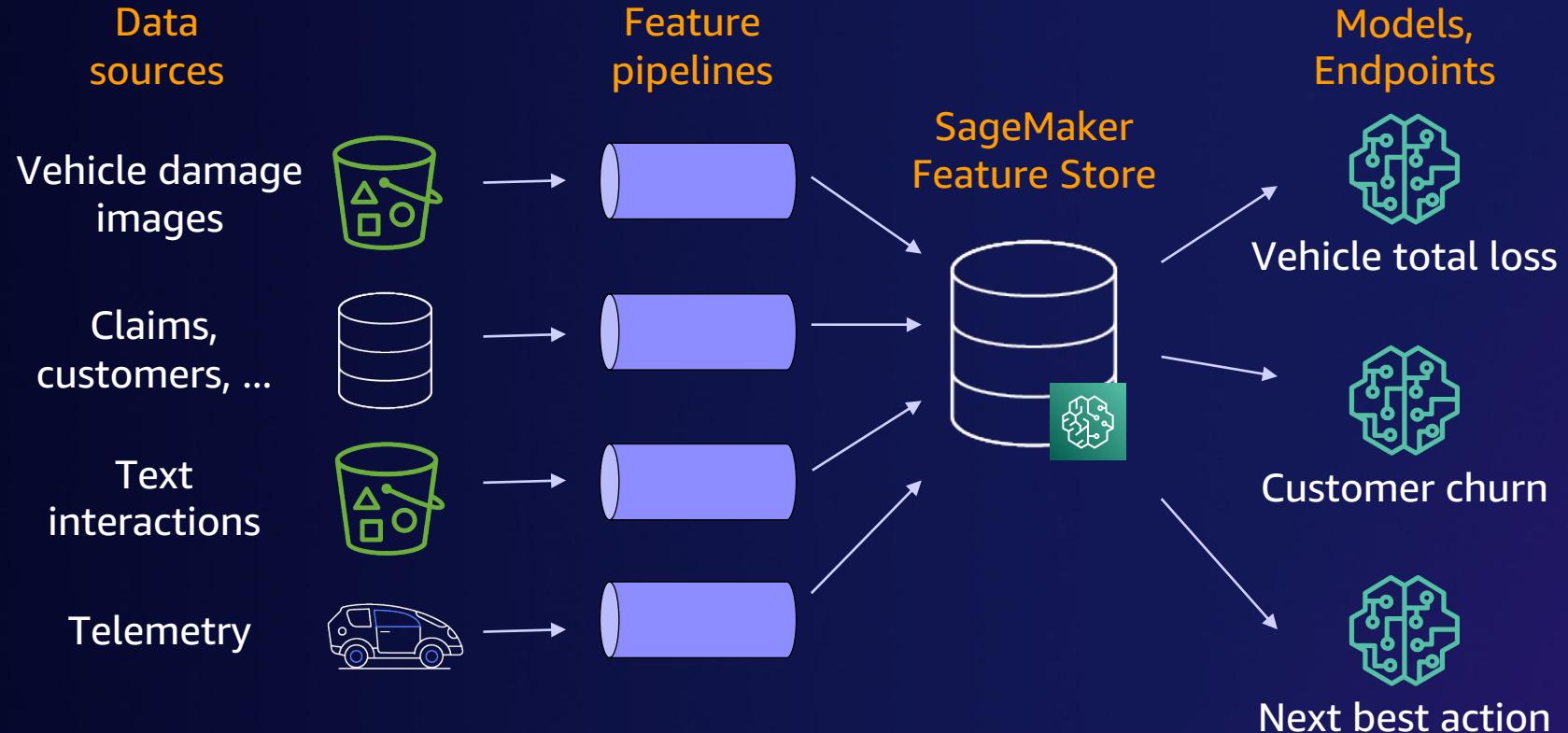
# Teams too often start from scratch

**Standalone feature engineering for each new model**



# With SageMaker Feature Store...

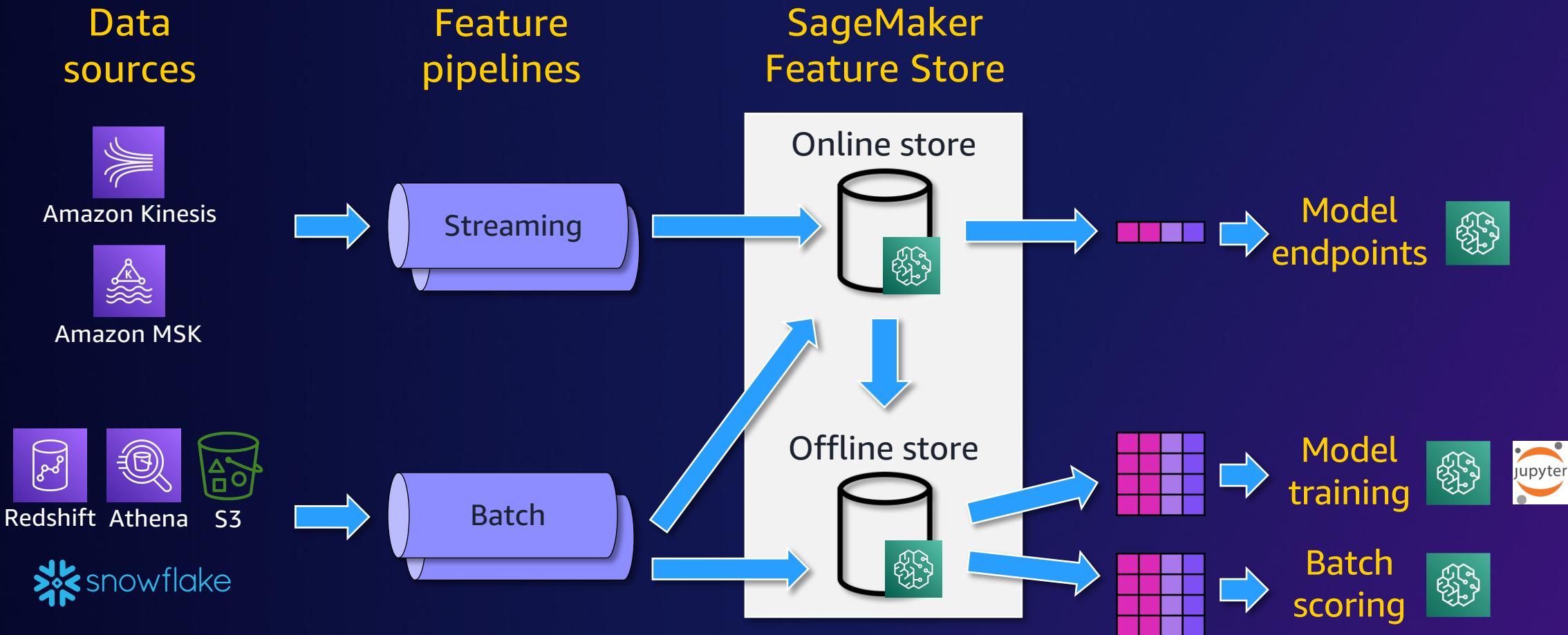
**Build features once, reuse them across teams and models**



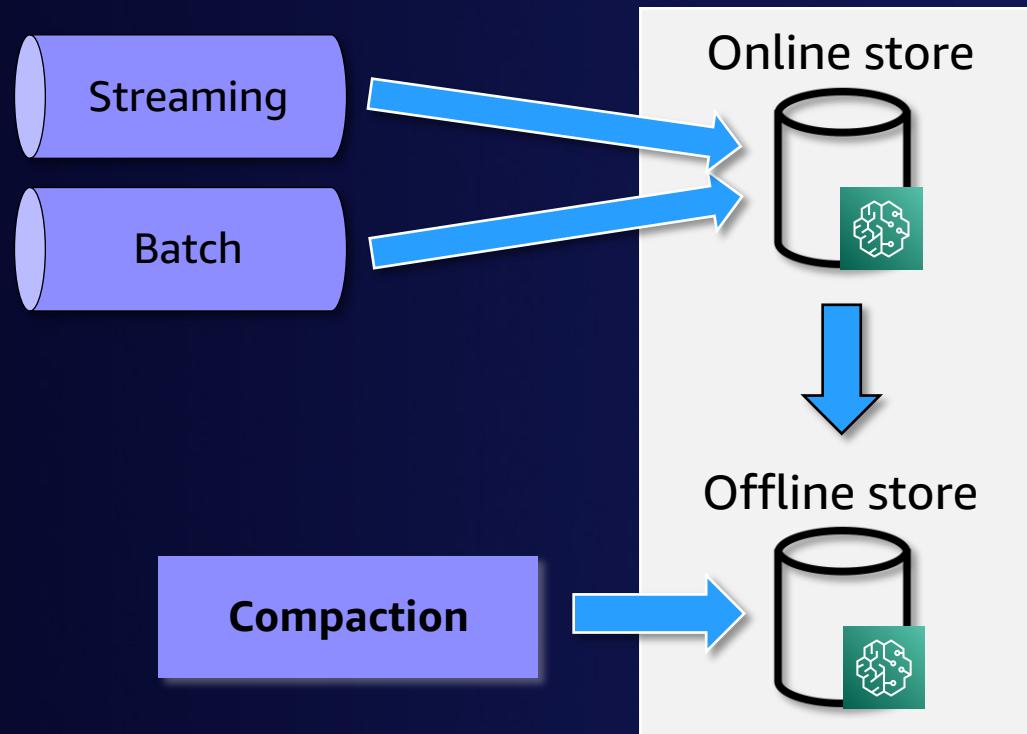
## Benefits

- Feature reuse
- Reproducible features
- Accurate training datasets
- Low latency inference
- Consistent features for training and inference

# Feature store in context



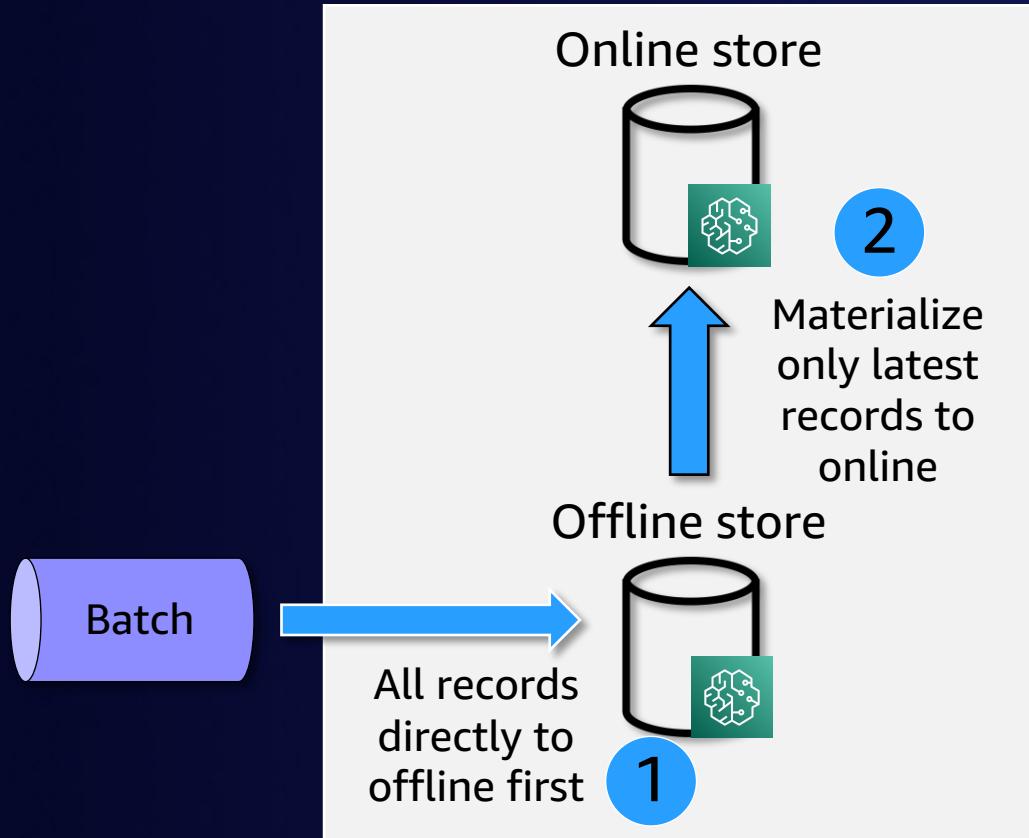
# Ingestion data flow pattern 1: Online to offline replication



## Pattern highlights

- Automatically keep offline store in sync with online store, avoiding train / inference skew
- Works well for workloads requiring streaming feature values
- Couple this pattern with use of offline store compaction and Iceberg table format for fast offline queries for training datasets

# Ingestion data flow pattern 2: Direct to offline store, and then materialize latest from offline to online



## Pattern highlights

- Avoids cost and time of online store when backfilling history, materializing only the latest features
- Directly ingests to offline store using Spark Connector
- Great for nightly feature refresh, should be default pattern for this use case
- Great for large scale feature backfill
- Populates offline store significantly faster than indirect ingestion through online
- Speeds offline queries for training data, minimizing need for separate compaction
- Lets you choose whether or not to materialize to online store for inference

# Demo + Q&A

# Lab 1 - Glue Interactive sessions with Spark and Ray

**<https://catalog.us-east-1.prod.workshops.aws/join>**

**Access Code: 3e17-0d3b81-dc**

# Model Training

- SageMaker training jobs
- SageMaker experiments
- Automatic model tuning
- Overview of distributed training
- Cost optimization options
  - Manage Spot instances
  - SageMaker Savings Plan
- Debugger

# Build ML models

**Fully managed  
shareable notebooks on  
Amazon EC2**



**Fully managed, sharable Jupyter notebooks**  
Run notebooks on elastic compute resources



**Built-in algorithms**  
15 built-in algorithms available in prebuilt container images



**Prebuilt solutions and open-source models**  
Over 150 popular open-source models



**AutoML**  
Automatically create ML models with full visibility



**Support for major frameworks and toolkits**  
Optimized for popular deep learning (DL) frameworks such as TensorFlow, PyTorch, Apache MXNet, and Hugging Face

# Amazon SageMaker has built-in algorithms or bring your own

## Classification

Linear Learner | XGBoost | KNN

## Computer vision

Image classification | Object detection |  
Semantic segmentation

## Topic modeling

LDA | NTM

## Working with text

BlazingText | Supervised | Unsupervised

## Regression

Linear Learner | XGBoost | KNN

## Clustering

KMeans

## Sequence translation

Seq2Seq

## Anomaly detection

Random cut forests | IP Insights

## Feature reduction

PCA

# Train ML models

**Fast and cost-effective  
ML model training**



**Experiment management and model tuning**  
Save weeks of effort by automatically tracking training runs and tuning hyperparameters



**Debug and profile training runs**  
Use real-time metrics to correct performance problems



**Distributed training**  
Complete distributed training up to 40% faster

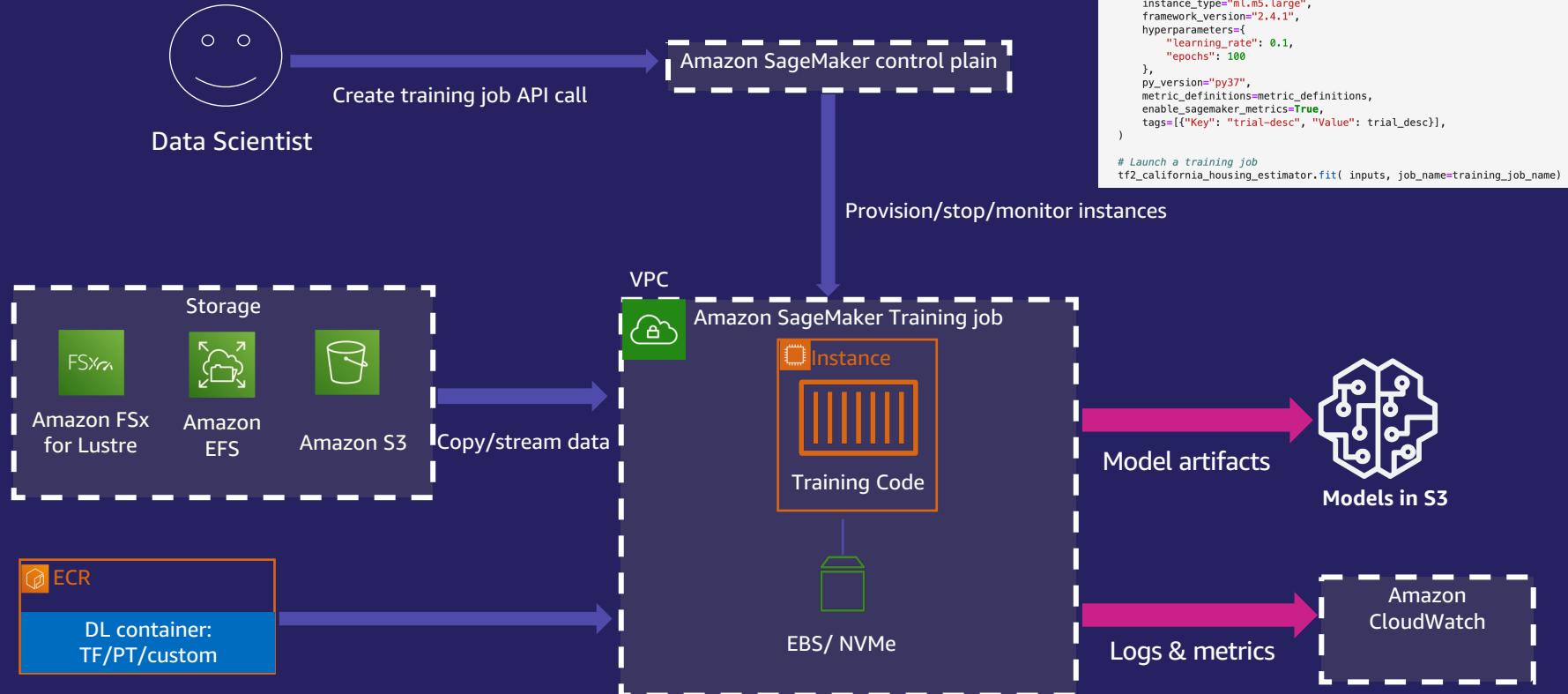


**Training compiler**  
Accelerate training times by up to 50% through more efficient use of GPUs



**Managed spot training**  
Reduce the costs of training by up to 90%

# Training on Amazon SageMaker



# Amazon SageMaker Experiments

Organize, track, and  
compare machine learning  
experiments



## Tracking at scale

Track parameters and metrics across experiments and users



## Custom organization

Organize experiments by teams, goals, and hypotheses



## Visualization

Easily visualize experiments and compare



## Metrics and logging

Log custom metrics using the Python SDK and APIs



## Fast iteration

Quickly go back and forth, and maintain high-quality

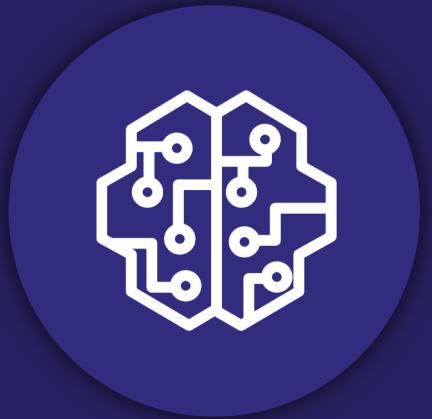
**New** Amazon SageMaker Experiments  
Manage ML experiments at scale



**Set up Experiments from  
any notebook client**



**Visually identify the best  
performing models**



**Ensure models are reliable  
and reproducible**

# SageMaker Experiments Concepts

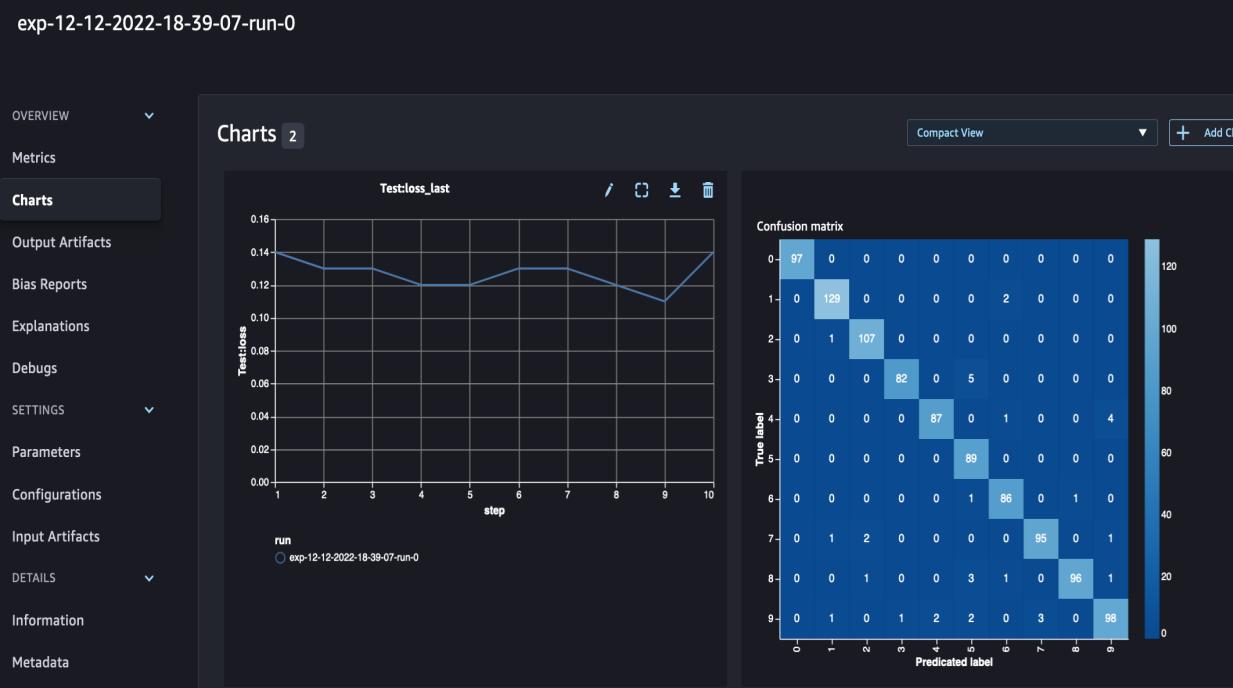
1. **Experiment:** An experiment is a collection of runs. When you initialize a run in your training loop, you include the name of the experiment that the run belongs to. Experiment names must be unique within your AWS account.
2. **Run:** A run consists of all the inputs, parameters, configurations, and results for one iteration of model training. Initialize an experiment run for tracking a training job with `Run()`.

## local-experiment-example-11-12-2022-21-34-24

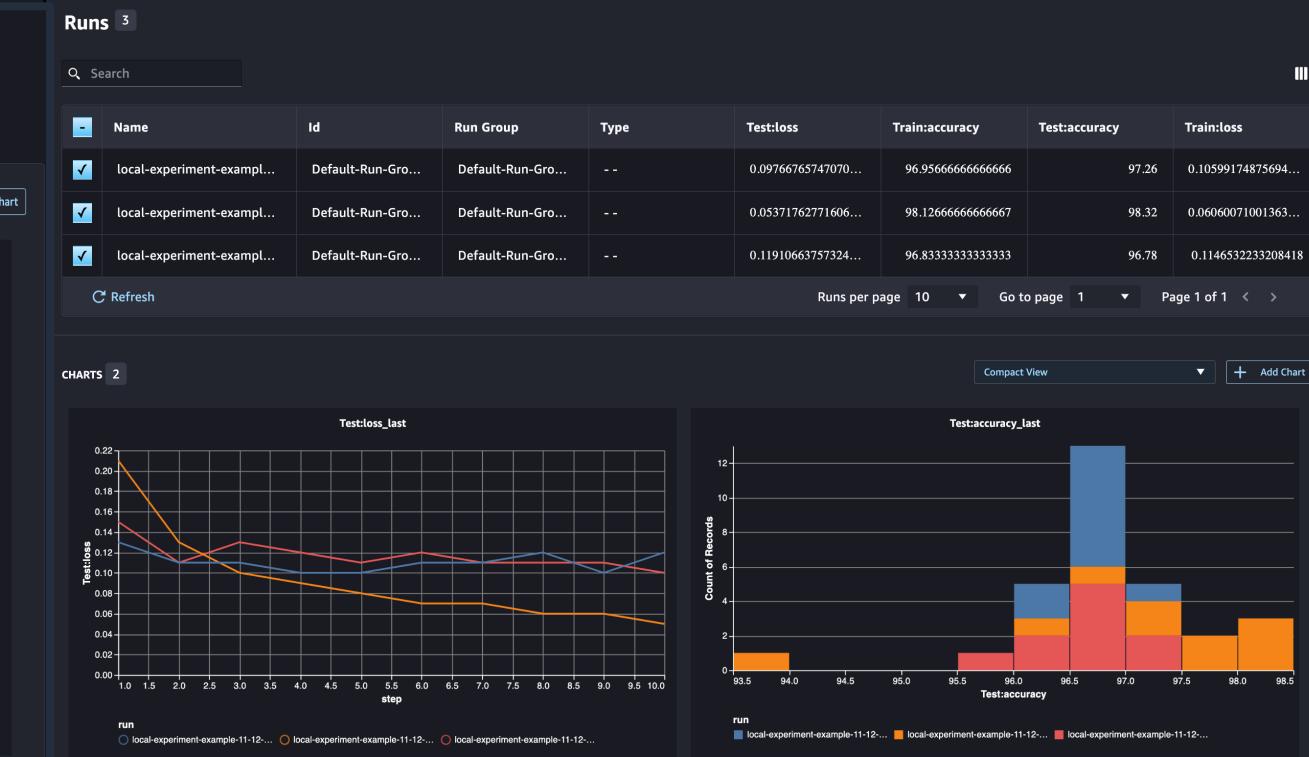
OVERVIEW
Runs
DETAILS
Information
Metadata

Runs 4					
	Name	Test:loss	Train:accuracy	Test:accuracy	Train:loss
<input checked="" type="checkbox"/>	local-experiment-example...	0.097667657470...	96.95666666666...	97.26	0.105991748756...
<input checked="" type="checkbox"/>	local-experiment-example...	0.053717627716...	98.12666666666...	98.32	0.060600710013...
<input checked="" type="checkbox"/>	local-experiment-example...	0.119106637573...	96.83333333333...	96.78	0.114653223320...
<input type="checkbox"/>	local-experiment-example...	0.141925717163...	95.465	95.64	0.148446399235...

Refresh Runs per page 10 Go to page 1 Page 1 of 1 < >



## Analyze and compare runs



# Amazon SageMaker Automatic Model Tuning

Automatically tune  
hyperparameters in  
your algorithms



## Tuning at scale

Adjust thousands of different combinations of algorithm parameters



## Automated

Uses ML to find the best parameters



## Faster

Eliminate days or weeks of tedious manual work



## Decision trees

Tree depth | Max leaf nodes | Gamma | Eta | Lambda | Alpha

## Neural networks

Number of layers | Hidden layer width | Learning rate | Embedding dimensions | Dropout

# Amazon SageMaker Automatic Model Tuning

## Hyperparameter Tuning



# Setting up hyper parameter tuning job

1. Pick hyperparameters and ranges

```
: hyperparameter_ranges = {'eta': ContinuousParameter(0, 1),  
                           'min_child_weight': ContinuousParameter(1, 10),  
                           'alpha': ContinuousParameter(0, 2),  
                           'max_depth': IntegerParameter(1, 10)}
```

2. Pick objective metric

```
: objective_metric_name = 'validation:auc'
```

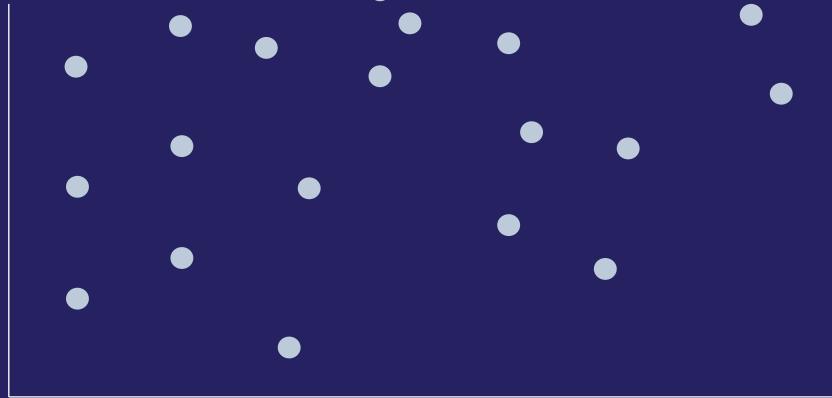
3. Pick job parameters

```
: tuner = HyperparameterTuner(xgb,  
                               objective_metric_name,  
                               hyperparameter_ranges,  
                               max_jobs=20,  
                               max_parallel_jobs=3)
```

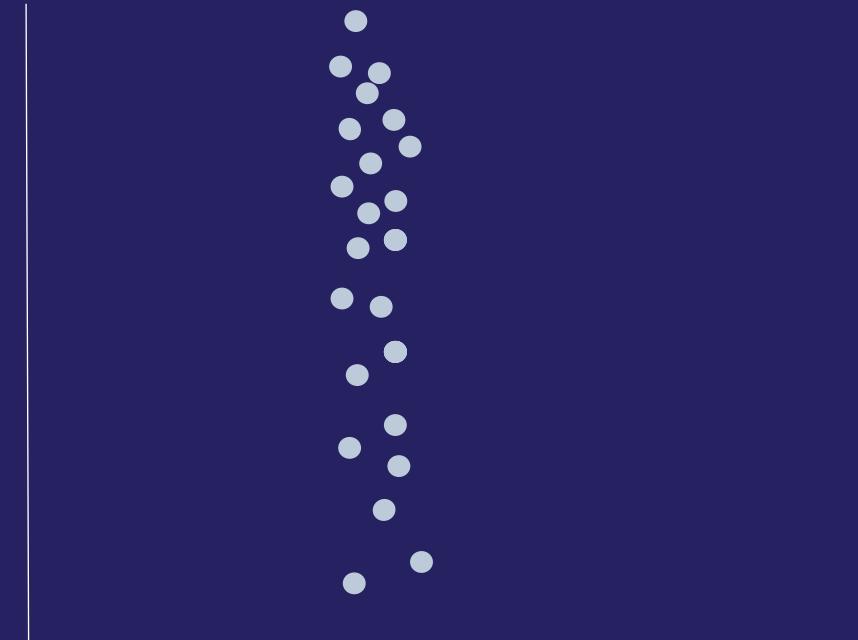
# Amazon SageMaker Automatic Model Tuning

**What if I need all my jobs tuned at the same time ?**

Bayesian Search



Random Search



# Amazon SageMaker Automatic Model Tuning

## Can I use hyperparameter tuning with my own model ?

1

Built-in  
Algorithms

2

Docker

3

Script Mode

Fully Customizable

# Amazon SageMaker Automatic Model Tuning

## Can I use hyperparameter tuning with my own model ?

### Setting the hyperparameters

```
In [5]: hyperparameters = dict(batch_size=32, data_augmentation=True, learning_rate=.0001,  
                           width_shift_range=.1, height_shift_range=.1, epochs=1)  
hyperparameters
```

```
Out[5]: {'batch_size': 32,  
         'data_augmentation': True,  
         'learning_rate': 0.0001,  
         'width_shift_range': 0.1,  
         'height_shift_range': 0.1,  
         'epochs': 1}
```

Docker



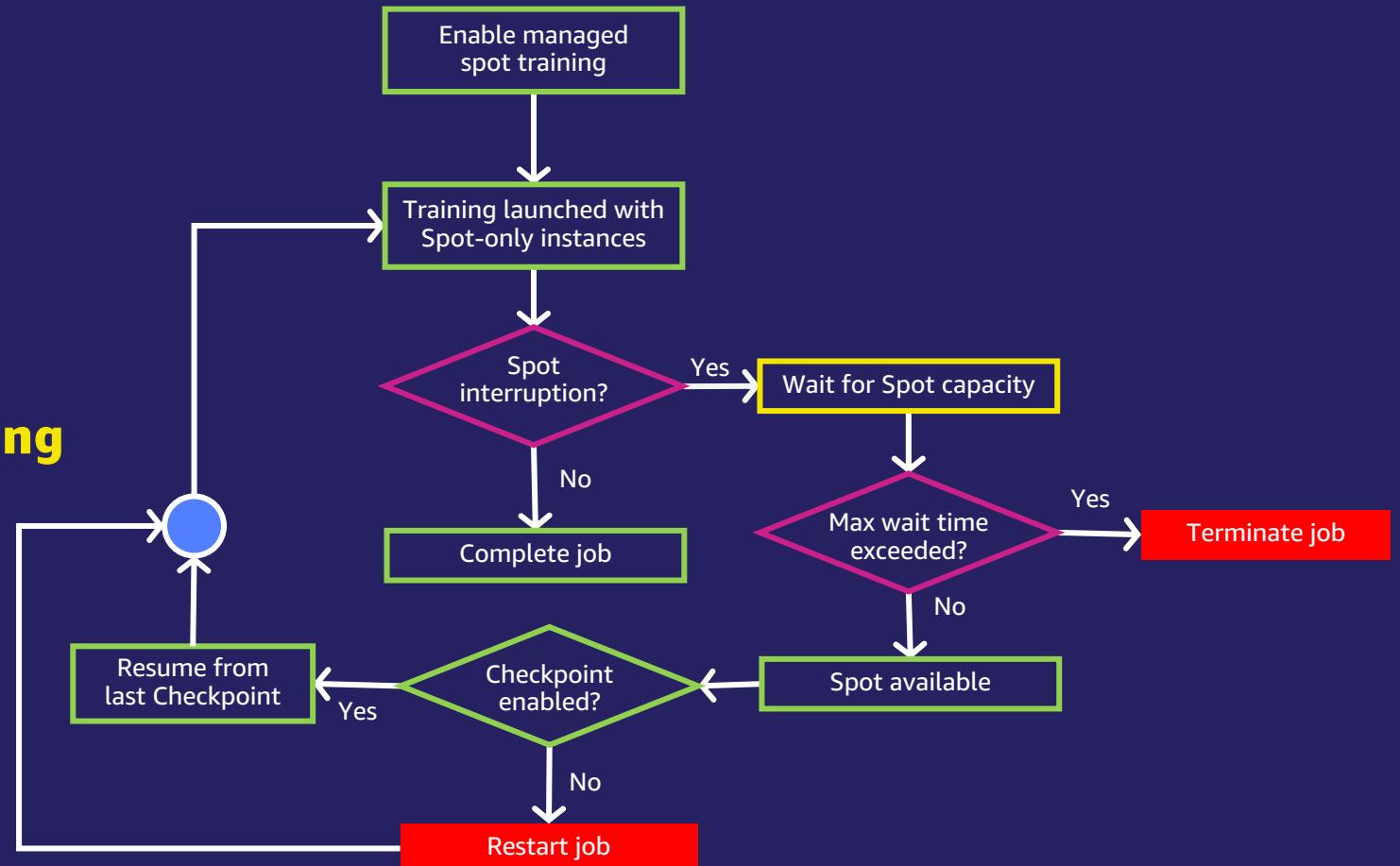
Script Mode



# Cost optimization options

# Managed Spot Training

Save up to 90% on model training costs



# Key considerations



## Training with only Spot

Interrupted jobs resume if checkpointed  
and if Spot instances become available;  
Jobs restart if not checkpointed\*

Works with Automated Model Tuning

Training jobs can run only with a single  
instance - type in a single - AZ

Does not integrate with Spot Fleet and  
Spot Block today



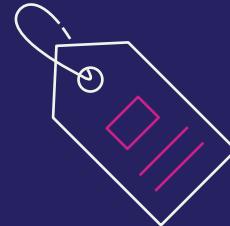
## Checkpoint

Built-in algorithms  
checkpoint automatically

For custom models,  
checkpointing should be enabled

Checkpoints are saved to S3

Models that don't checkpoint are  
subject to *MaxWaitTime* of 60 mins



## Pricing

View savings on AWS console or use  
DescribeTrainingJob API

Charged for the run duration before  
completion or termination;  
not charged for idle time, billing starts  
when instances are ready

Charged for data download time only  
once even if the job is interrupted  
multiple times

\* Checkpointing is a best practice and is highly recommended

# Debugger



## Generate ML models faster

Detect bottlenecks and issues during training in real-time and correct problems to deploy models faster, with a single, unified tool



## Optimize resources with no additional code

Monitor and profile system resources without code and get recommendations to optimize resources effectively



## Make ML training transparent

Get complete insights into the ML training process in real-time and offline

# Amazon SageMaker Debugger—How it works



# Distributed training

**The fastest and easiest way to train large deep learning models**



## Reduced training time

Reduce training time by 25% with synchronization across GPUs



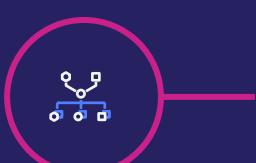
## Optimized for AWS

Achieve near-linear scaling efficiency with data parallelism designed for AWS



## Support for popular ML framework APIs

Re-use existing APIs such as Horovod without custom training code



## Automatic and efficient model partitioning

Avoid experimentation with automated model profiling and partitioning



## Minimal code change

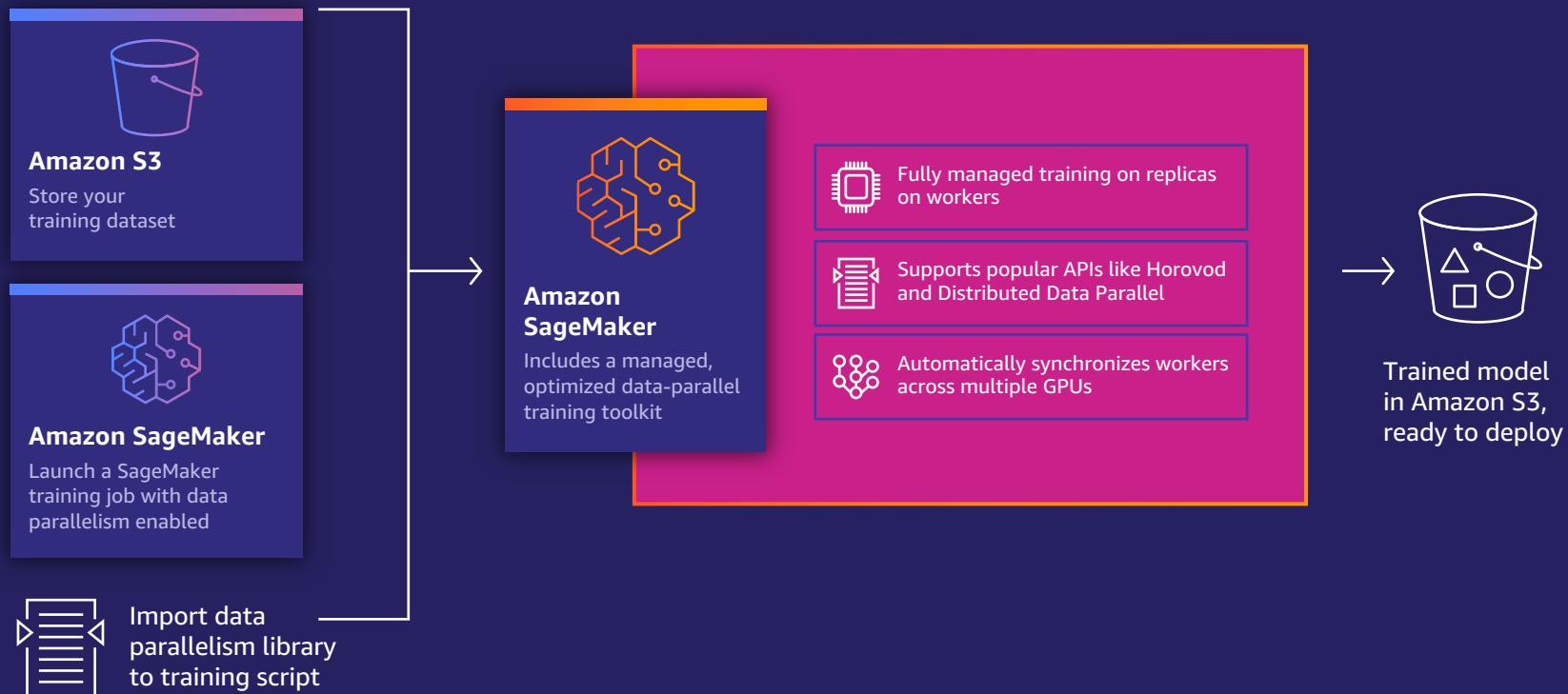
Implement model parallelism with fewer than 10 lines of code change



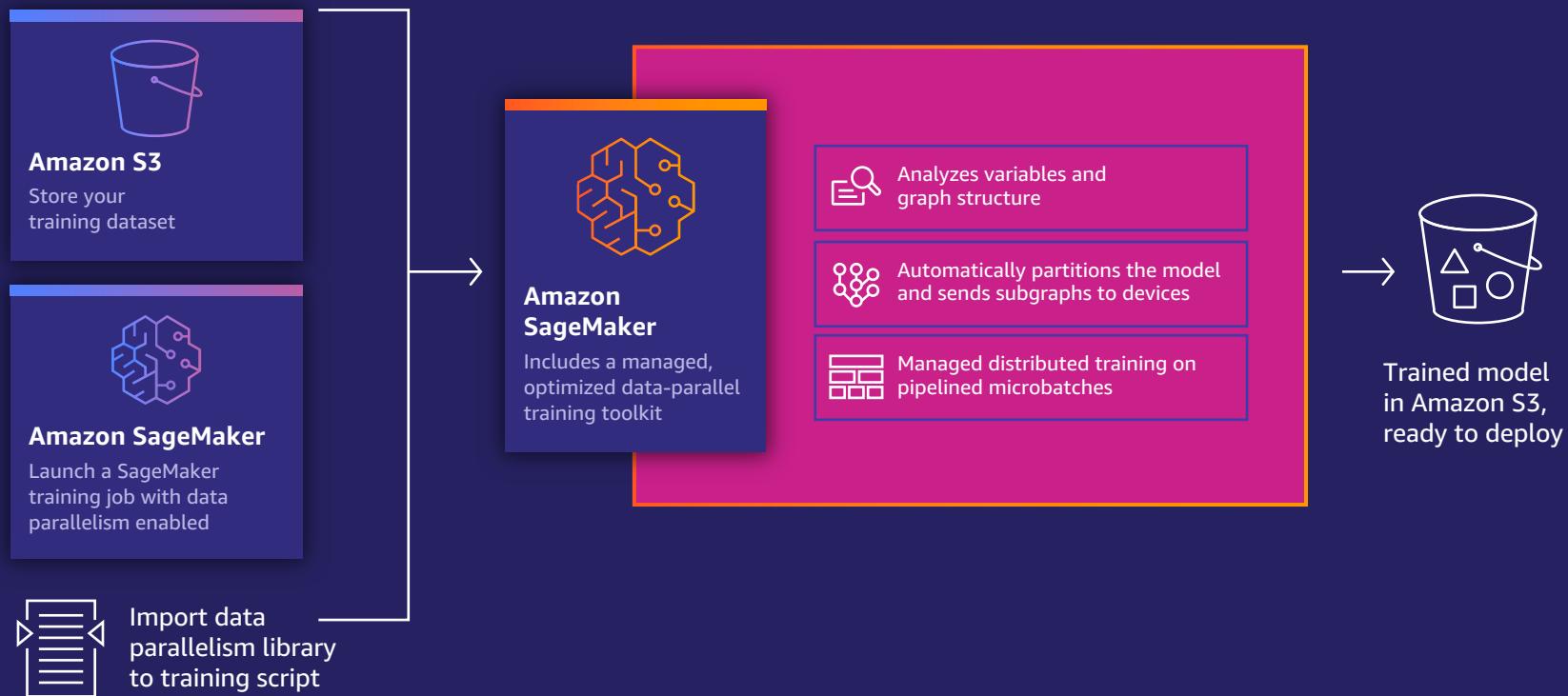
## Efficient pipelining

Maximize resource usage with pipelining of micro-batches that keeps all GPUs active

# How it works: Data parallelism library



# How it works: Model parallelism library



# Lab 2 – Model training and automatic model tuning

**<https://catalog.us-east-1.prod.workshops.aws/join>**

**Access Code: 3e17-0d3b81-dc**

# Model Deployment

- Please refer to deck provided in Day1-Part2

# Lab 3 – Model deployment

# Lab 3 – Model deployment

**<https://catalog.us-east-1.prod.workshops.aws/join>**

**Access Code: 3e17-0d3b81-dc**

# SageMaker MLOps & ML Platform Intro

**Please refer to deck provided  
Day1-Part3**

# Amazon SageMaker

## Next Steps

### Onboarding & Processing

- <https://docs.aws.amazon.com/sagemaker/latest/dg/gs-studio-onboard.html>
- <https://docs.aws.amazon.com/sagemaker/latest/dg/processing-job.html>

<https://github.com/aws/amazon-sagemaker-examples>

<https://sagemaker.readthedocs.io/en/stable/index.html>

### Training

- <https://docs.aws.amazon.com/sagemaker/latest/dg/train-model.html>
- <https://docs.aws.amazon.com/sagemaker/latest/dg/distribute-d-training.html>
- <https://aws.amazon.com/sagemaker/debugger>

### Deployment

- <https://docs.aws.amazon.com/sagemaker/latest/dg/realtimed-endpoints.html>
- <https://docs.aws.amazon.com/sagemaker/latest/dg/serverless-endpoints.html>
- <https://docs.aws.amazon.com/sagemaker/latest/dg/async-inference.html>
- <https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform.html>

# Survey!





# Thank you!