Amazon SageMaker

End to End Machine Learning

Today's Agenda

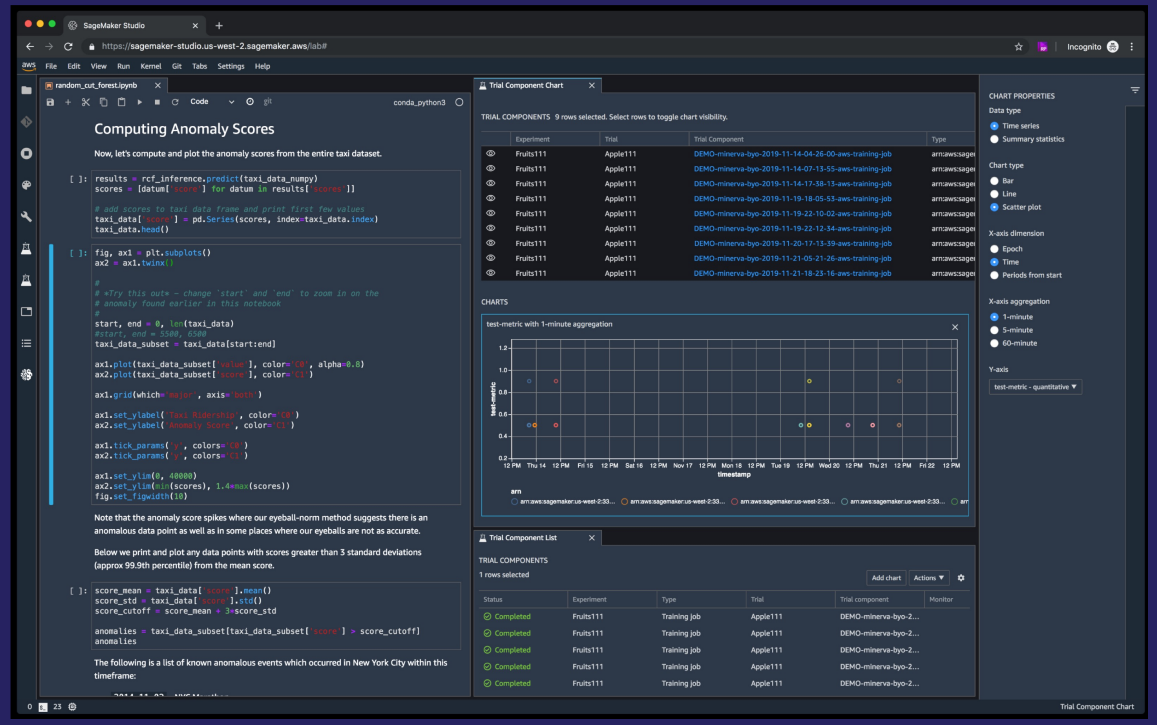
- Amazon SageMaker Studio
- Data Preparation for ML
- Model Training
- Model Deployment
- MLOPs Introduction

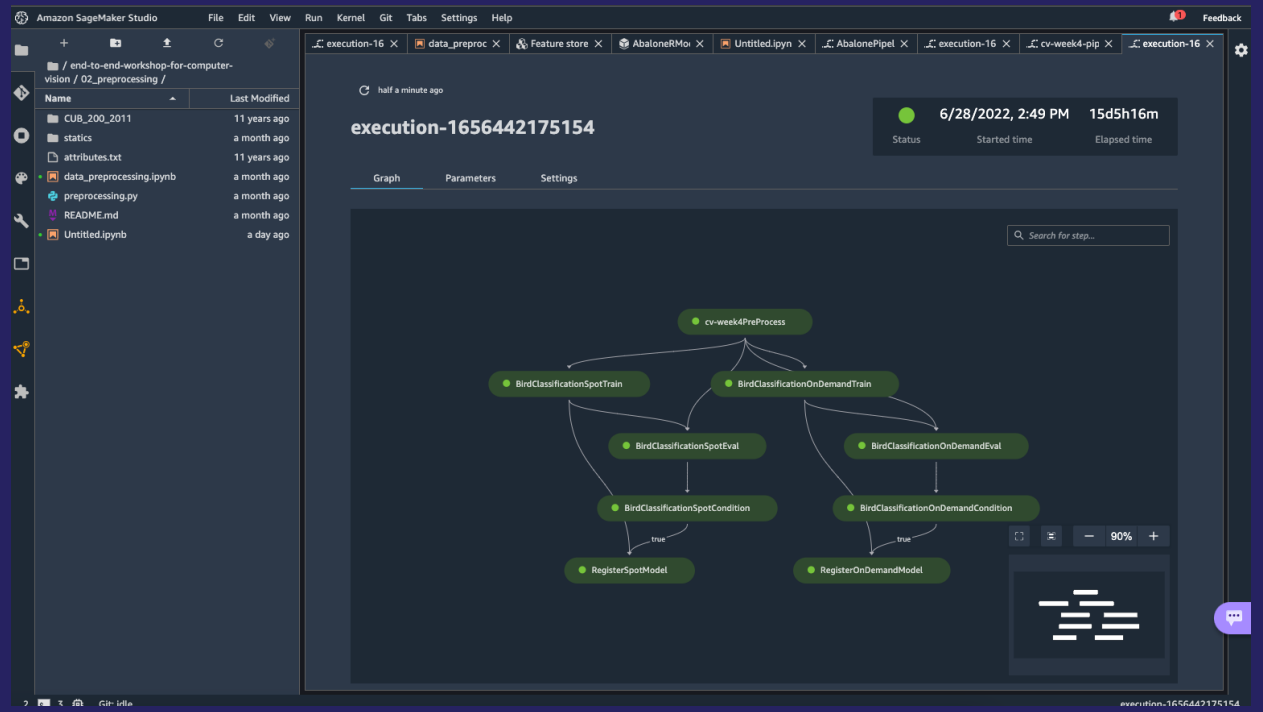
SageMaker Studio

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

SageMaker Studio

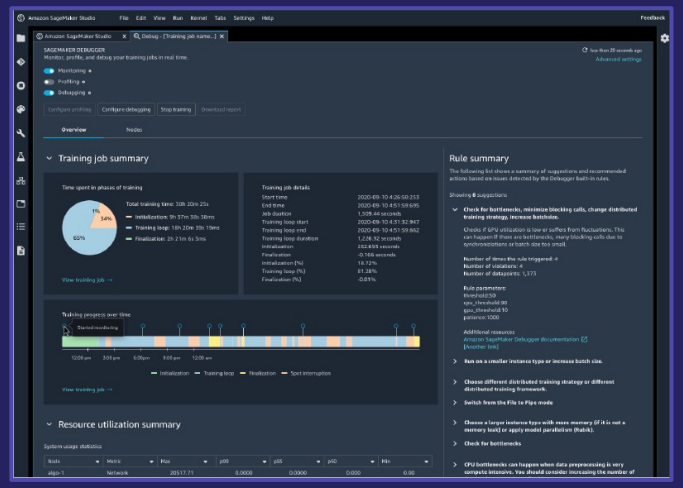
Provides end to end machine learning platform, increasing ML productivity by up to 10x





aws machine learning

Train Models Cost Effectively in SageMaker Studio Notebooks

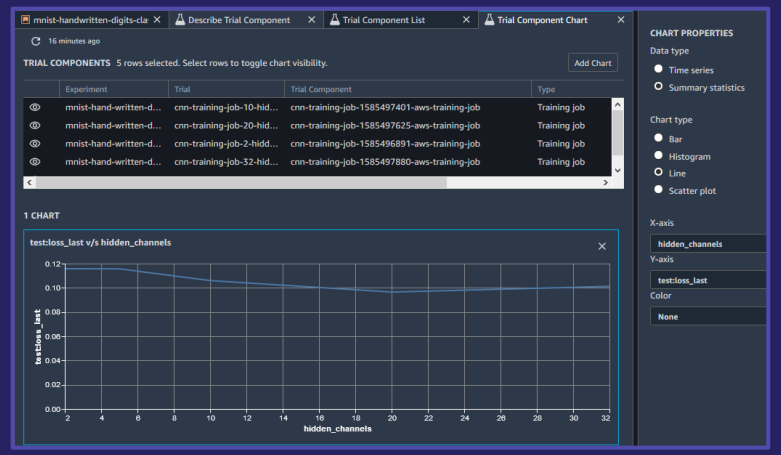


Experiment management and model tuning

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

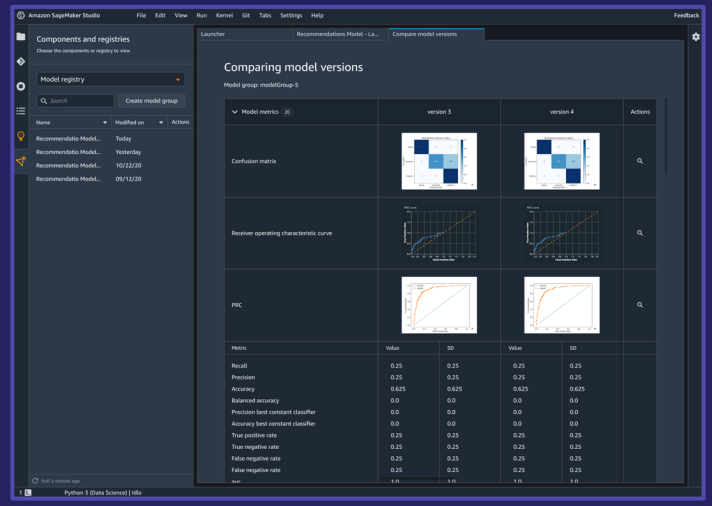
Easily scale compute using SageMaker Training jobs

aws machine learning

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

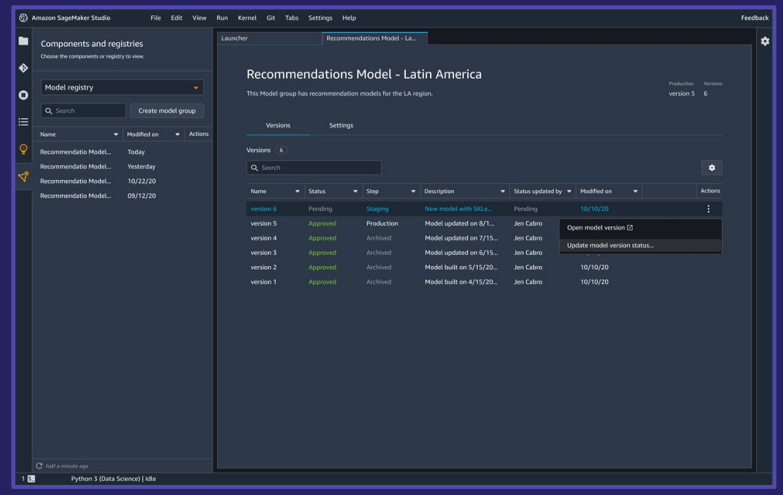


Monitor hosted endpoints directly from Studio UI

Review model quality, data quality and endpoint configuration settings

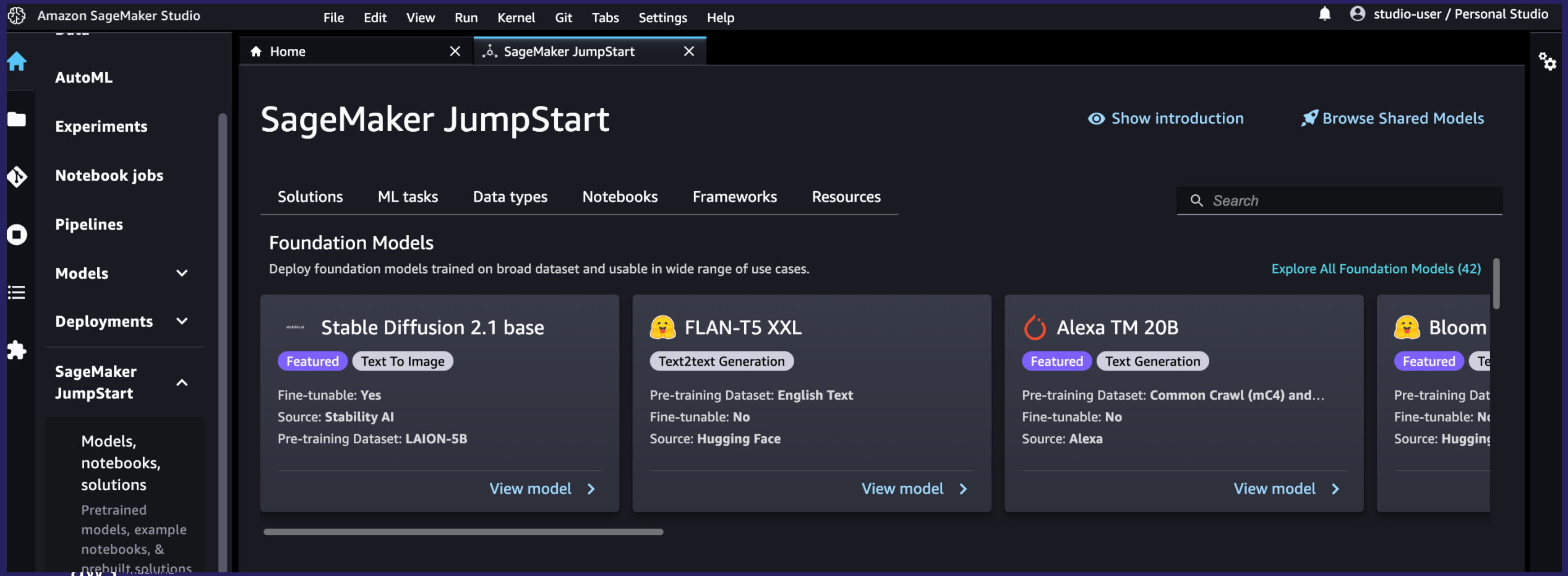
Deploy and test hosted endpoints from SageMaker Studio

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

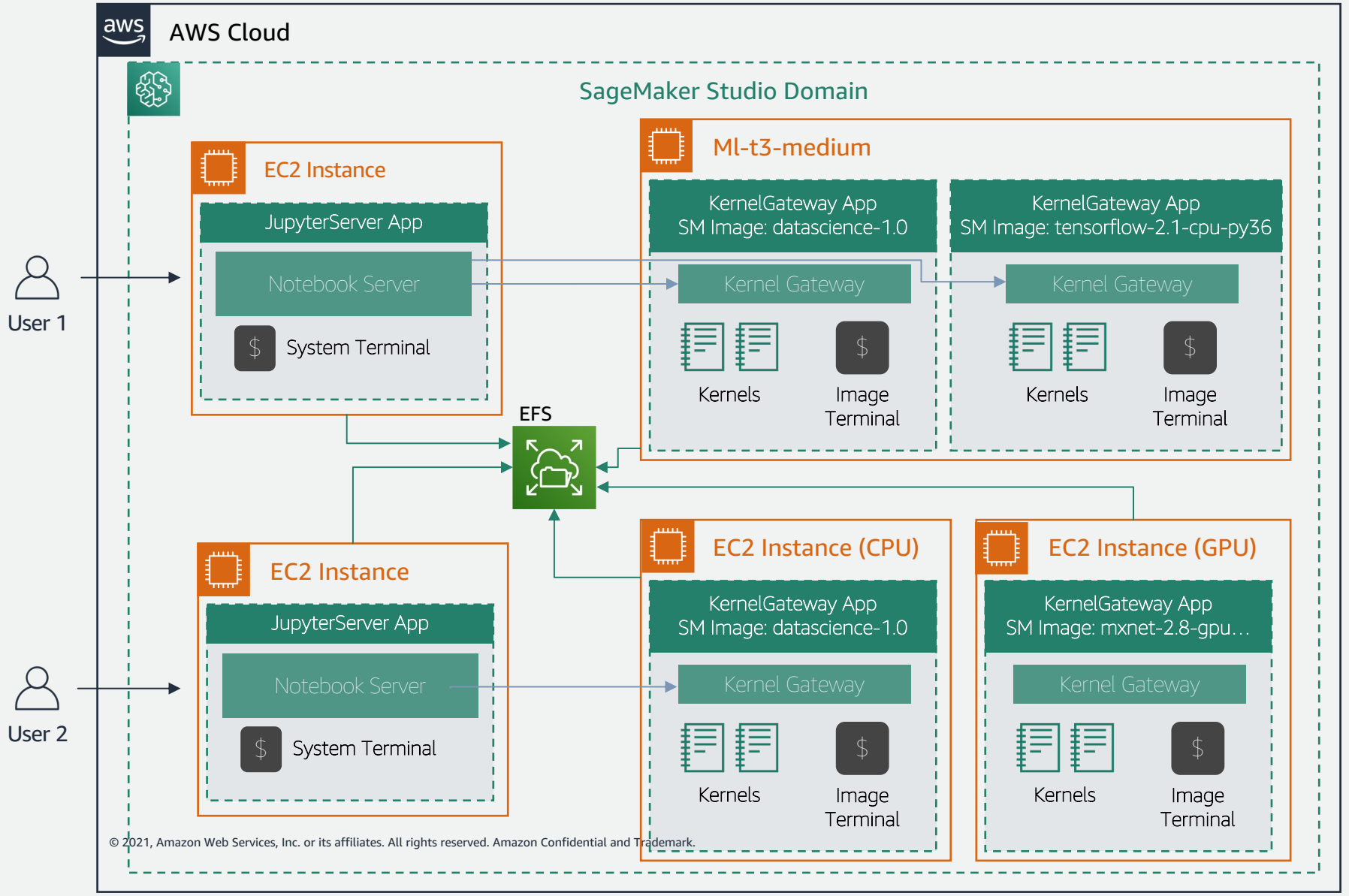


SageMaker Studio Architecture

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).



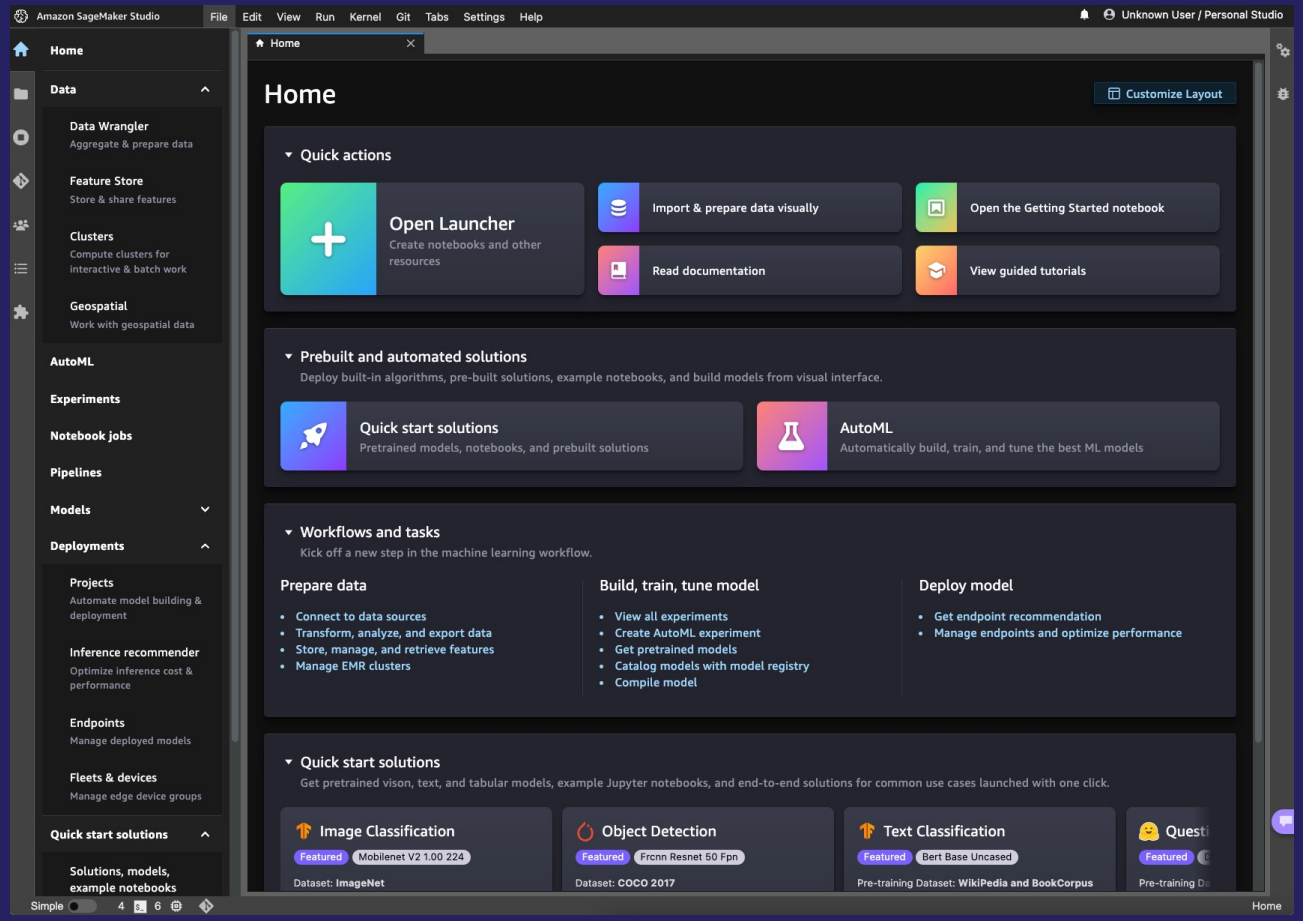
aws machine learning

Updated SageMaker Studio User Interface

Revamped interface enables better feature discoverability

Updated SageMaker Studio UI:

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

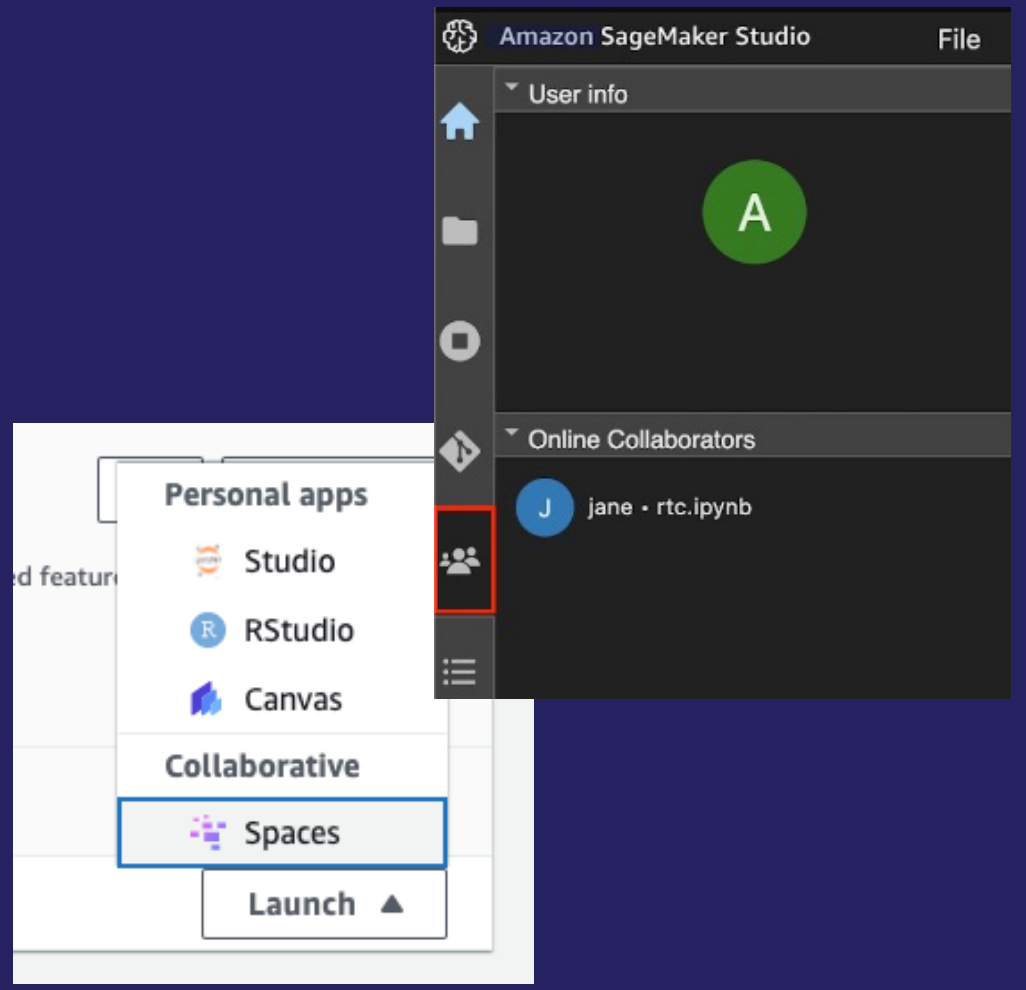


SageMaker Studio now has shared spaces

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.



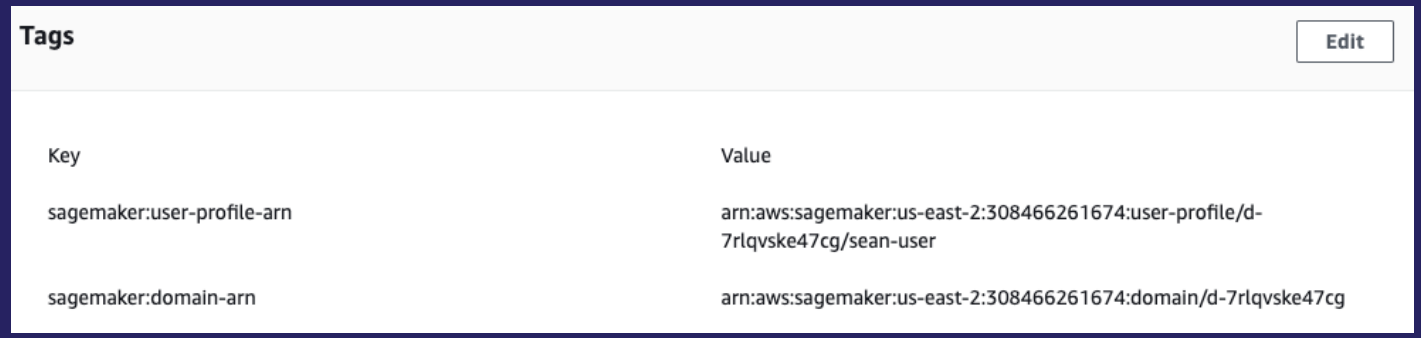
re:Invent launch

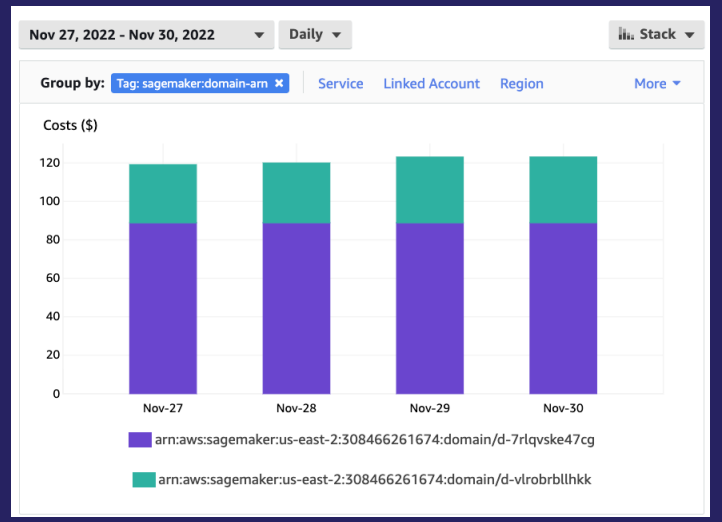
SageMaker Studio now supports automated tagging

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



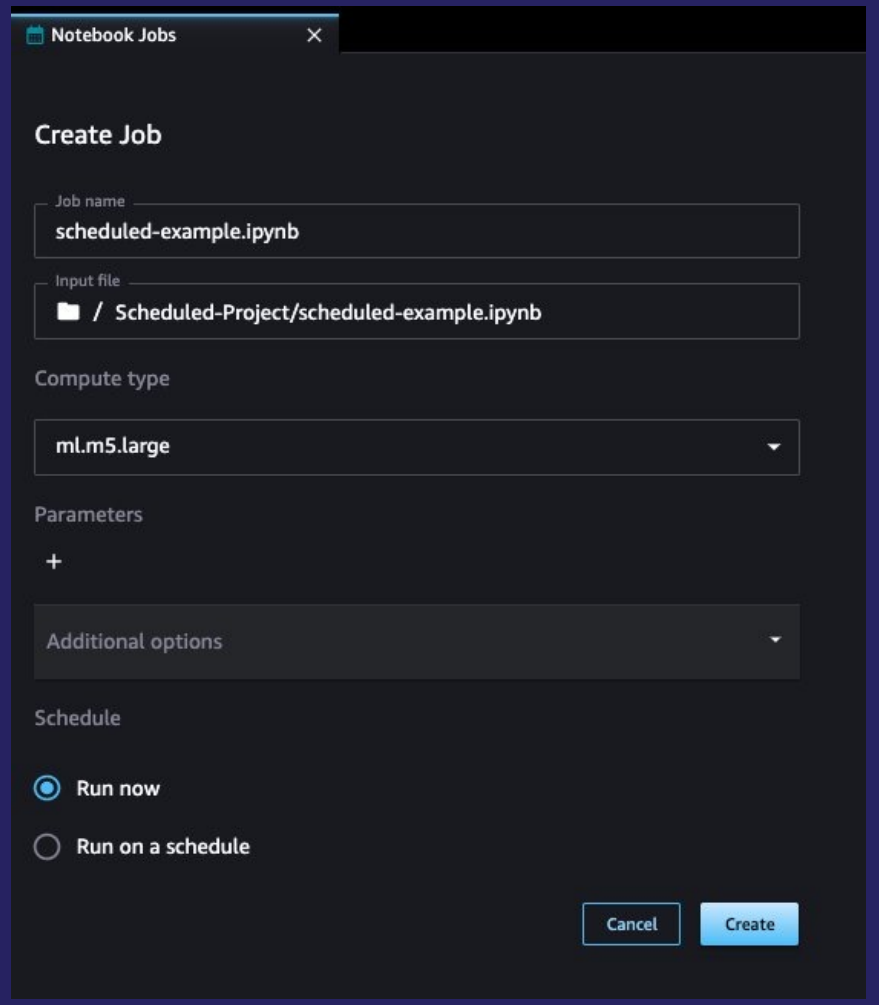


SageMaker Studio now offers Notebook Jobs

Operationalize notebooks in two or three clicks

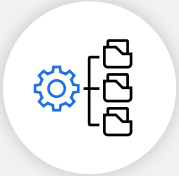
With SageMaker notebook jobs:

- 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

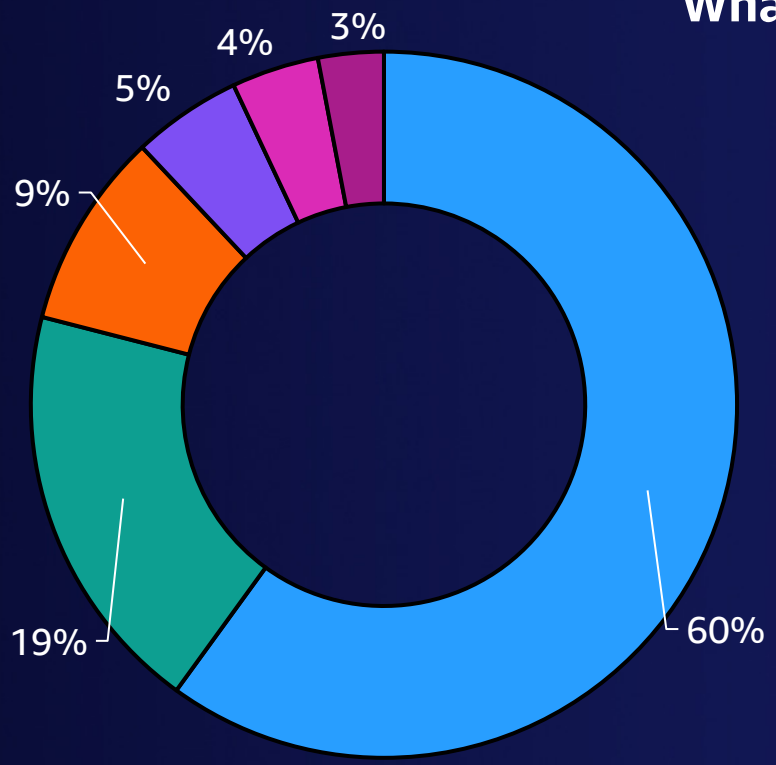
Demo + Q&A

aws machine learning

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



What data scientists spend the most time doing

- Cleaning and organizing data
- Collecting data sets
- Mining data for patterns
- Other
- Refining algorithms
- Building training sets

Source: Forbes survey of 80 data scientists, March 2016

SageMaker Studio Notebooks

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

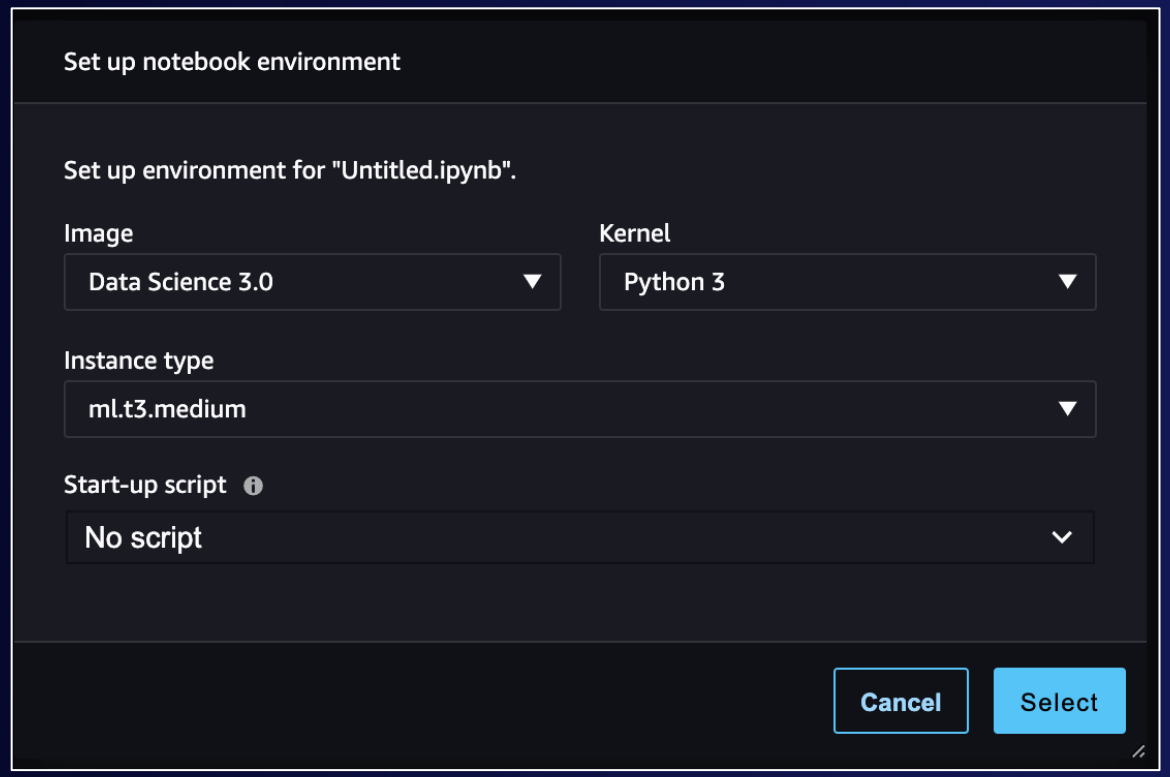
Filter bar for the file browser

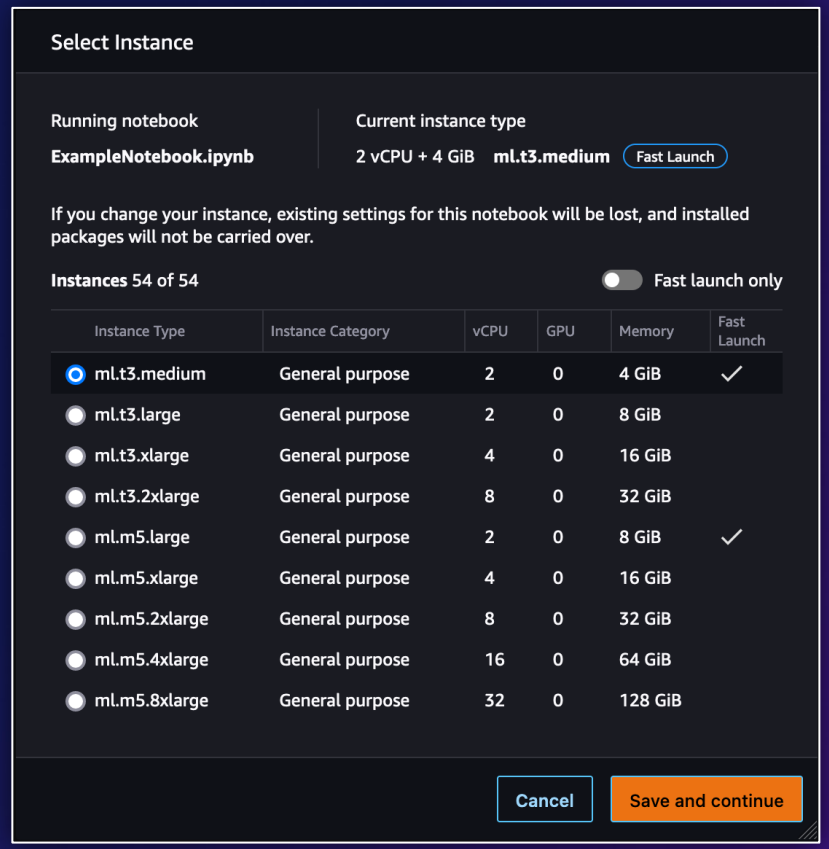
Support for multiple display languages

Install extensions through pip, Conda, and Mamba

SageMaker Studio Notebooks

Managed Environments and Flexible Compute





Amazon SageMaker Processing Job

Managed solution for data processing and model evaluation jobs

Achieve distributed processing for clusters

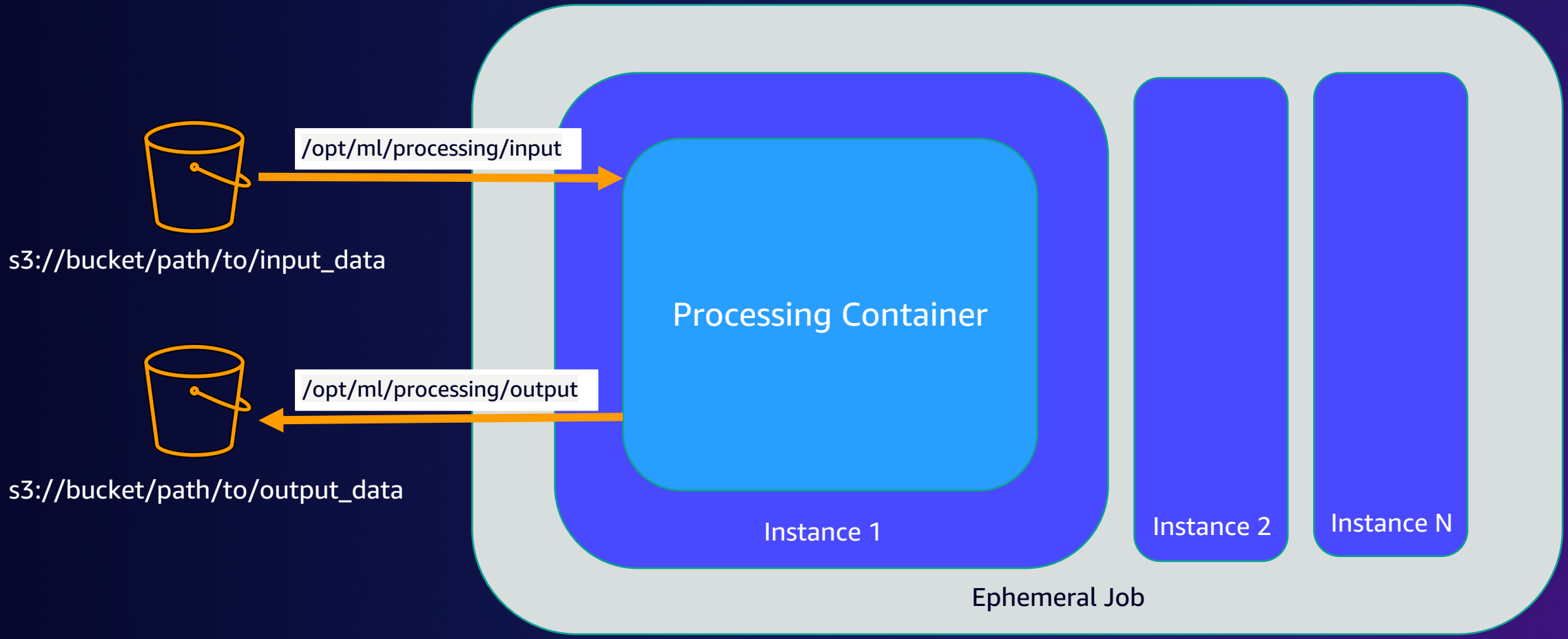
Bring your own script for feature engineering

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.processing 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

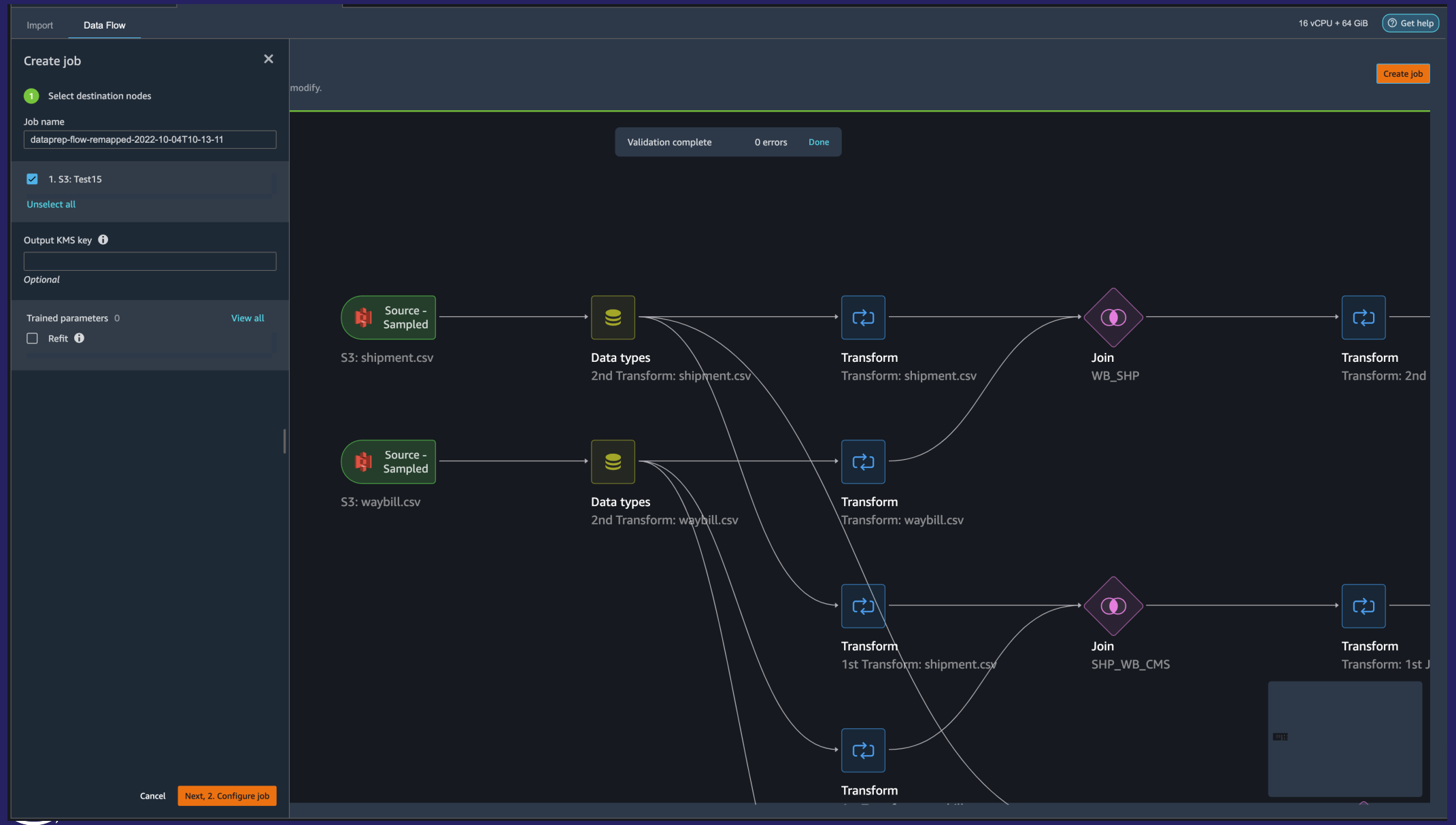
No-code data preparation

Single visual interface for common data prep techniques

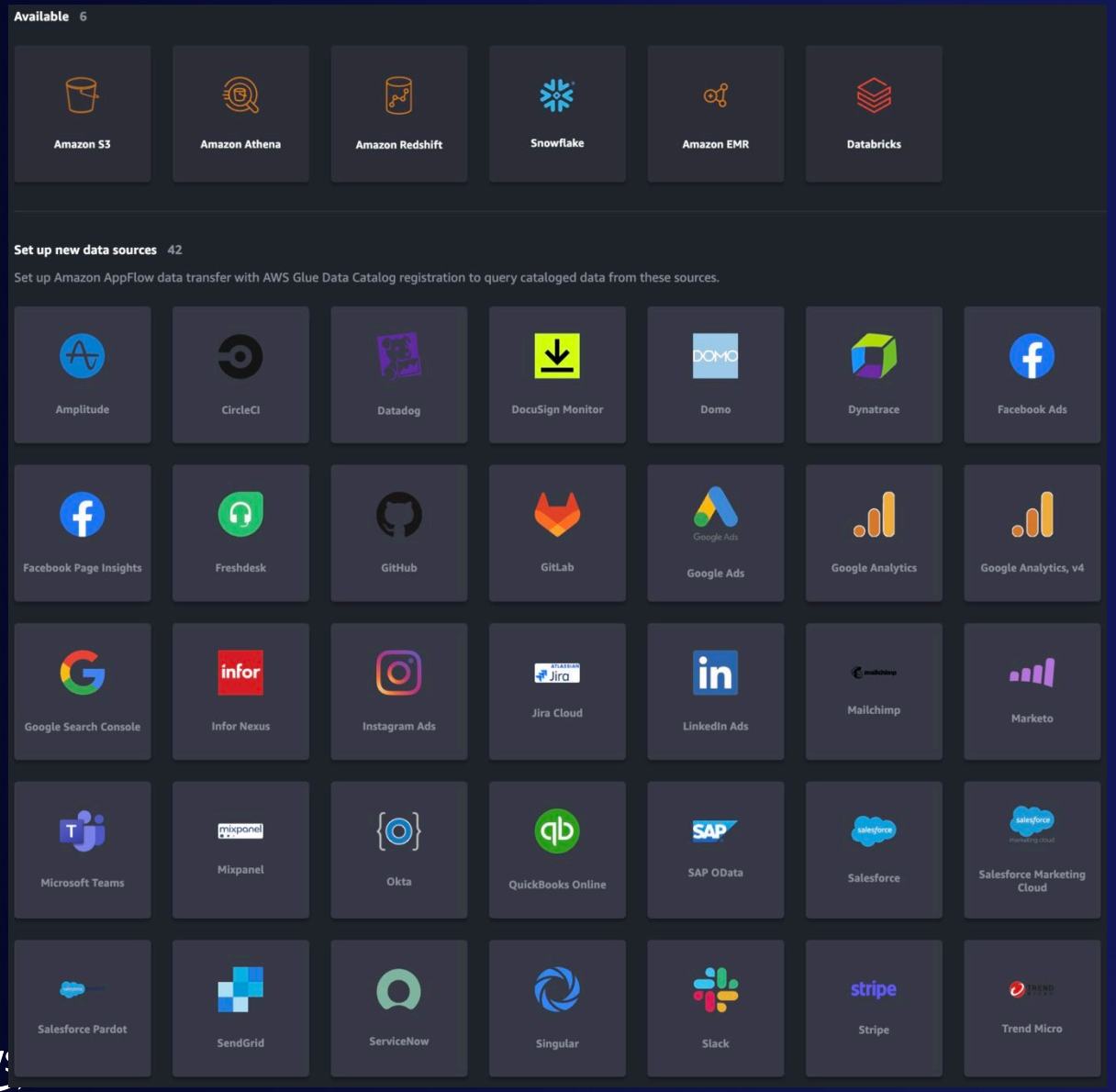
Select data from multiple sources

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

Visual Workflow Interface



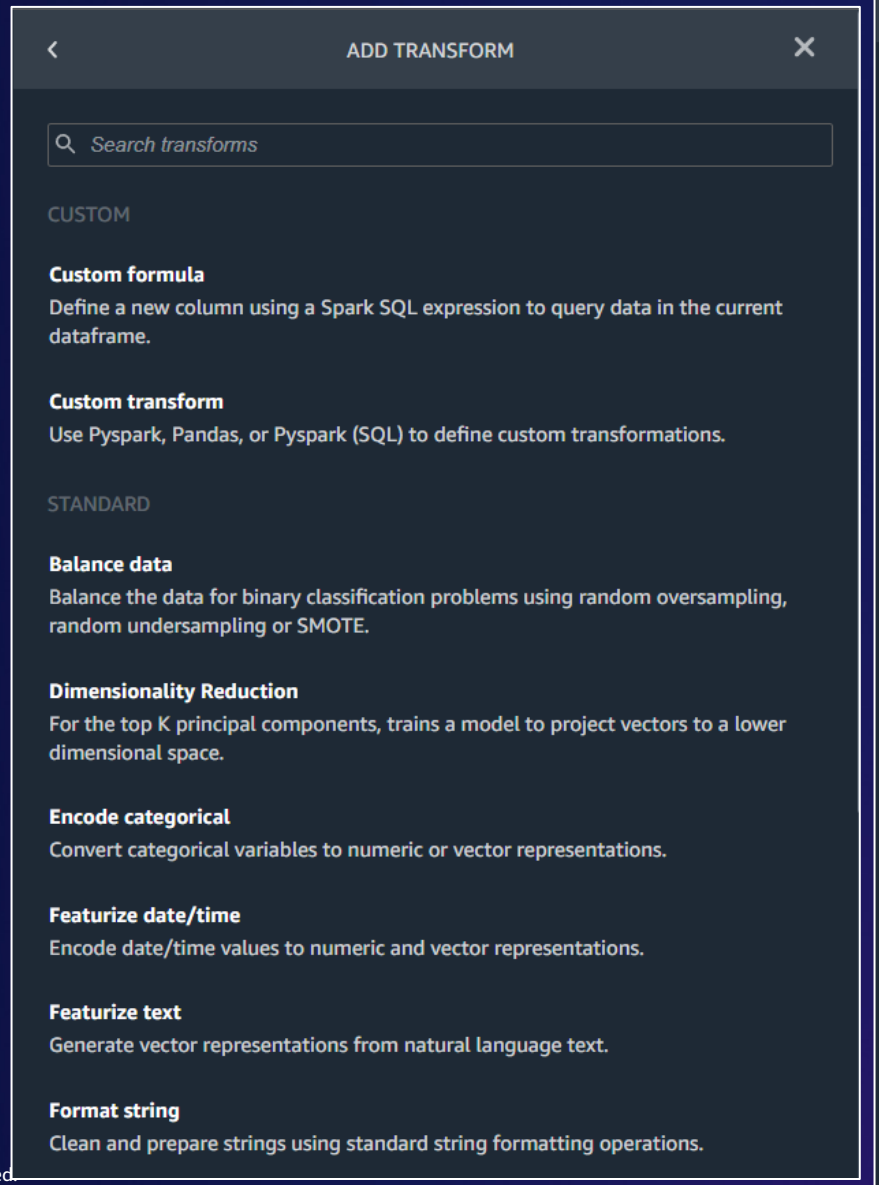
Quickly connect and import data

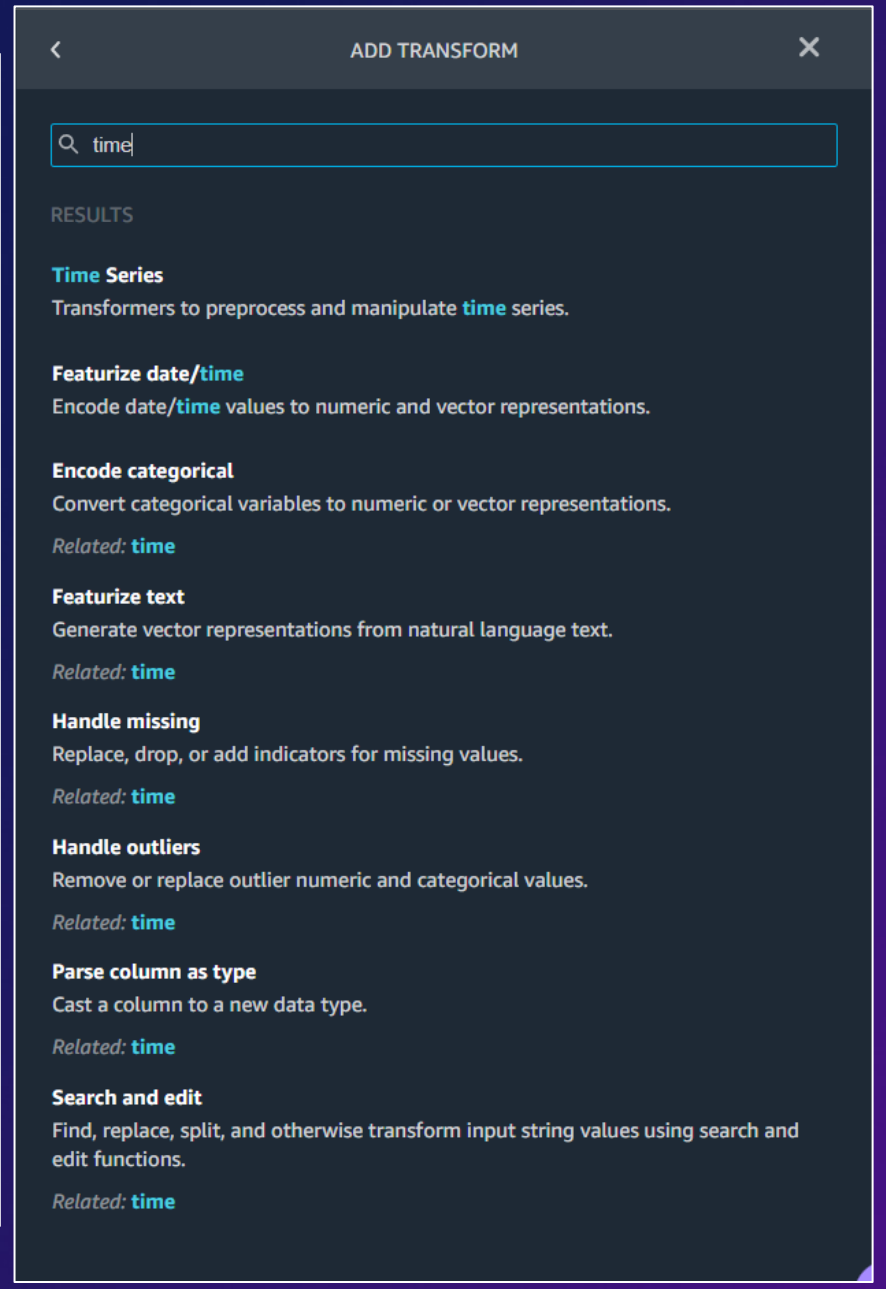


- **Connect to multiple data sources**
  - Amazon S3, Amazon Athena, Amazon Redshift, Snowflake, Databricks
  - Salesforce Customer Data Platform (Preview)
  - Amazon EMR
  - Support 40+ SaaS data sources
- **Support for common file formats:**
  - CSV, Parquet, JSON and JSONL, ORC, Database tables
- **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

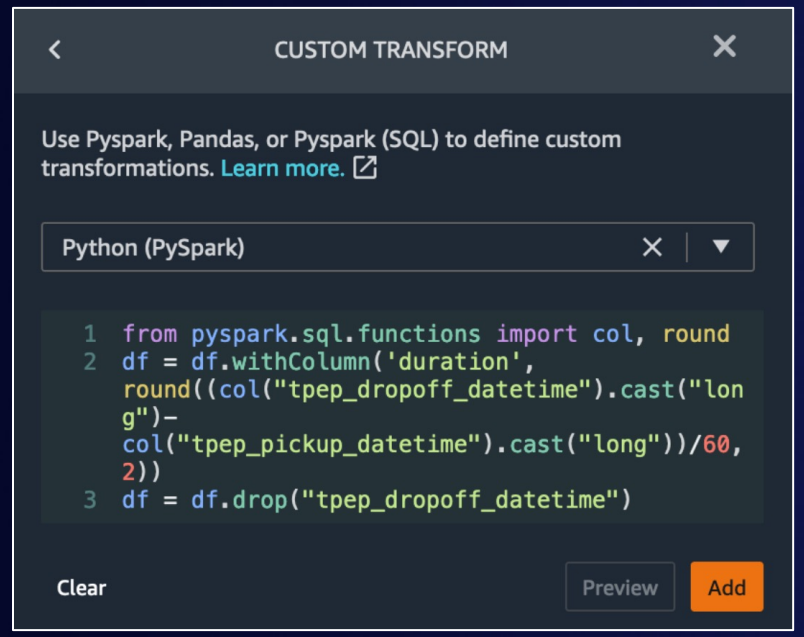
- Common and ML transforms
- Time Series, Train/Test split, Dimensionality Reduction, Multi-collinearity, PCA…



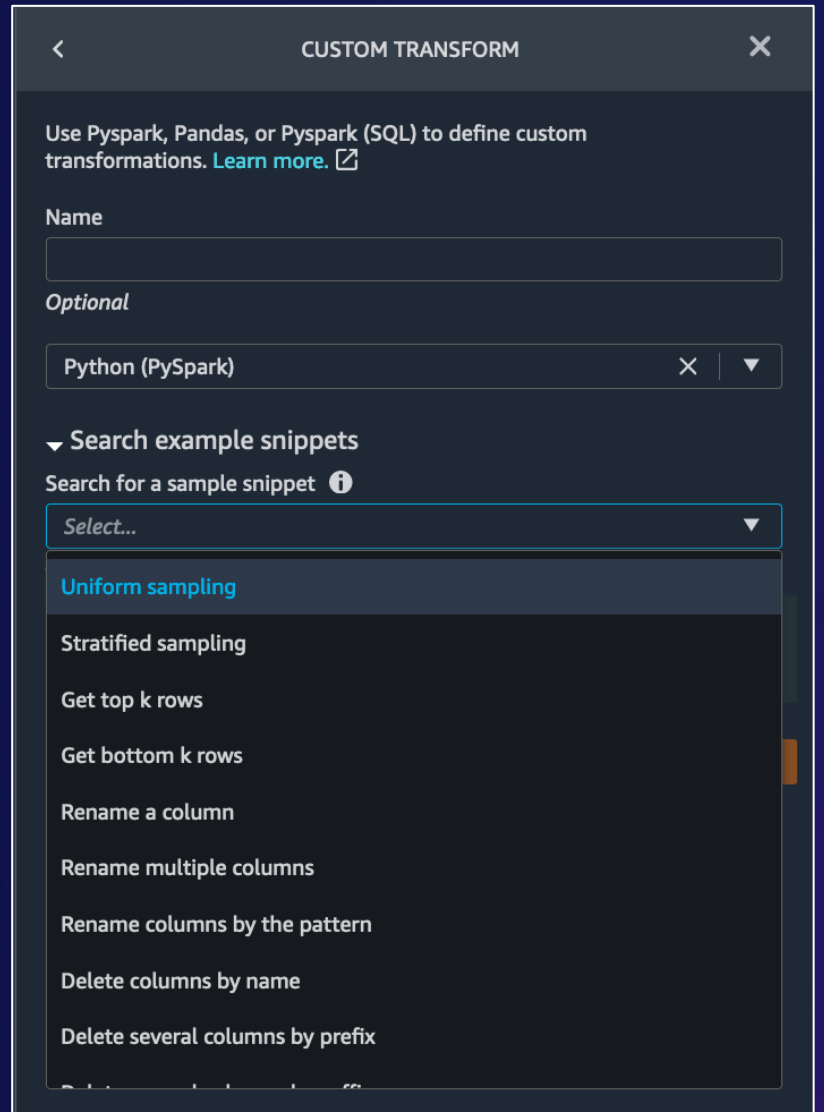


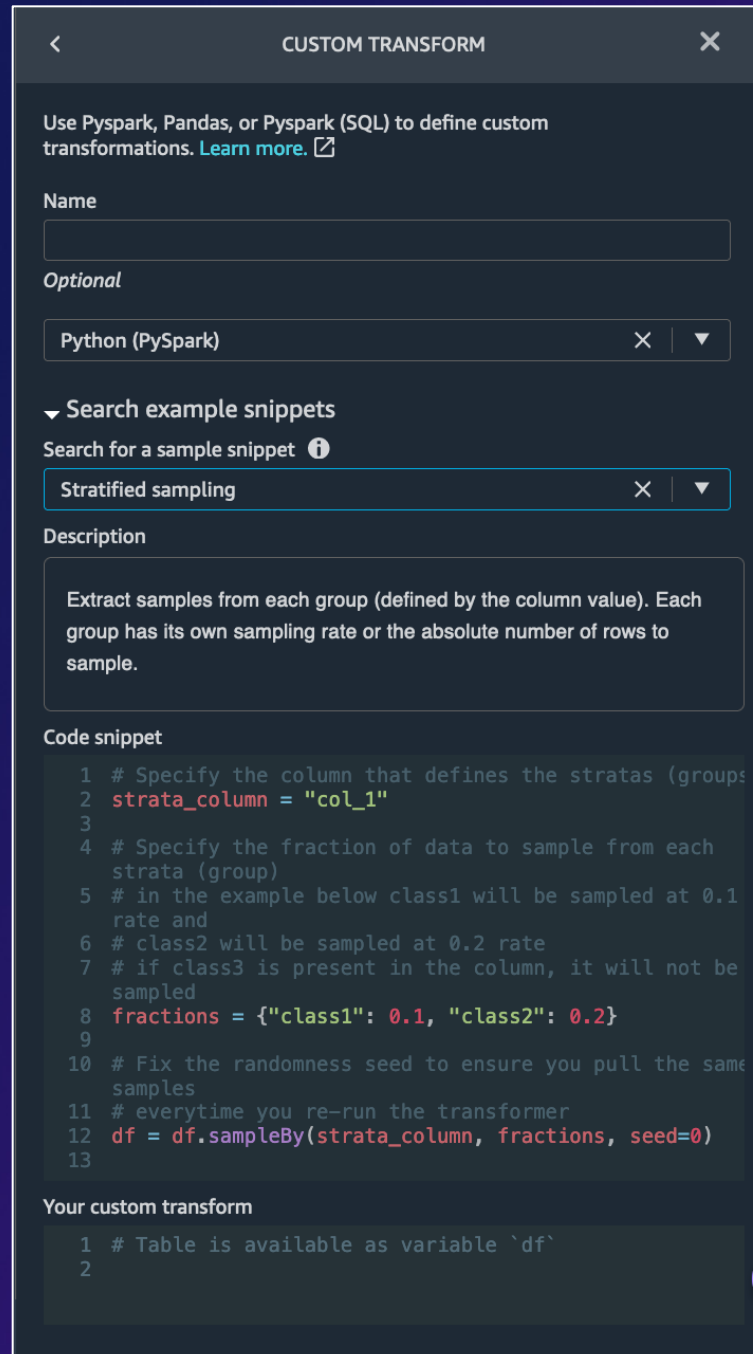
Custom transforms & code snippets

Panda, Spark SQL, PySpark



Common code snippet library

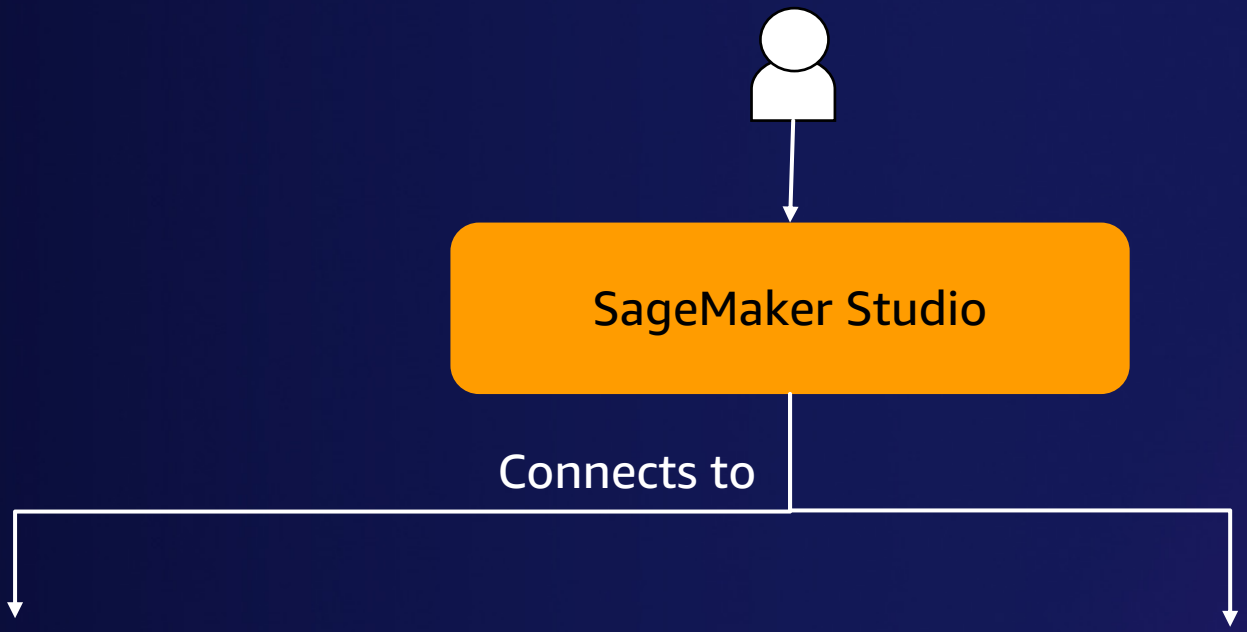




Understand data visually: identify quality issues

- 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



Amazon EMR

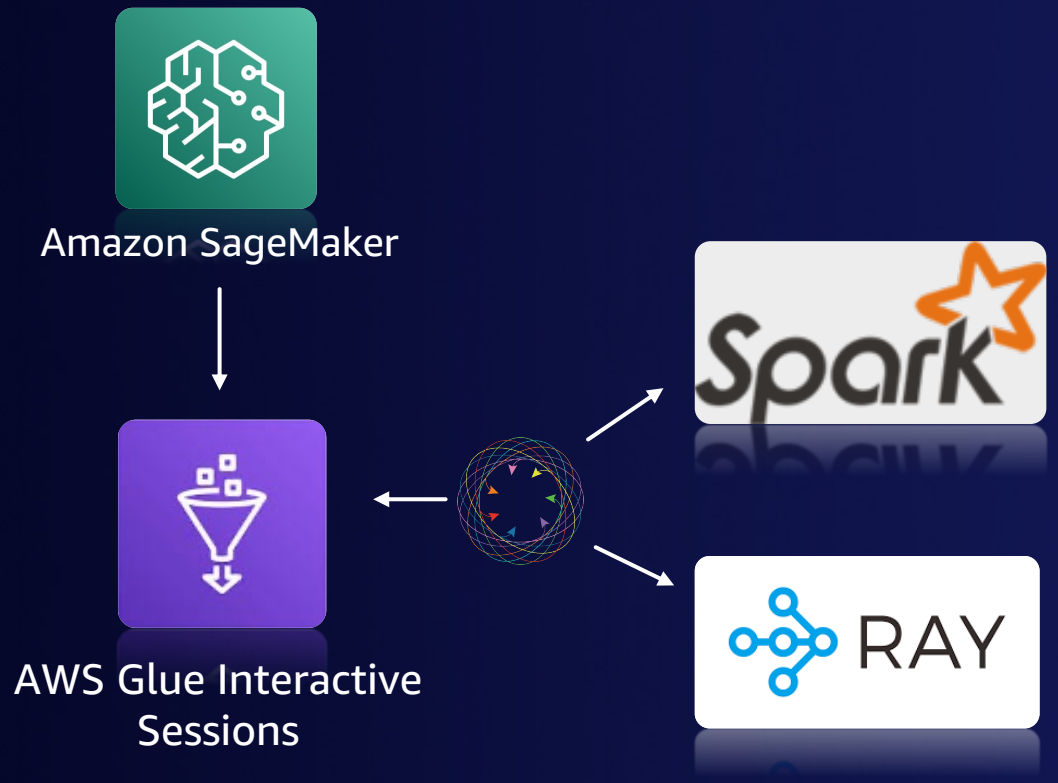
Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data

AWS Glue

AWS Glue is a serverless data integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources.

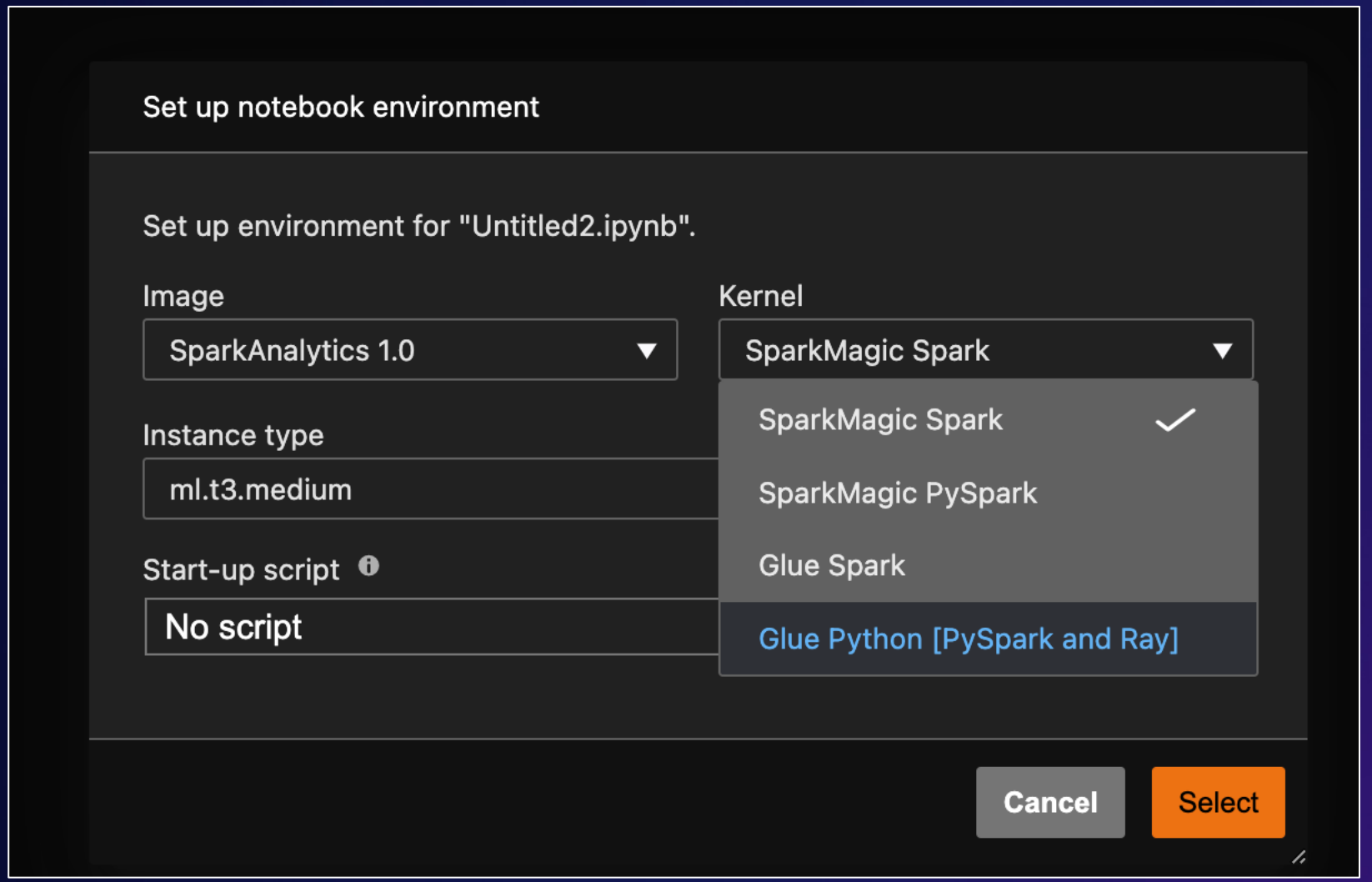
SageMaker Studio + Glue Interactive Sessions

Scaling your data integration workloads using interactive Apache Spark and Ray

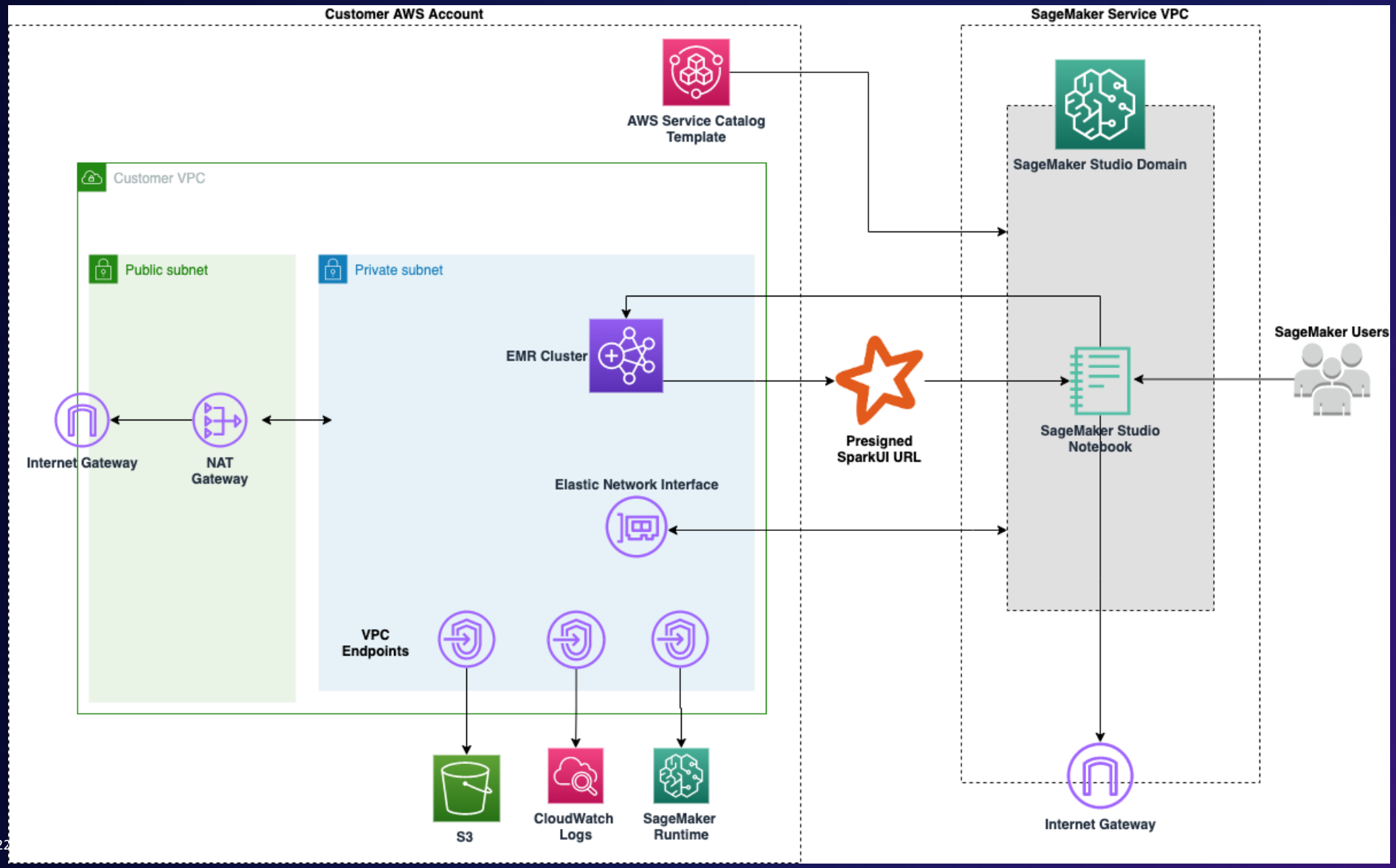


- From SageMaker Studio, access AWS Glue Interactive Sessions to access large, serverless, distributed clusters for Apache Spark and Ray
- Data engineers and ML practitioners can use Apache Spark or Ray to process large datasets with Python and popular Python libraries.
- Train distributed ML models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

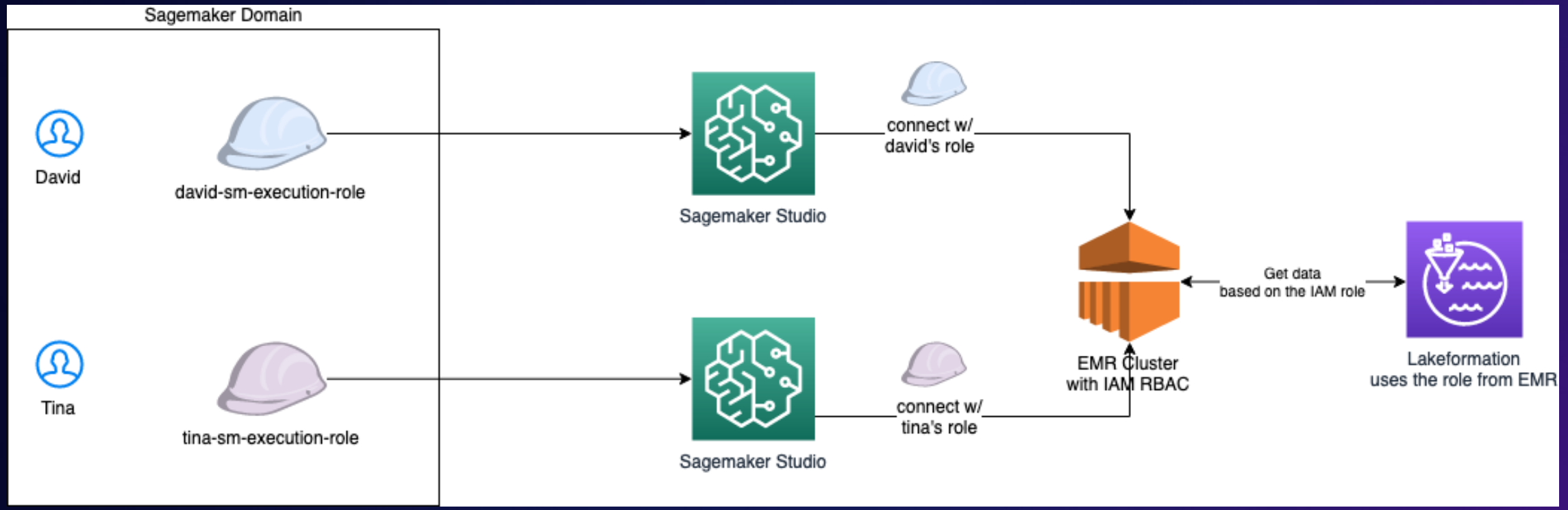
Glue Kernels Built into SageMaker Studio



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

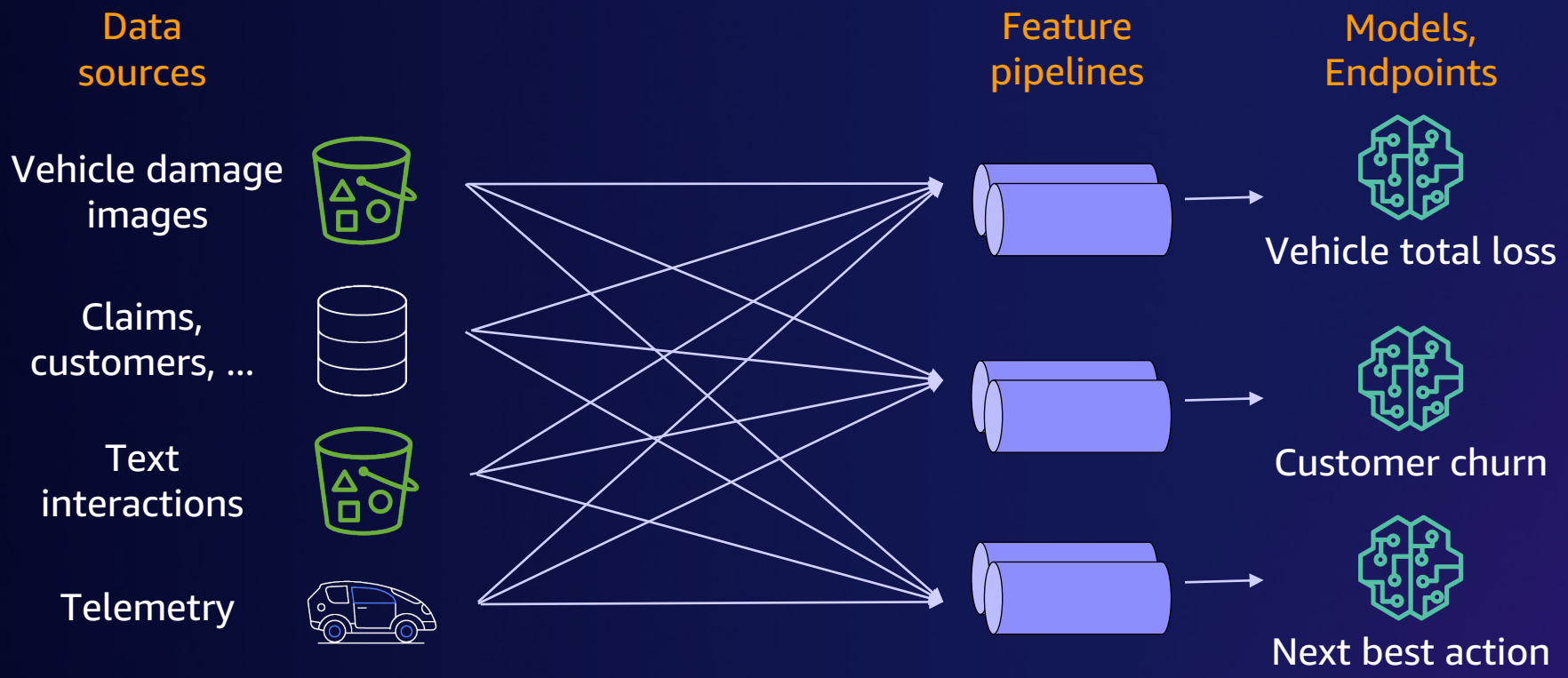
Fully managed

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

aws

Teams too often start from scratch

Standalone feature engineering for each new model

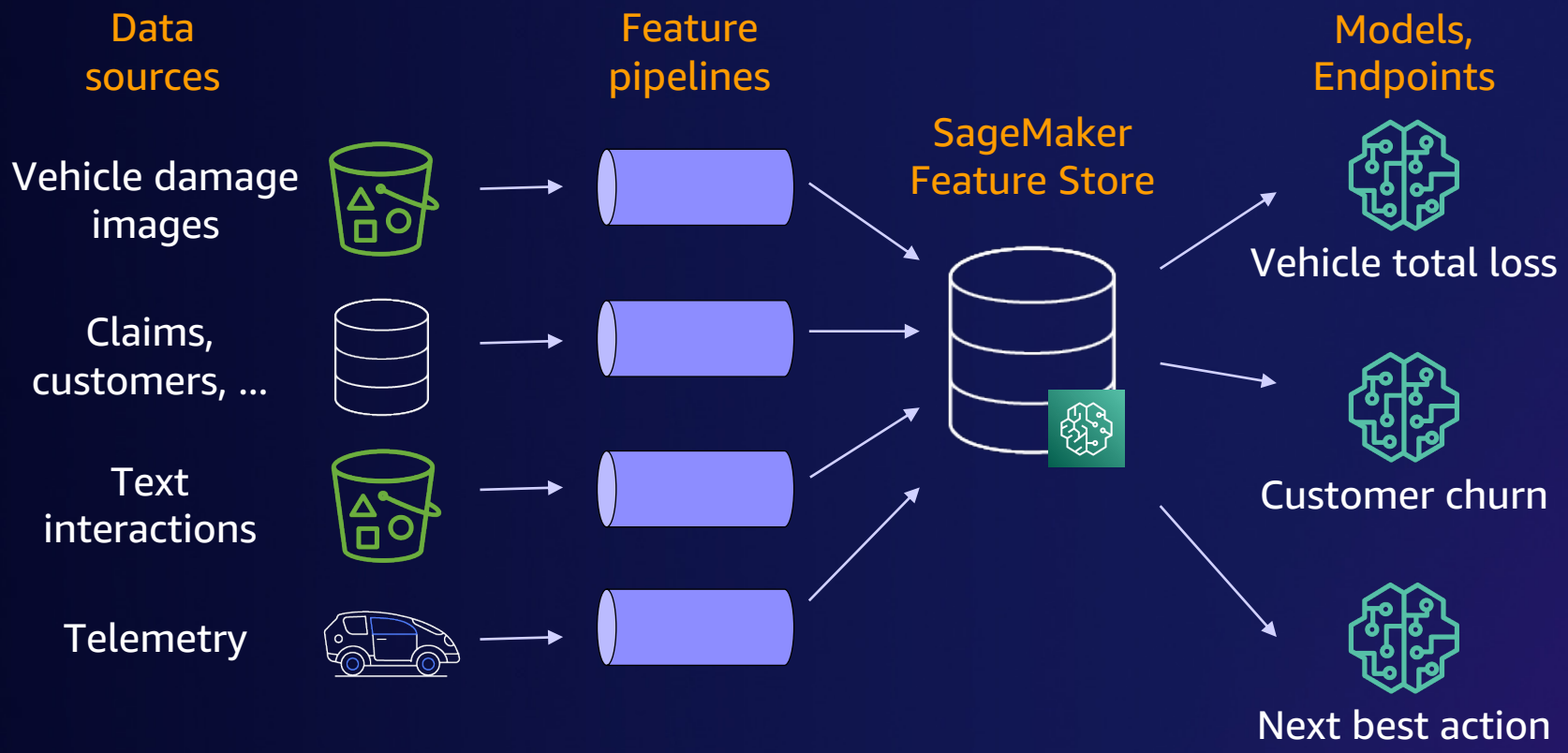


Challenges

- Feature duplication
- Slow time to market
- Inaccurate predictions

With SageMaker Feature Store...

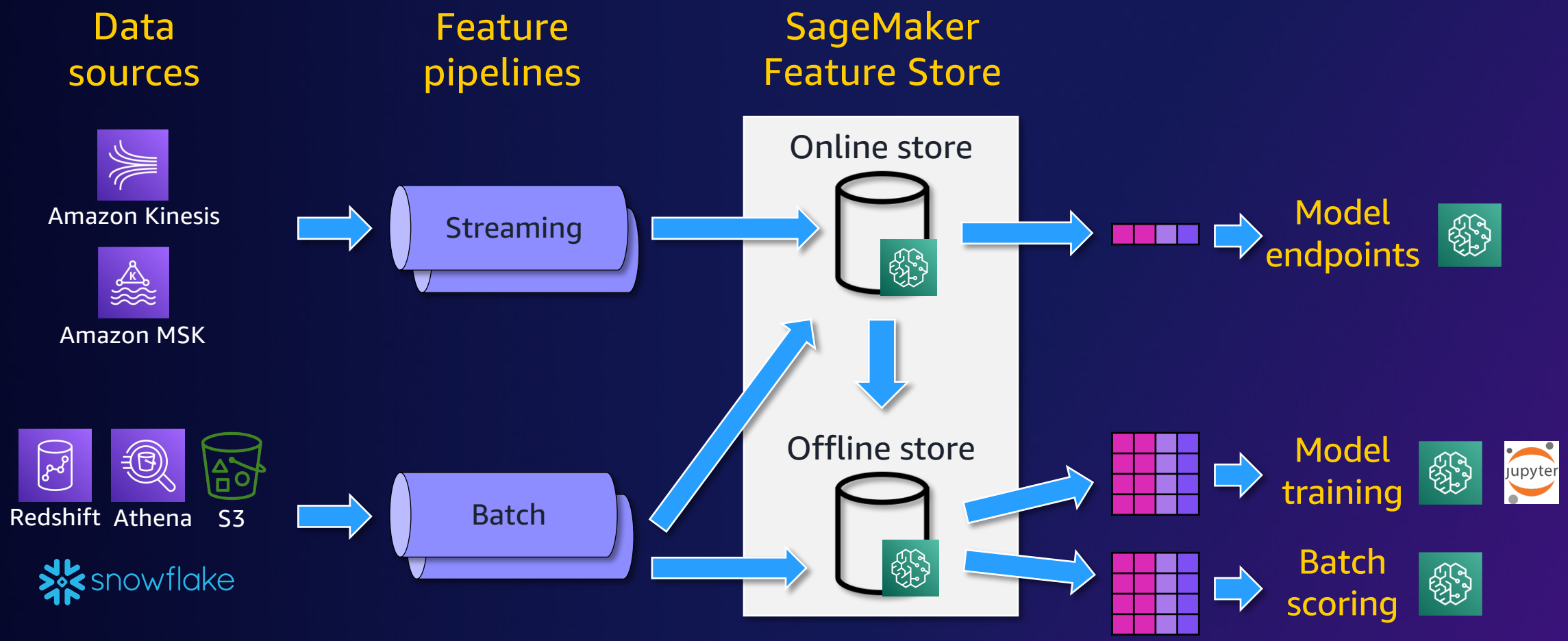
Build features once, reuse them across teams and models



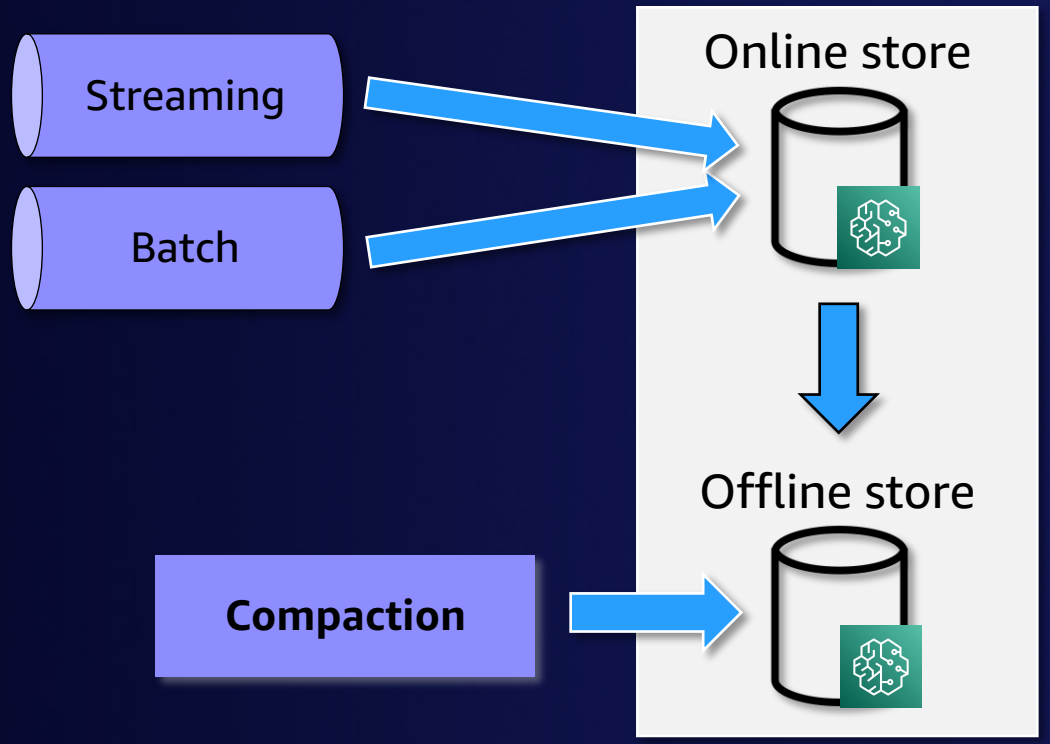
Benefits

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

Feature store in context



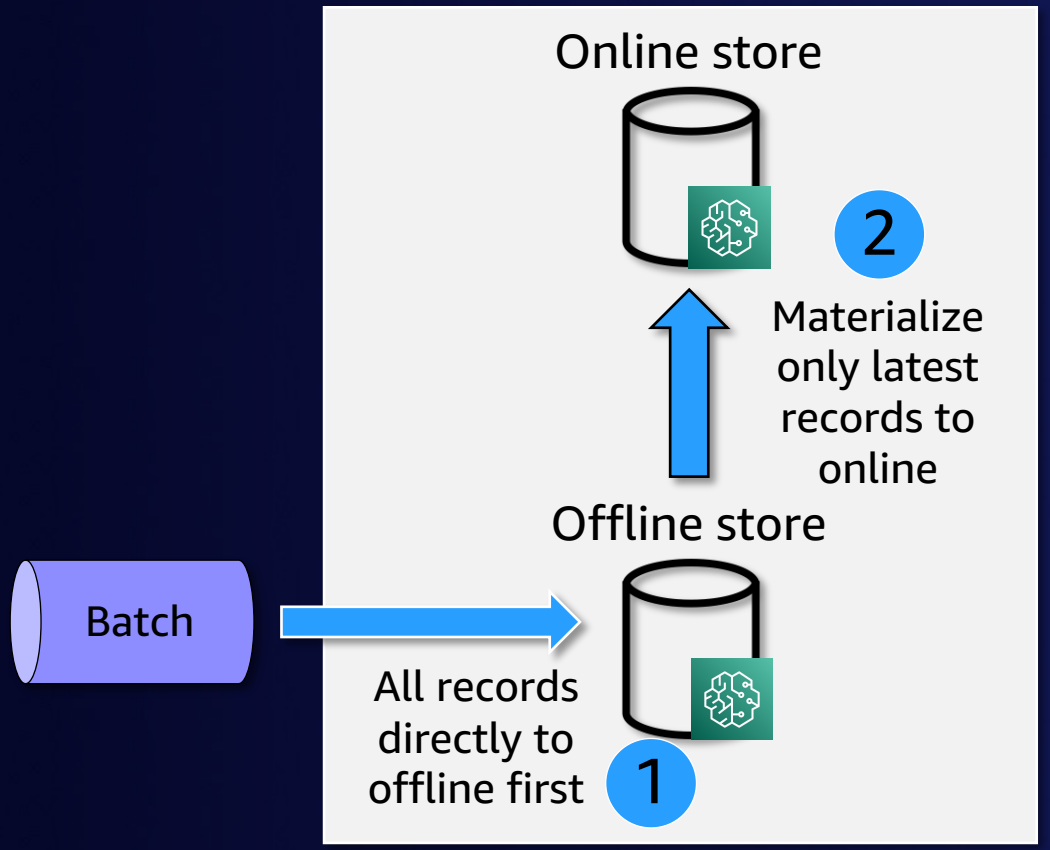
Ingestion data flow pattern 1: Online to offline replication



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

aws machine learning

Lab 1 - Glue Interactive sessions with Spark and Ray

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

Access Code: 3e17-0d3b81-dc

aws machine learning

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

aws machine learning

Amazon SageMaker has built-in algorithms or bring your own

Computer vision
Image classification | Object detection | Semantic segmentation

Topic modeling
LDA | NTM

Classification
Linear Learner | XGBoost | KNN

Recommendation
Factorization machines

Forecasting
DeepAR

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%

aws machine learning

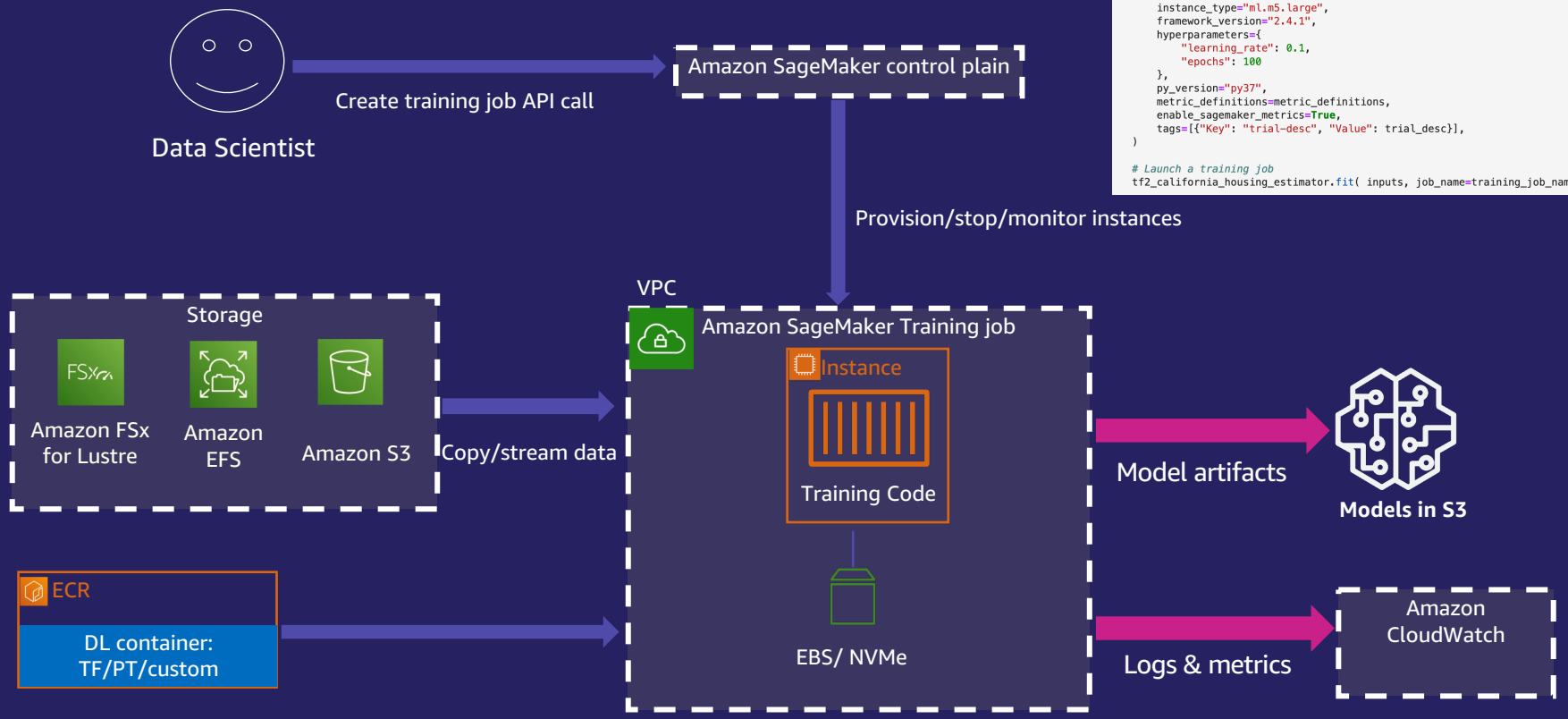
Training on Amazon SageMaker

```
# Input data from s3
inputs = {"train": s3_inputs_train, "test": s3_inputs_test}

metric_definitions = [
    {"Name": "loss", "Regex": "loss: ([0-9\\.]+)"},
    {"Name": "accuracy", "Regex": "accuracy: ([0-9\\.]+)"},
    {"Name": "val_loss", "Regex": "val_loss: ([0-9\\.]+)"},
    {"Name": "val_accuracy", "Regex": "val_accuracy: ([0-9\\.]+)"},
]

# Create a TensorFlow Estimator
tf2_california_housing_estimator = TensorFlow(
    entry_point="california_housing_tf2.py",
    source_dir="code",
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type="ml.m5.large",
    framework_version="2.4.1",
    hyperparameters={
        "learning_rate": 0.1,
        "epochs": 100
    },
    py_version="py37",
    metric_definitions=metric_definitions,
    enable_sagemaker_metrics=True,
    tags=[{"Key": "trial-desc", "Value": trial_desc}],
)

# Launch a training job
tf2_california_housing_estimator.fit( inputs, job_name=training_job_name)
```



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

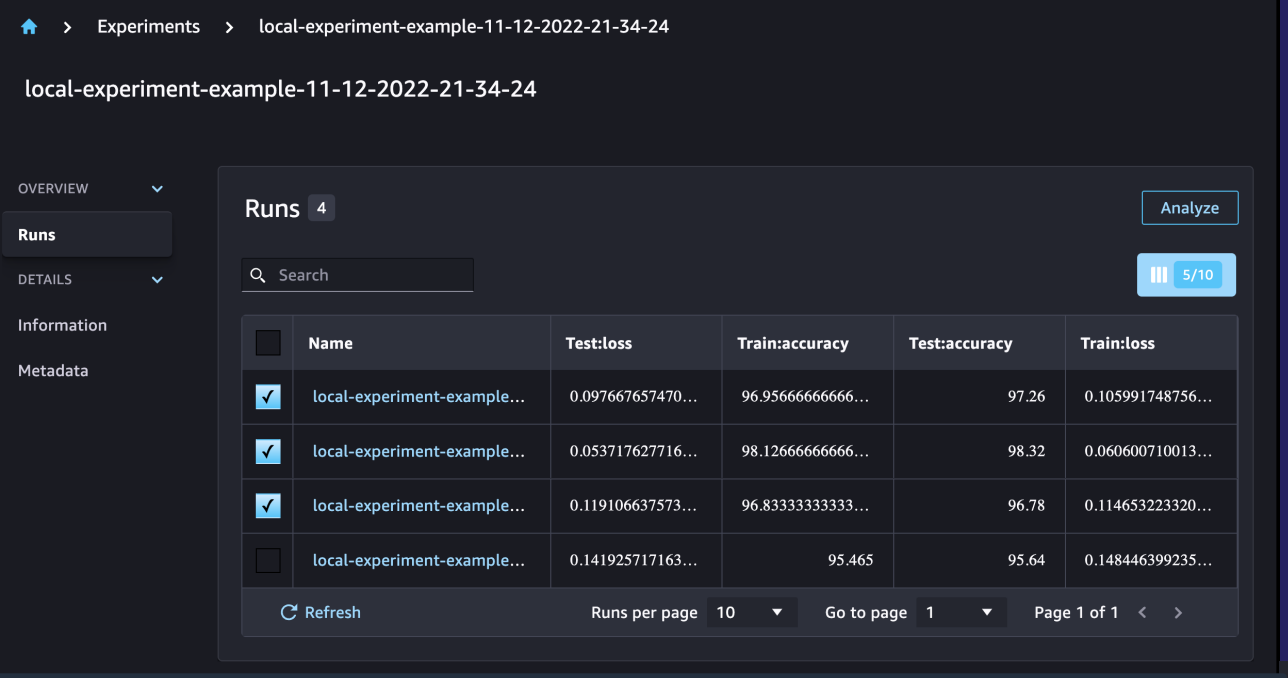
Visually identify the best performing models

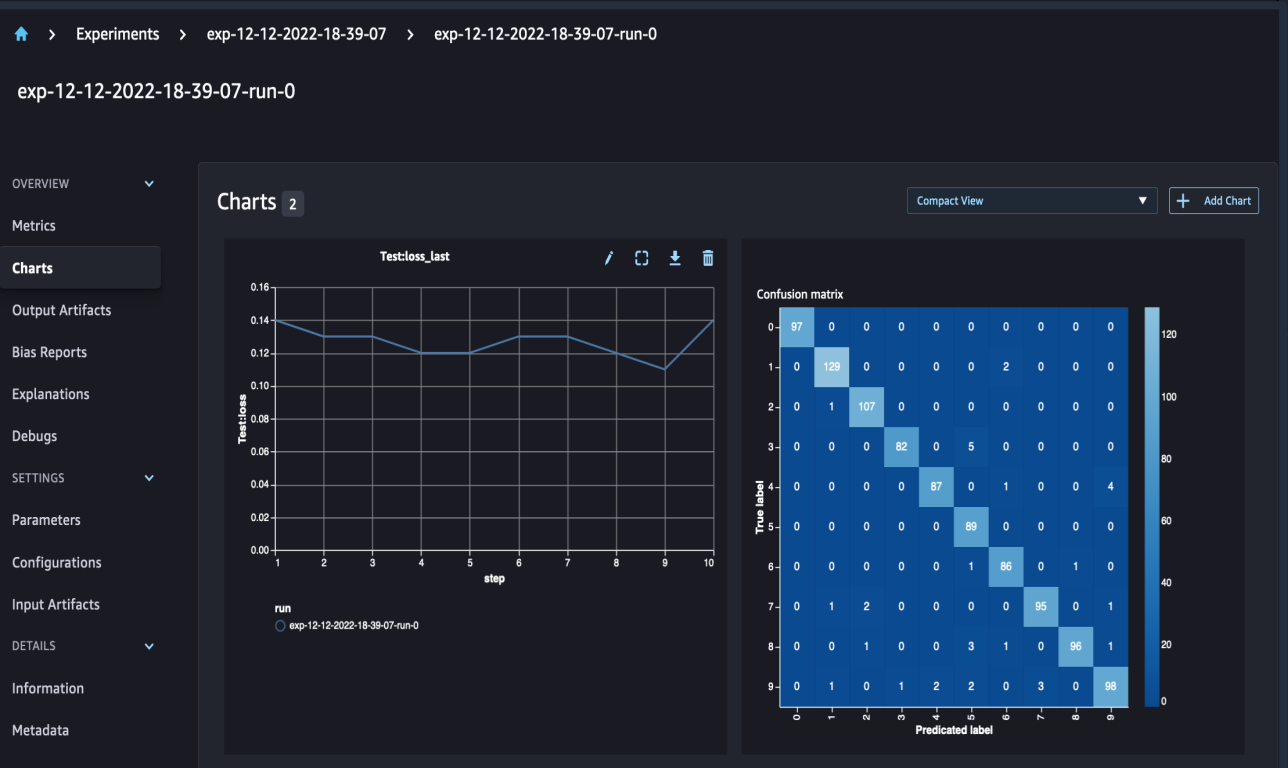
Ensure models are reliable and reproducible

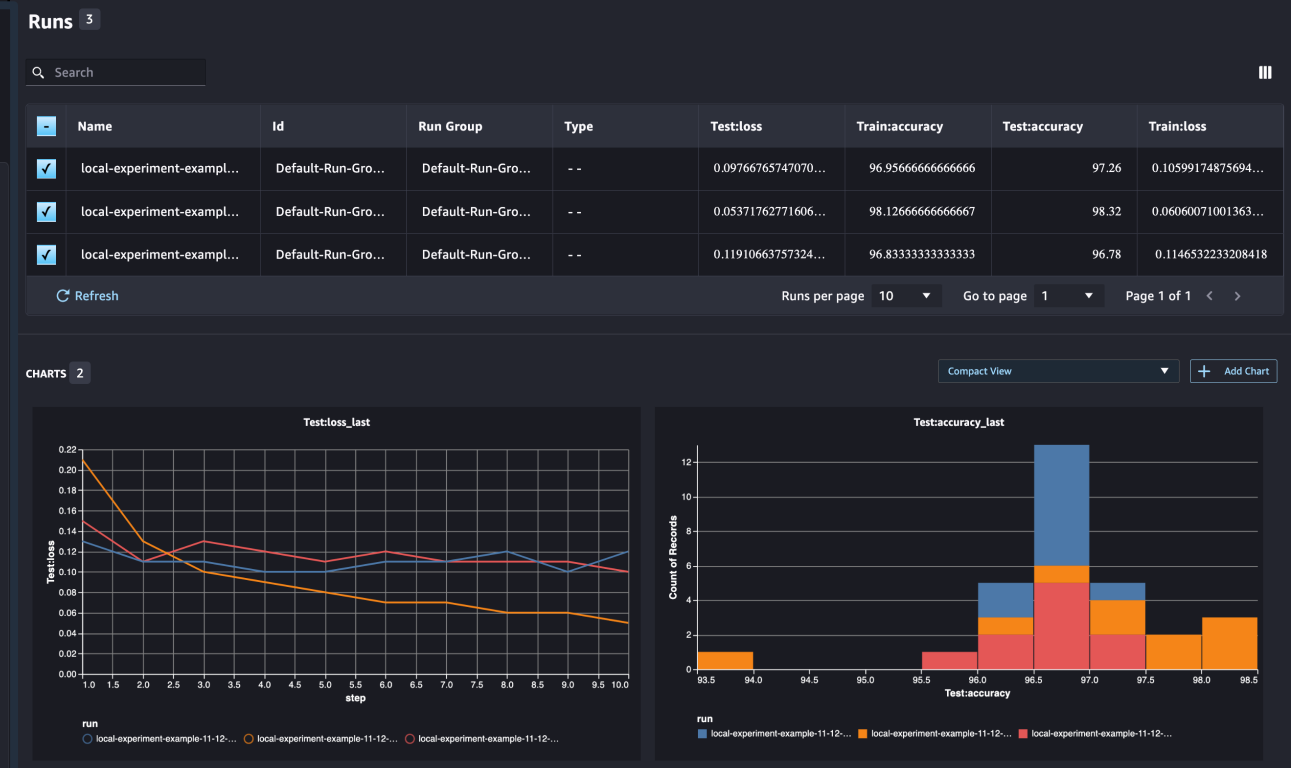
SageMaker Experiments Concepts

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().

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

EXAMPLES

Decision trees

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

Neural networks

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

aws machine learning

Amazon SageMaker Automatic Model Tuning

Hyperparameter Tuning

aws machine learning

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

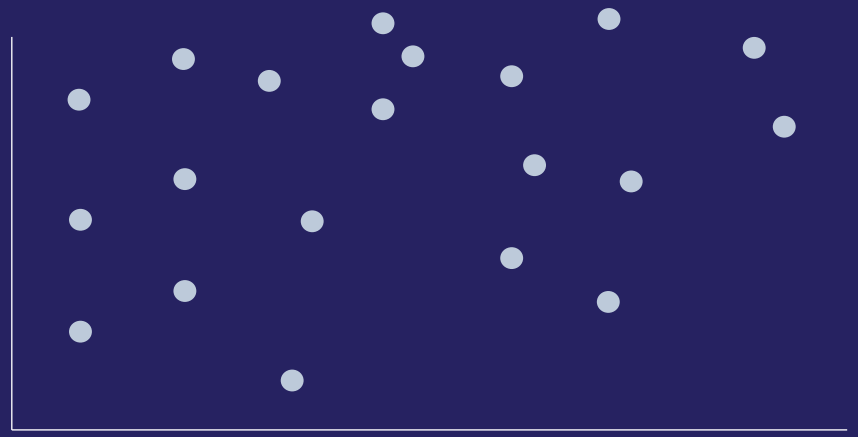
```
tuner = HyperparameterTuner(xgb,
                            objective_metric_name,
                            hyperparameter_ranges,
                            max_jobs=20,
                            max_parallel_jobs=3)
```

aws machine learning

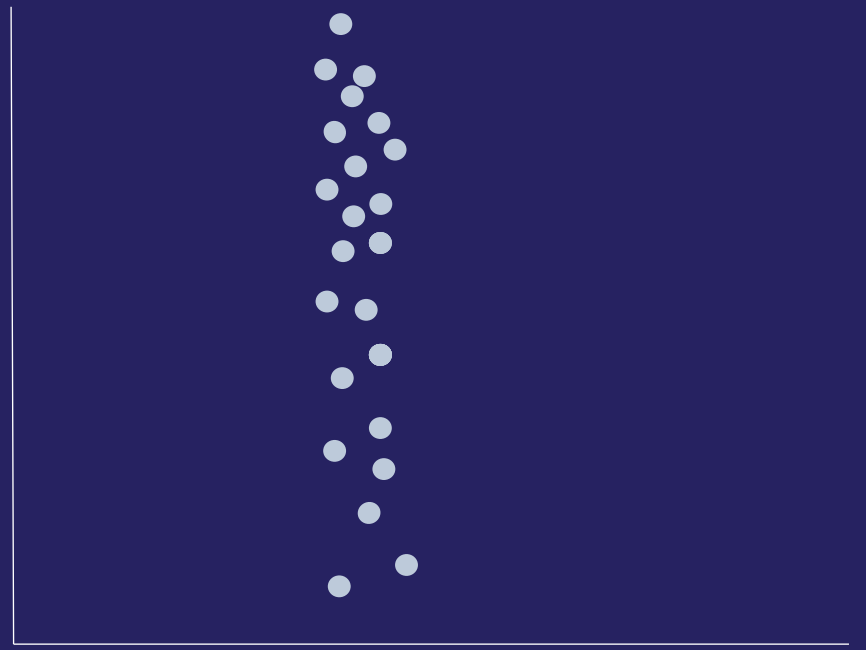
Amazon SageMaker Automatic Model Tuning

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

Bayesian Search



Random Search



aws machine learning

Amazon SageMaker Automatic Model Tuning

Can I use hyperparameter tuning with my own model ?

Built-in Algorithms

Docker

Script Mode

Fully Customizable

aws machine learning

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 

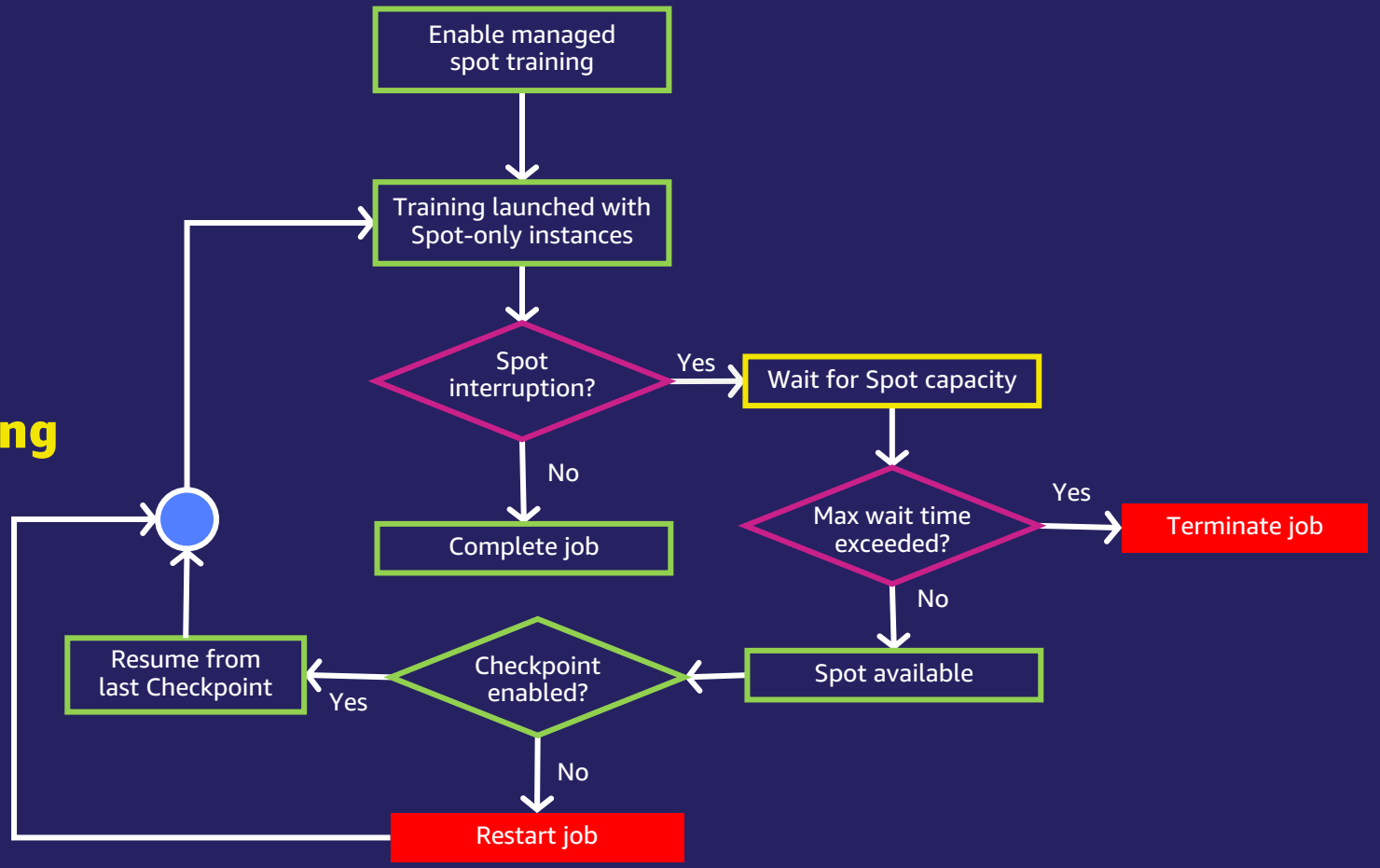
Script Mode 

aws machine learning

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

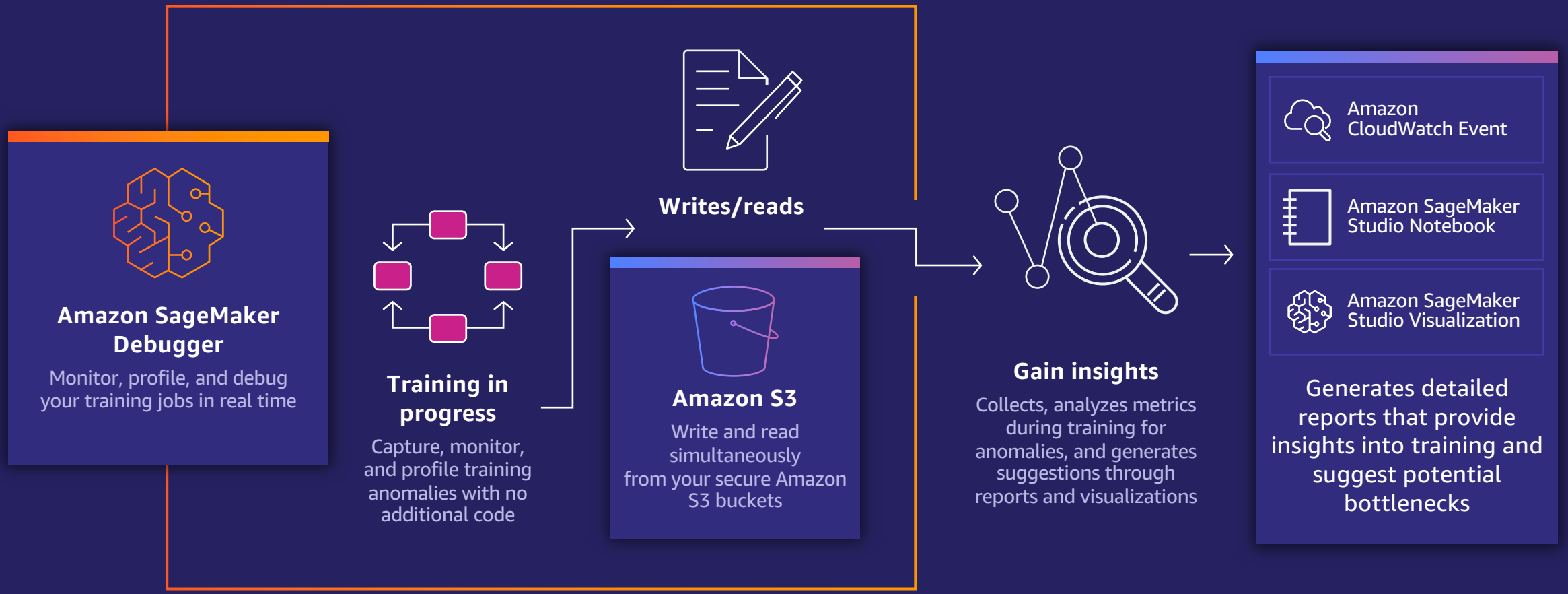
Optimize resources with no additional code

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



aws machine learning

Distributed training

The fastest and easiest way to train large deep learning models

Reduced training time

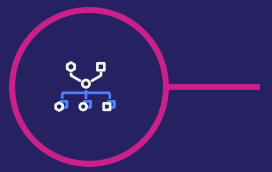
Reduce training time by 25% with synchronization across GPUs

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

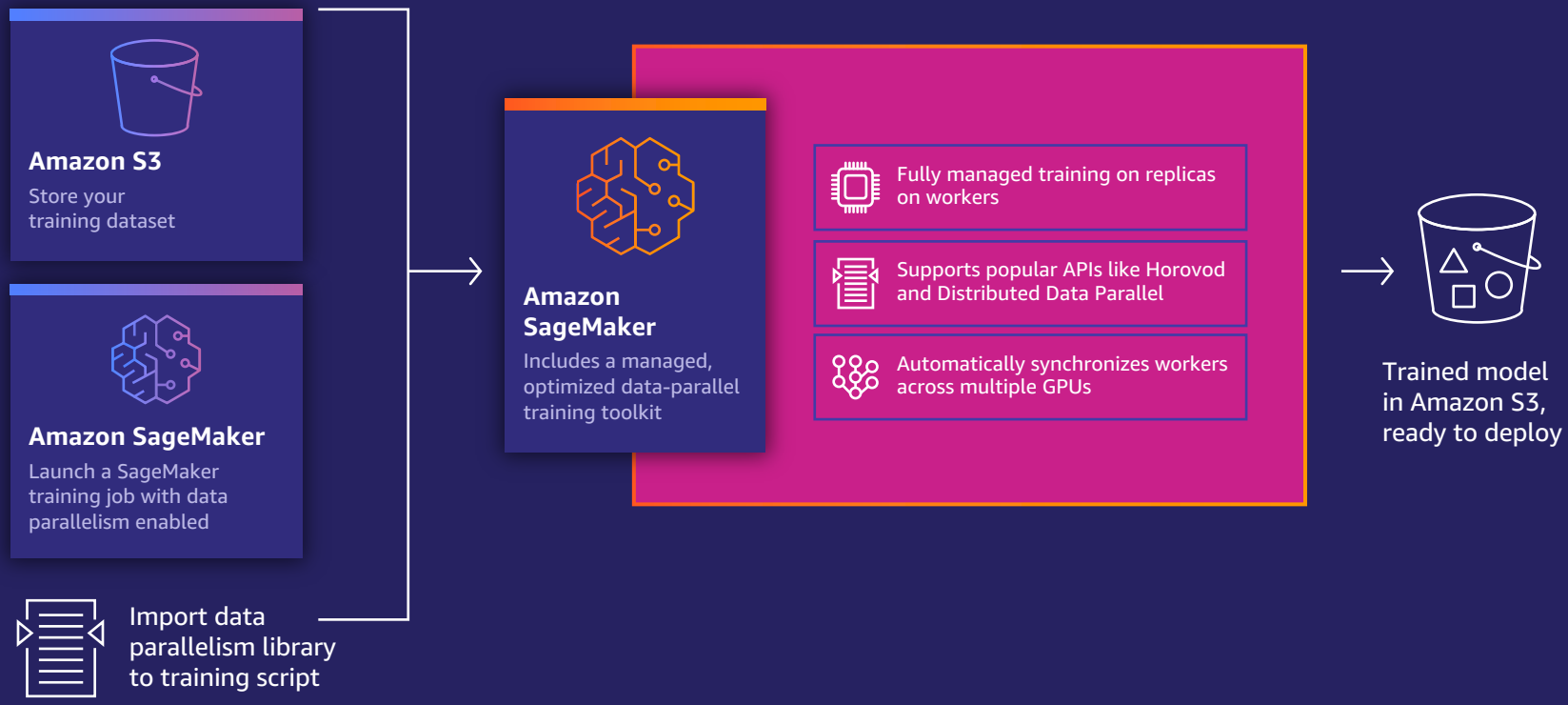
Implement model parallelism with fewer than 10 lines of code change

Efficient pipelining

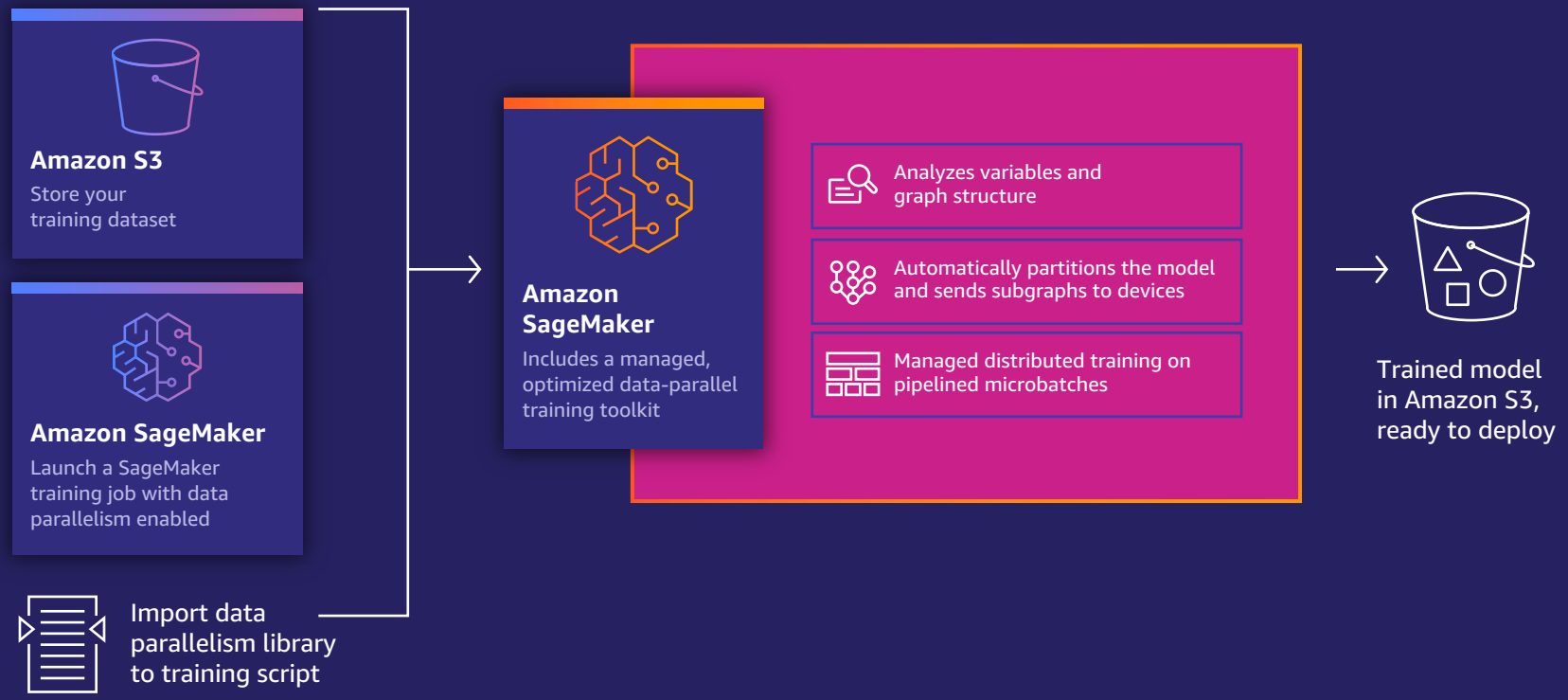
Maximize resource usage with pipelining of micro-batches that keeps all GPUs active

aws machine learning

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

aws machine learning

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

aws machine learning

SageMaker MLOps & ML Platform Intro

aws machine learning

Please refer to deck provided Day1-Part3

aws machine learning

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

Training

- https://docs.aws.amazon.com/sagemaker/latest/dg/train-model.html
- https://docs.aws.amazon.com/sagemaker/latest/dg/distributed-training.html
- https://aws.amazon.com/sagemaker/debugger

Deployment

- https://docs.aws.amazon.com/sagemaker/latest/dg/realtime-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

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

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

aws machine learning

Survey!

Thank you!