Agenda

Overview of AI/ML operation on AWS (90 mins)

- Introduction to AWS ML services
- Train & deploy a ML model on AWS
- Quick demo

TNC215

Amazon SageMaker AI

VPBank Hackathon Hack2Hire – 18th Oct 2025

Tung Cao

Solutions Architect
Amazon Web Services

The AWS AI/ML stack

BROADEST AND MOST COMPLETE SET OF MACHINE LEARNING CAPABILITIES

AI SERVICES

SPECIALIZED

BUSINESS PROCESSES
- Amazon Personalize
- Amazon Forecast
- Amazon Fraud Detector
- Amazon Lookout for Metrics

SEARCH
- Amazon Kendra

CONVERSATION
- Amazon Lex
- Amazon Transcribe Call Analytics
- Contact Lens
- Voice ID

CODE + DEVOPS
- Amazon CodeGuru
- Amazon CodeWhisperer
- Amazon DevOps Guru

INDUSTRIAL
- Amazon Monitron
- Amazon Lookout for Equipment
- Amazon Lookout for Vision

HEALTH
- Amazon HealthLake
- Amazon Comprehend Medical
- Amazon Transcribe Medical
- Amazon Omics

CORE

TEXT
- Amazon Translate
- Amazon Comprehend

SPEECH
- Amazon Polly
- Amazon Transcribe

VISION
- Amazon Textract
- Amazon Rekognition
- AWS Panorama

AMAZON SAGEMAKER

CANVAS
No-code ML for business analysts

STUDIO LAB
Learn ML

GROUND TRUTH
Label data

STUDIO IDE
- Prepare data
- Geospatial ML
- Build with notebooks
- Train models
- Deploy in production
- Store features
- Tune parameters
- Manage and monitor

CI/CD | GOVERNANCE | RESPONSIBLE ML

EDGE MANAGER
Manage edge devices

ML FRAMEWORKS & INFRASTRUCTURE

PyTorch, Apache MXNet, TensorFlow | Amazon EC2 | CPUs | GPUs | AWS Inferentia | AWS Trainium | Habana Gaudi | FPGA

Overcoming the barriers to AI and ML adoption

Disparate data science tools

Accelerate model delivery with open-source FMs available in a hub

Ability to create high-quality labelled datasets

Managing underlying infrastructure

Tedious, manual ML operations

Challenging to govern gen AI and ML projects efficiently

Overcoming the barriers to AI and ML adoption

DISPARATE DATA SCIENCE TOOLS

Integrated ML tools in a single interface

Build, train, and deploy models using IDEs

ACCELERATE MODEL DELIVERY WITH PUBLICLY AVAILABLE FMS AVAILABLE IN A HUB

Choice of FMs

Access 250+ FMs that can be customized easily

ACCESS TO HIGH-QUALITY LABELLED DATASETS

Purpose-built data labeling workflows

Create high quality training datasets to improve model accuracy

MANAGING UNDERLYING INFRASTRUCTURE

Fully managed ML infrastructure

Purpose-built accelerators for deep learning training and inference

TEDIOUS, MANUAL AIML OPERATIONS

Built-in MLOps

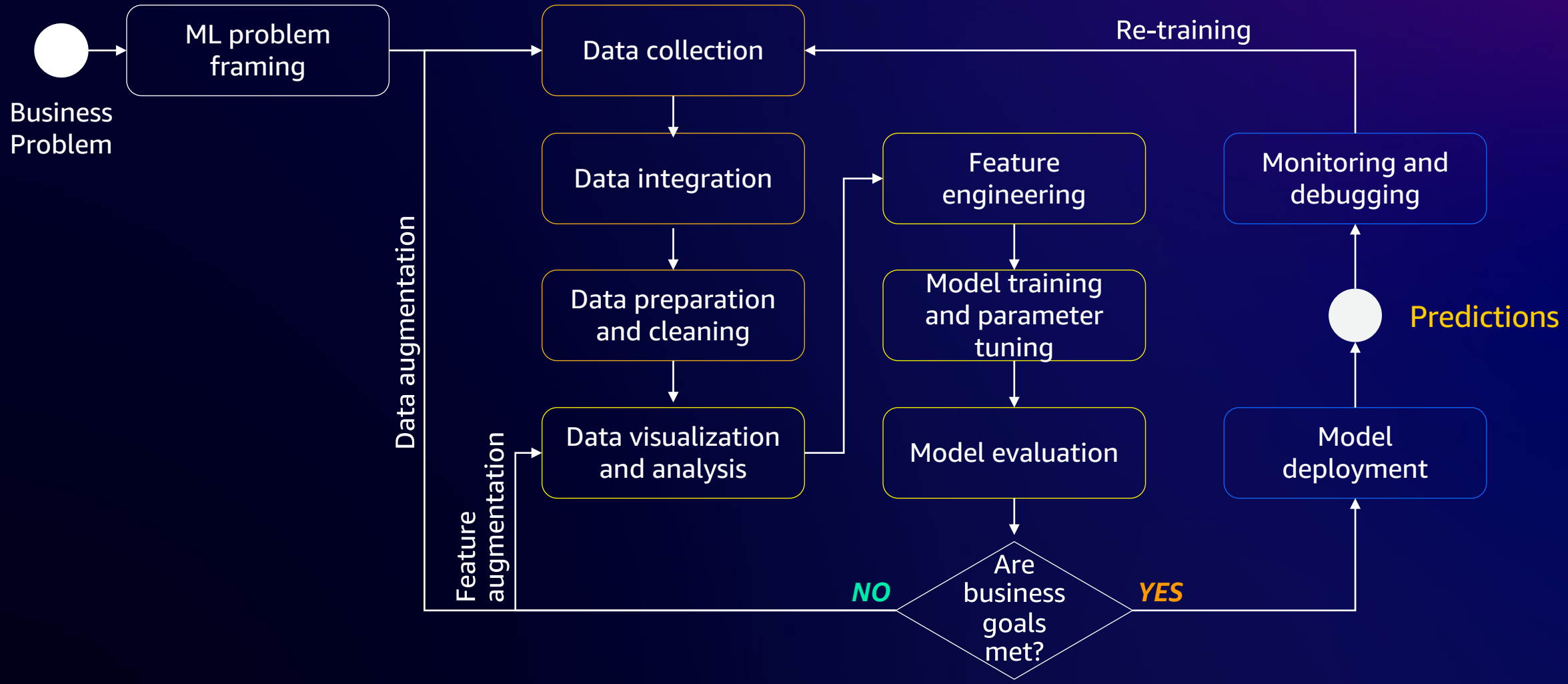
Automate and standardize MLOps practices

CHALLENGING TO GOVERN ML PROJECTS EFFICIENTLY

Out-of-box ML governance tools

Simplify access control and enhance transparency across ML lifecycle

Machine learning Solution Lifecycle



Amazon SageMaker AI and Amazon Bedrock

SageMaker AI

- Model development
- Build and customize Foundational Models using advanced techniques
- Configurable model deployment and inference
- Code-based IDE
- MLOps and FMOps

Use a model in SageMaker AI together with another model in Bedrock

Fine-tune a model in SageMaker AI and import to Bedrock for further customization and other use cases

Experiment using Bedrock, and move to production using SageMaker AI for control over cost, throughput, and latency

Bedrock

- Application development
- Built-in tooling for customization with RAG
- Built-in tooling for agentic workflow
- Access to Claude, Amazon FMs, and 3P providers via API calls
- Responsible AI

Amazon SageMaker AI

Build, train, and deploy ML models at scale, including FMs

Build FMs from scratch

Create your own ML models, including FMs, with integrated purpose-built tools and high-performance, cost-effective infrastructure

Customize foundation models

Access and evaluate 250+ FMs that can be customized easily for your use case

Implement MLOps and governance

Create reliable and repeatable workflows incorporating MLOps practices with purpose-built tooling

Improve ML governance

Enhance model governance and compliance with built-in governance tools

Manage and deploy models for inference

Easiest way to deploy AI & ML models including foundation models (FMs) to make inference requests at the best price performance for any use case

SageMaker AI supports ML, Deep Learning and GenAI

AMAZON SAGEMAKER AI

Machine Learning (Tabular Inputs)	Deep Learning (Unstructured Inputs)	Gen AI (Unstructured outputs)
Predictive maintenance	Computer vision	Summarization
Financial risk prediction	Meta data enrichment	Information extraction
Demand forecasting	Sentiment analysis	Visual content generation
Fraud detection	Topic modelling	Code generation
Churn prediction	Intelligent data processing	Audio/music generation
Personalized recommendations	Autonomous driving	Synthetic data generation

Classic ML and GenAI with Amazon SageMaker AI

CLASSIC ML

Data Prep

Prepare data assets and pipelines, Manage data quality and bias

Build

Experiment and automate the execution of build pipelines

Train

Train ML models at scale with automation

Deploy

Automate the execution of deploy pipelines into production

GENAI

Data Prep

Pre-train or Select

Select from FM hub or BYOFM, or Pre-train FM

Evaluate

Automated or human evaluation

Fine-tune

Fine-tune and/or optimize models for deployment

Deploy

AUTOMATE WITH AI OPS AND GOVERNANCE

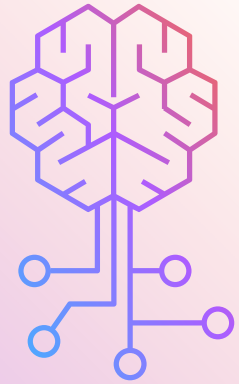
Amazon SageMaker AI simplifies

Model Building

Model Deployment

Model Training and Fine-Tuning

Scalable MLOps and governance



Model Building

Single, fully managed IDE for notebooks, code, and data

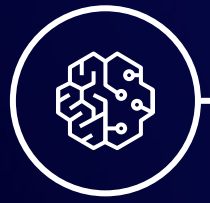
Build ML models

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 AI 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

Recommendation

Factorization machines

Forecasting

DeepAR

Sequence translation

Seq2Seq

Regression

Linear Learner | XGBoost | KNN

Clustering

KMeans

Anomaly detection

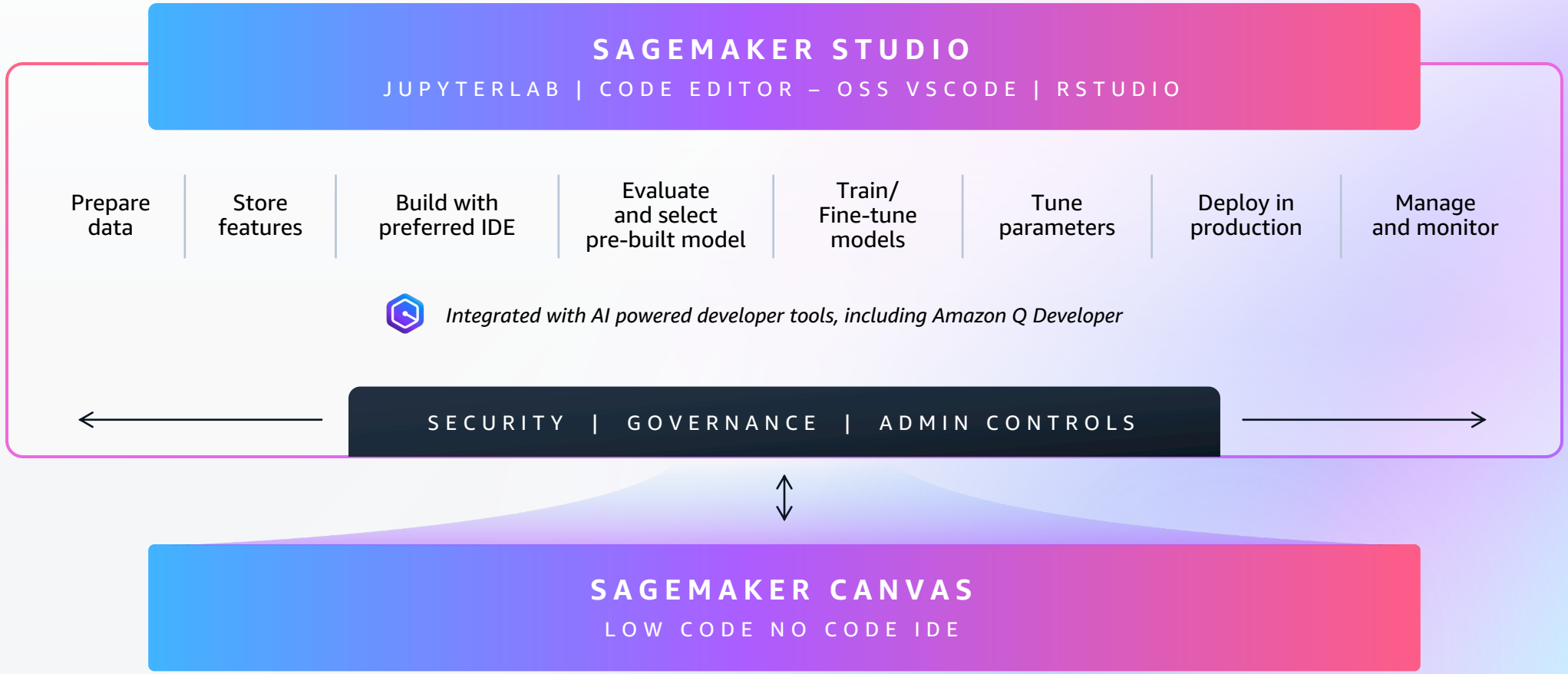
Random cut forests | IP Insights

Feature reduction

PCA

AMAZON SAGEMAKER STUDIO

Tools for every step of the ML lifecycle under one unified visual user interface



AMAZON SAGEMAKER STUDIO

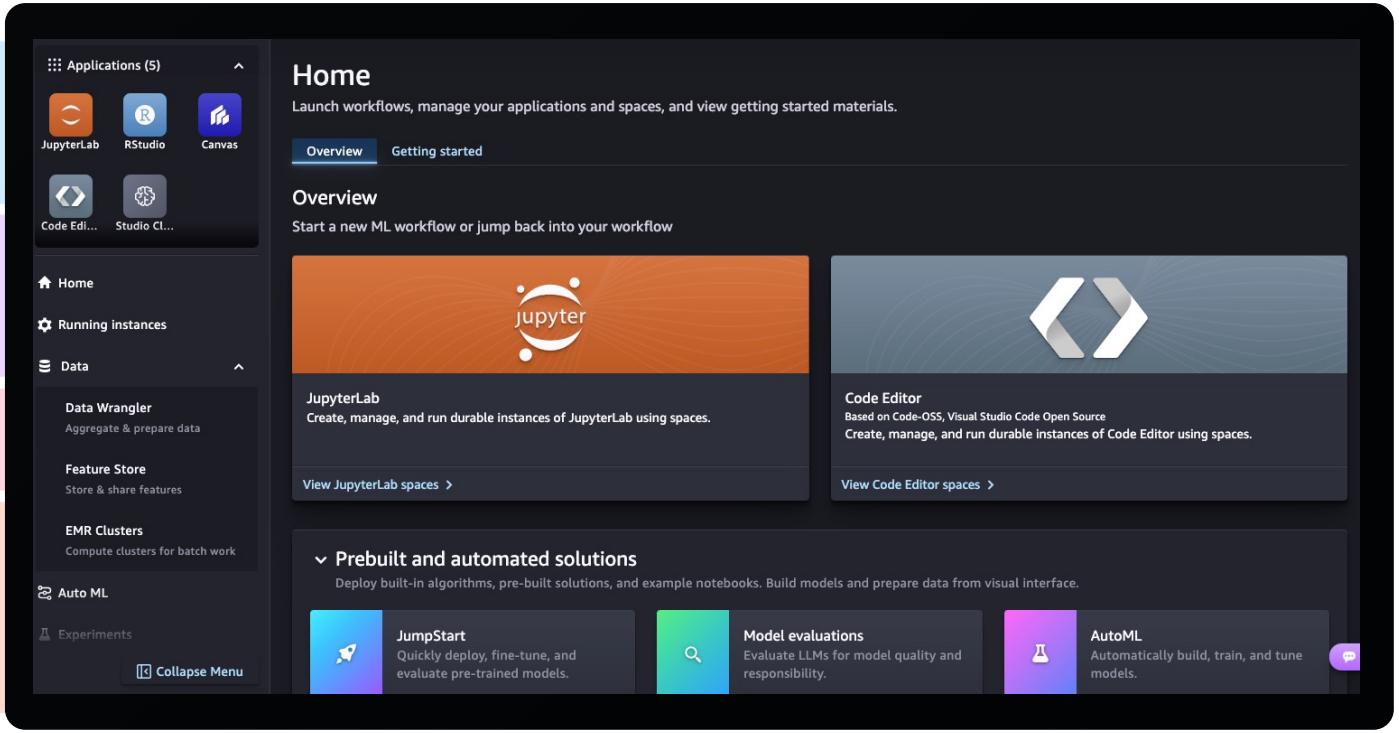
Tools for every step of the model development lifecycle under one unified visual user interface

Choice of fully managed IDE's

Purpose-built tools for AI/ML development w/ genAI support

Securely run anywhere

Specialized genAI and ML development apps from leading partners



Amazon SageMaker Studio

NEW

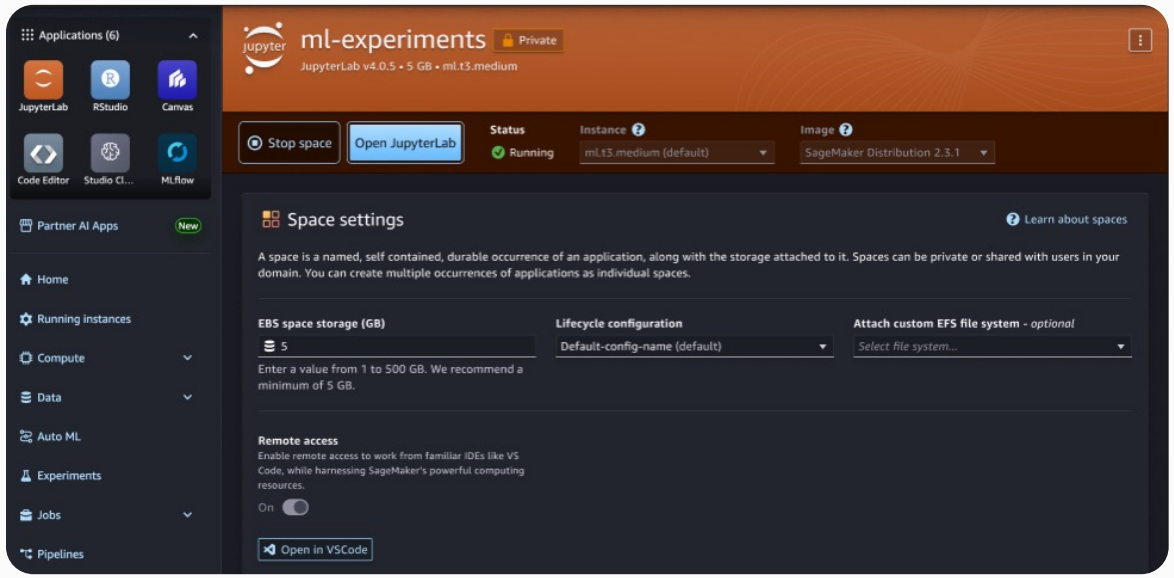
Amazon SageMaker Studio Connect to Local IDE's

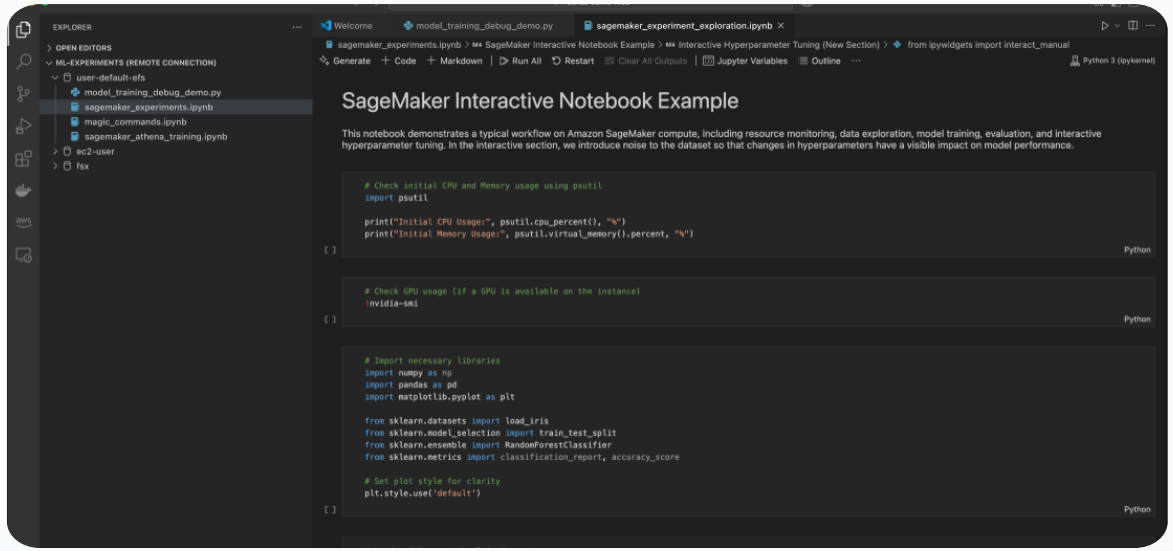
Maximize productivity with simplified access and enhanced security

Seamless one-click connectivity to SageMaker Studio spaces with your preferred local IDE, such as VS Code

Leverage SageMaker AI's powerful compute resources and governed data to analyze, process data and develop ML models

Maintain all existing SageMaker AI security controls and permissions without additional configuration



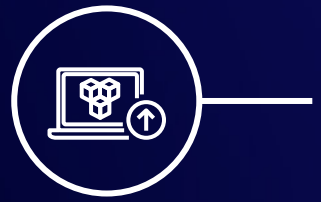


Amazon Sagemaker Data Wrangler

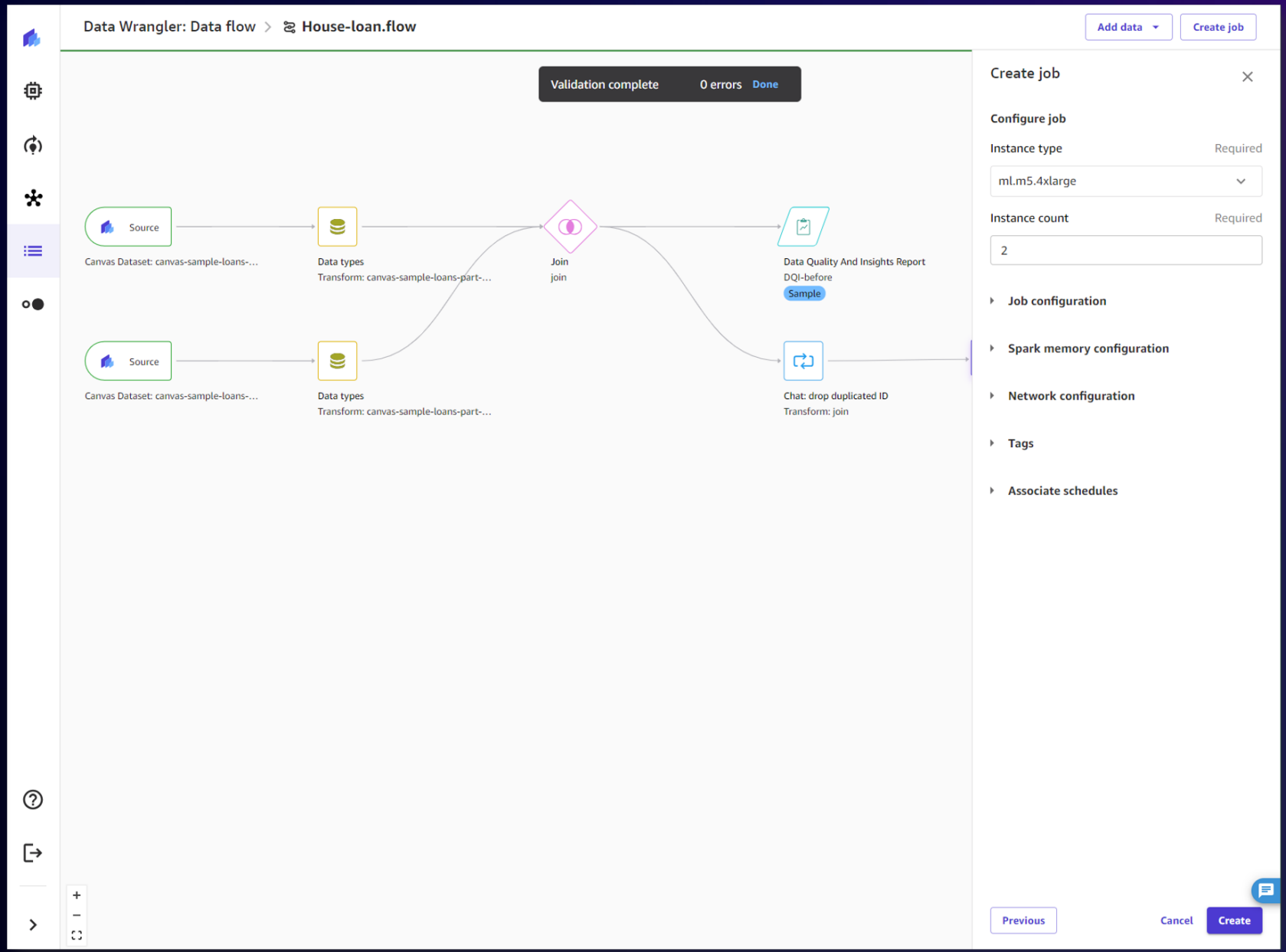
No-code data preparation

Single visual interface for common data prep techniques

Select data from multiple sources



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



Amazon Sagemaker AI Processing Job

Managed solution for data processing and model evaluation jobs

Achieve distributed processing for clusters

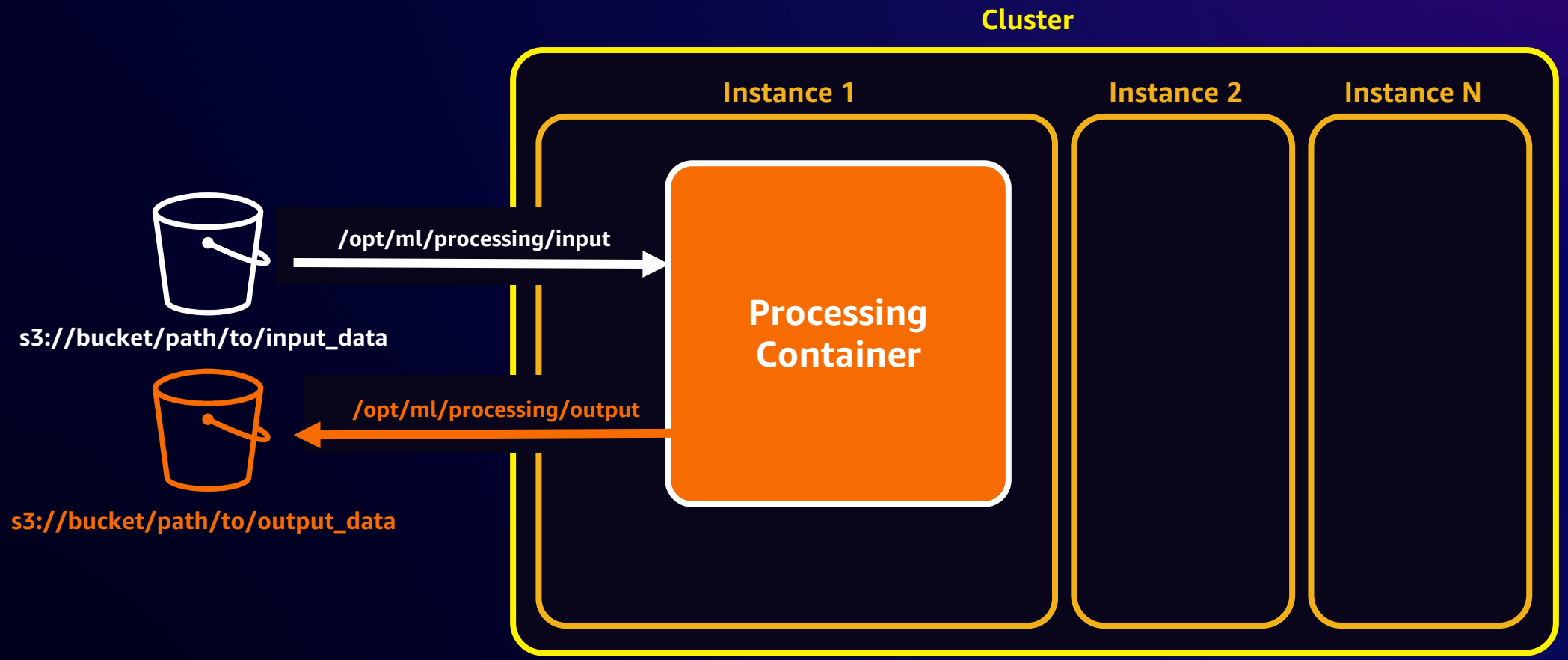
Bring your own script for feature engineering

Use Sagemaker AI's built-in containers or bring your own

Leverage Sagemaker AI's security and compliance features

Your resources are created, configured, and terminated automatically

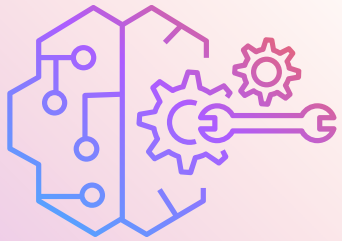
Data Processing on Amazon Sagemaker AI



Data Processing on Amazon Sagemaker AI

```python
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
)
```

```python
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"),
    ],
)
```



Model Training and Fine-Tuning

Fast and cost-efficient ML and Gen AI model training

SageMaker AI offers two training options

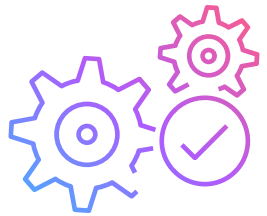
Purpose-built infrastructure for FM training

Fully managed training jobs

Fully managed resilient infrastructure for large-scale and cost-effective training

Focus on model building rather than IT

Provide access to flexible on-demand GPU cluster with a pay as you go option



Amazon SageMaker HyperPod

Resilient and **self orchestration** infrastructure for maximum resource control

Customize and manage cluster orchestration (Slurm or EKS)

Schedule workloads to maximize cluster utilization across teams

Fully managed training jobs

Fast and cost-effective AI and ML model training

Many ways to train models

Local mode, script, BYOC, deep learning containers

Purpose built infrastructure for large scale model training

Fully managed training jobs for scale and resiliency

Distributed training libraries

With data and model parallel libraries complete distributed training 40% faster

Access an expert on demand workforce

Get human-generated data to customize models, dynamically scale your workflows, and help meet your security, privacy, and compliance requirements

Debug and profile training runs, automatic model tuning and fine tuning

Use real-time metrics to correct performance problems and find the best version of a model with automatic hyperparameter optimization

Experiment management and managed spot training

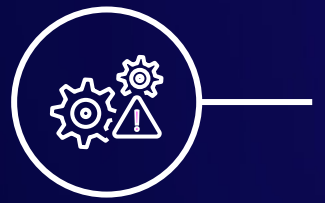
Track ML model iterations easily by automatically capturing input parameters, configurations, and results. Reduce costs by up to 90% by automatically running training jobs when compute becomes available

Automate training workflows

Create a repeatable process to orchestrate the steps for rapid experimentation and model retraining

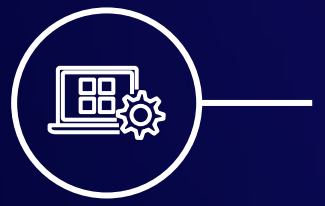
Train ML models

Fast and cost-effective ML model training



Experiment management and model tuning

Save weeks of effort by automatically tracking training runs and tuning hyperparameters

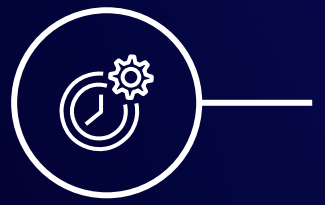


Debug and profile training runs

Use real-time metrics to correct performance problems

Distributed training

Complete distributed training up to 40% faster



Training compiler

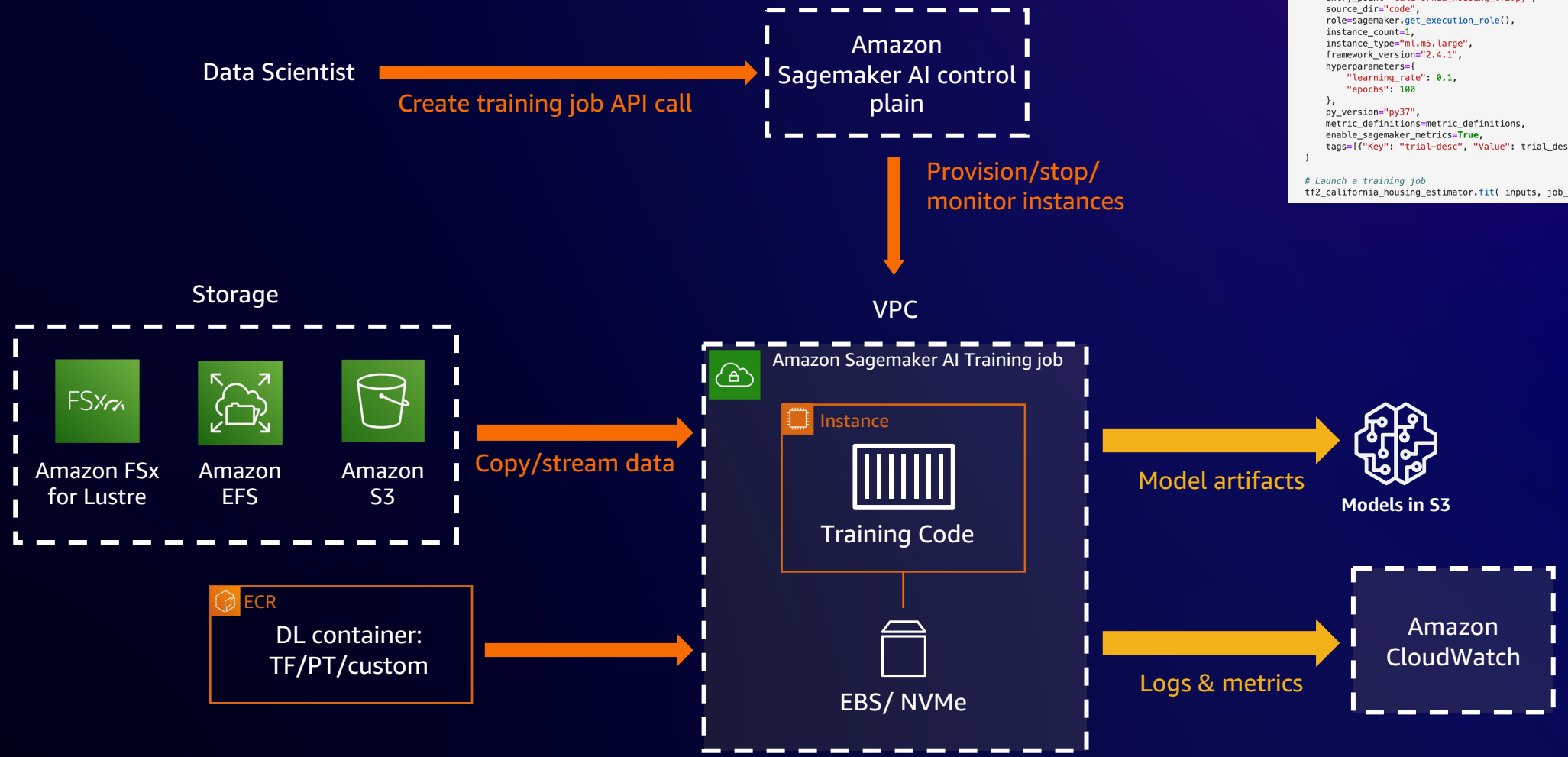
Accelerate training times by up to 50% through more efficient use of GPUs

Managed spot training

Reduce the costs of training by up to 90%

Training on Amazon Sagemaker AI

```python
# 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)
```



Training Estimator

```
# 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 AI 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

Example

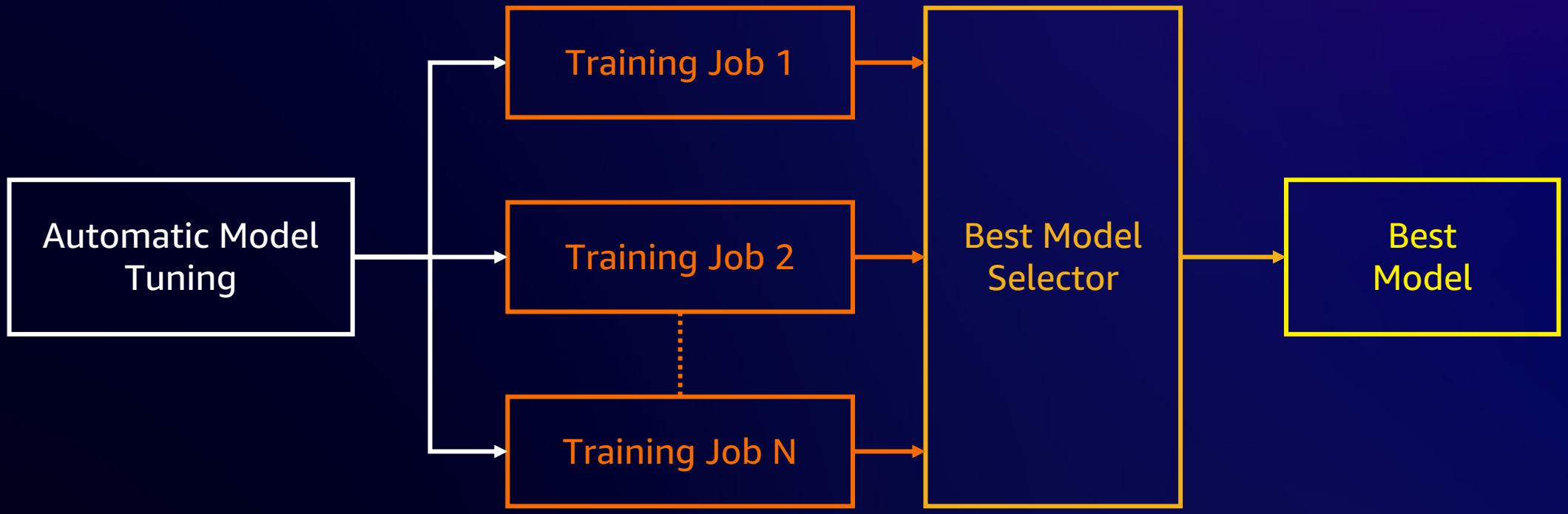
Decision trees
Tree depth | Max leaf nodes | Gamma | Eta | Lambda | Alpha

Example

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

Amazon Sagemaker AI 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'
```

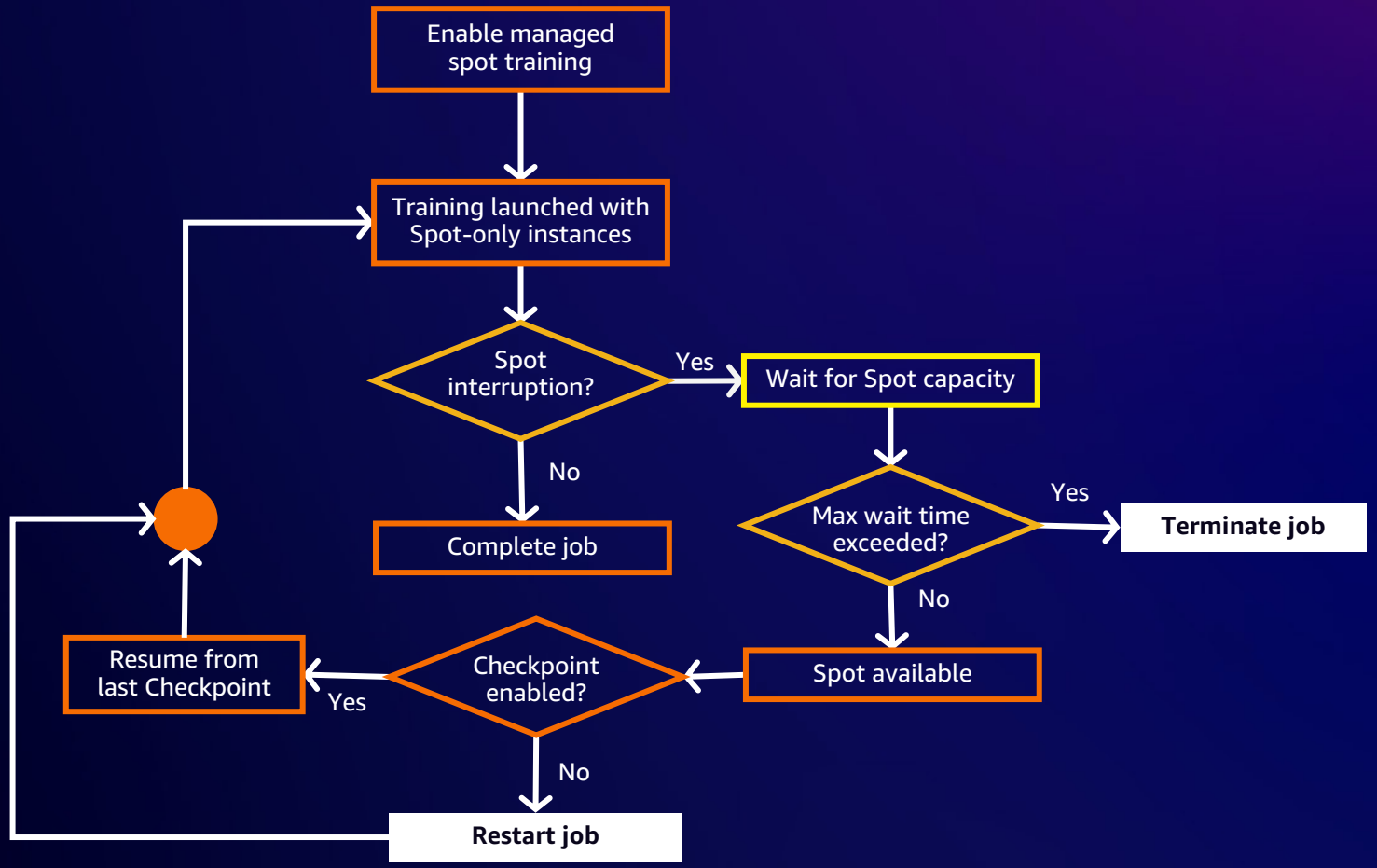
3. Pick job parameters

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

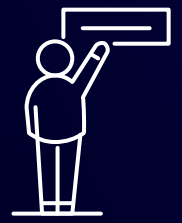
Managed Spot Training

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

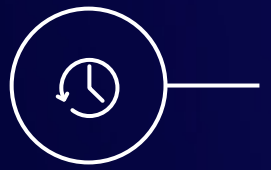
Charged for the run duration before completion or termination; not charged for idle time, billing starts when instances are ready

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

* Checkpointing is a best practice and is highly recommended

Distributed training

The fastest and easiest way to train large deep learning models

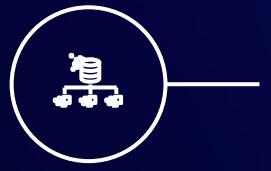


Reduced training time

Reduce training time by 25% with synchronization across GPUs

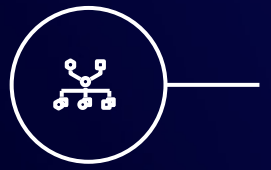
Optimized for AWS

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

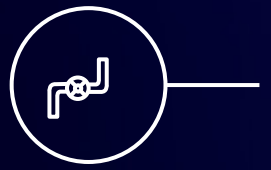


Automatic and efficient model partitioning

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

Model Deployment

Easily deploy AI and ML models — From low latency and high throughput to long-running inference

Amazon SageMaker JumpStart

Model hub with foundation models, built-in algorithms, and prebuilt AIML solutions that you can deploy with just a few clicks

ML hub with foundation models

Access hundreds of foundation models including top open-weight and proprietary models that can be fine tuned easily for your use case

Ease of use

Easily use pre-trained models on SageMaker AI instances like Inf2 and optimized hosting configurations through presets

Evaluate and customize

Evaluate, fine-tune, and optimize deployment using a few clicks

Data security and access control

Keep inference and training data private and curate who can access and use models within your organization

Share and collaborate within your organization

Share models and notebooks with others within your organization, and allow them to train with their own data or deploy as-is for inferencing

Amazon SageMaker JumpStart

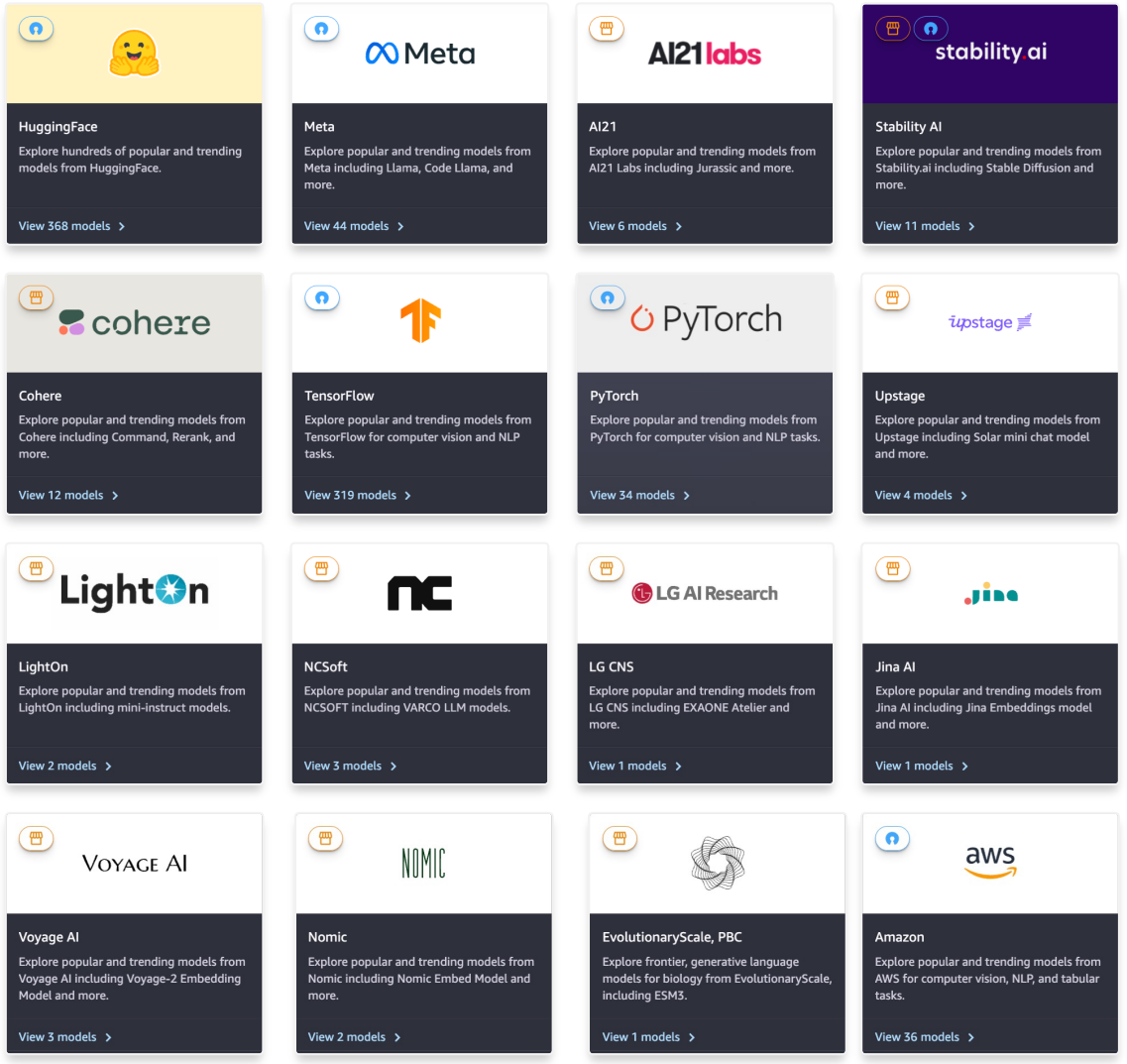
Model hub with foundation models, built-in algorithms, and prebuilt AIML solutions

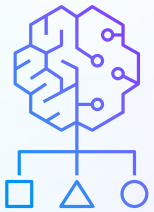
Over 250+ publicly available foundation models

Hundreds of built-in algorithms with pretrained models form popular model hubs

Fully customizable solutions for common use cases with reference architectures

Share AI and ML models and notebooks across your organization





Deploy AI and ML models

Fully managed deployment for inference at scale

Wide selection of infrastructure

70+ instance types with varying levels of compute and memory to meet the needs of every use case. Deliver up to 40% better inference price performance with Inf2 instances

Deploy models in production for inference for any use case

From low latency and high throughput to long-running inference

Cost-effective deployment

Reduce inference cost by at least 50% with multi-model/multi-container endpoints, serverless inference, and elastic scaling

Shadow testing & automatic deployment guardrails

Validate the performance of new ML models against production models. Minimize risk when deploying new model versions on SageMaker AI using linear, canary, or blue green traffic switching

Built-in integration for MLOps

ML workflows, CI/CD, feature management, lineage tracking, and model management

Large model inference container

Achieve best price performance with the latest inference optimizations tools, model servers, and libraries packaged into a single container

Model deployment on Amazon SageMaker AI

Real-time synchronous response

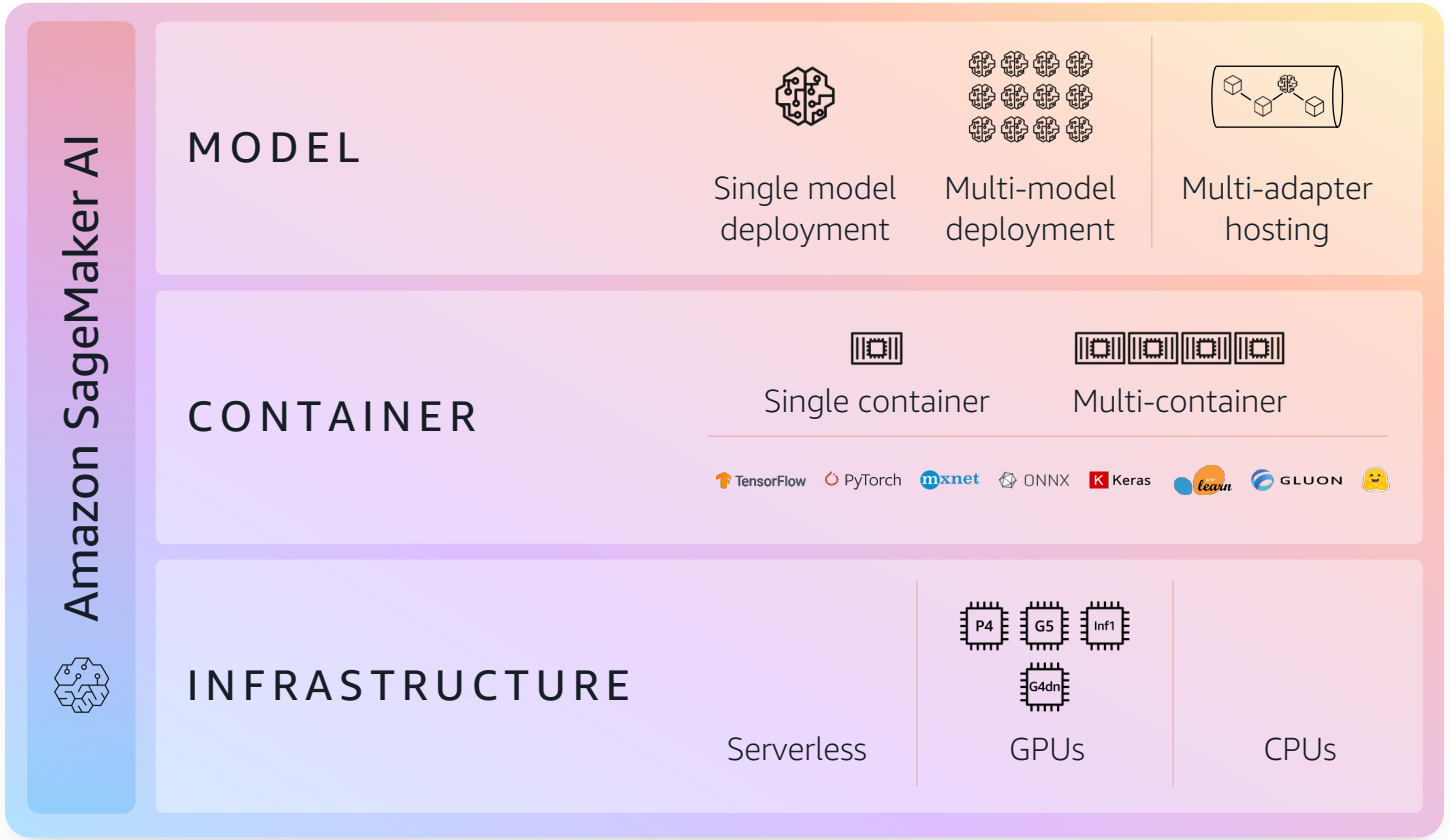
Invoke →
← Response

Near real-time asynchronous response

Invoke →
← Response

Offline batch inference

Submit →
← Complete



Deploy models to serve inference

AMAZON SAGEMAKER

SageMaker Studio

Real-time inference | Async inference | Serverless inference | Batch inference | Multi-model endpoints | Multi-container endpoints | Inference pipelines | Manage and version models | MLOps | Model monitoring | Shadow Testing | Metrics and logging in CloudWatch

SageMaker JumpStart

Containers

TensorFlow | PyTorch | mxnet | Keras | scikit-learn | GLUON | Hugging Face | ONNX | LMI | DJL DeepJavaLibrary | NVIDIA | BYOC

ML Compute Instances and Accelerators

CPUs | GPUs | Inferentia | Graviton (ARM)

Deep learning compilers and runtimes

SageMaker Neo | NVIDIA TensorRT/ cuDNN | Intel oneDNN | ARM Compute Library

Sagemaker AI inference options

Real-time inference

Low latency

Ultra high throughput

Multi-model endpoints

A/B testing

Batch transform

Process large datasets

Job-based system

Asynchronous inference

Near real-time

Large payloads (1 GB)

Long timeouts (15 mins)

Serverless Inference Fully managed offering

Managed infrastructure

Security

Monitoring

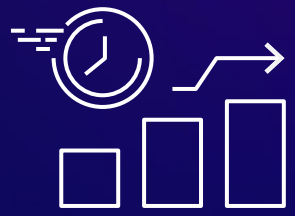
Logging

Built in availability and fault tolerance

Serverless

No need to select instance types or provision capacity

Choose memory options based on inference processing needs

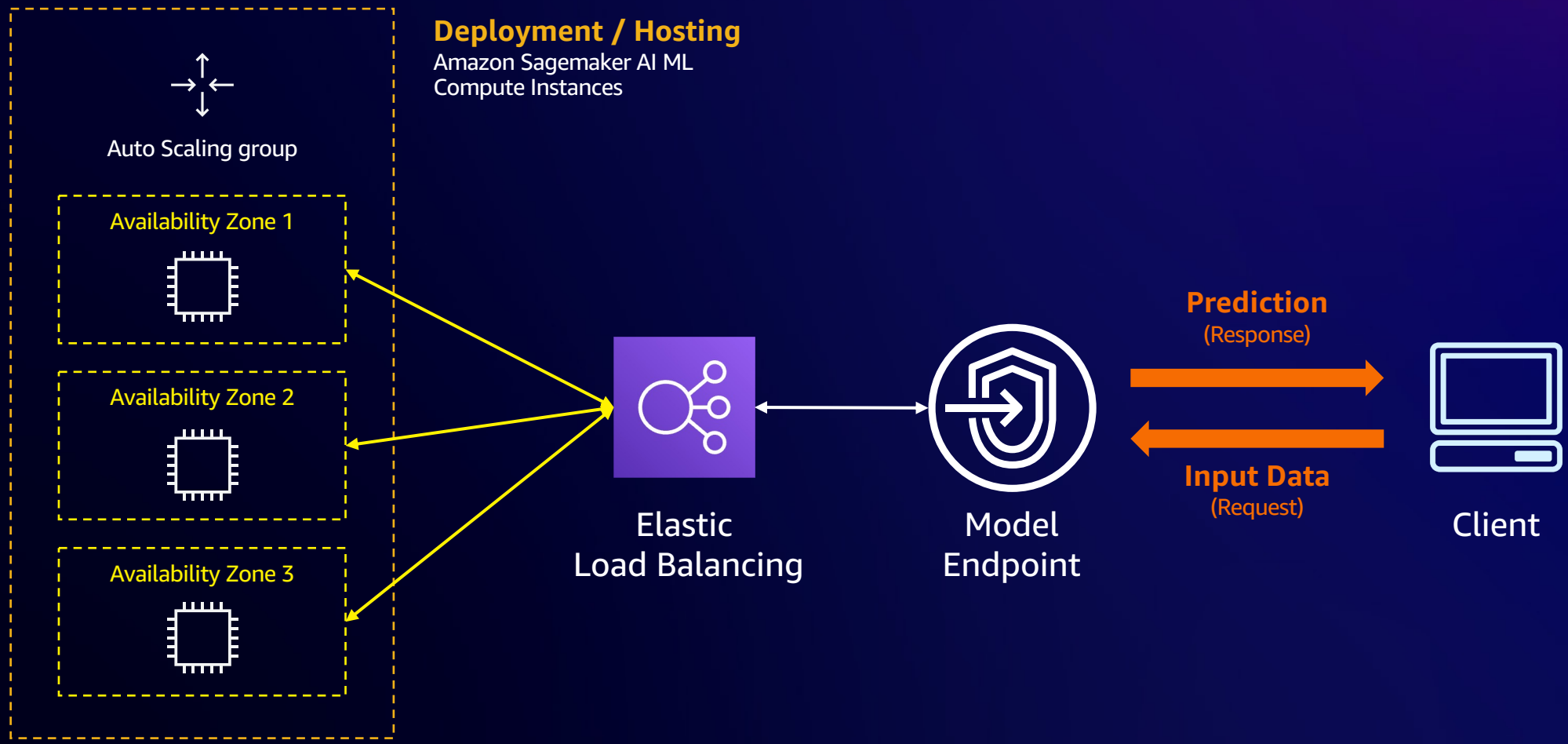


Automatically scale out, in, and down to 0

No need to set scaling policies

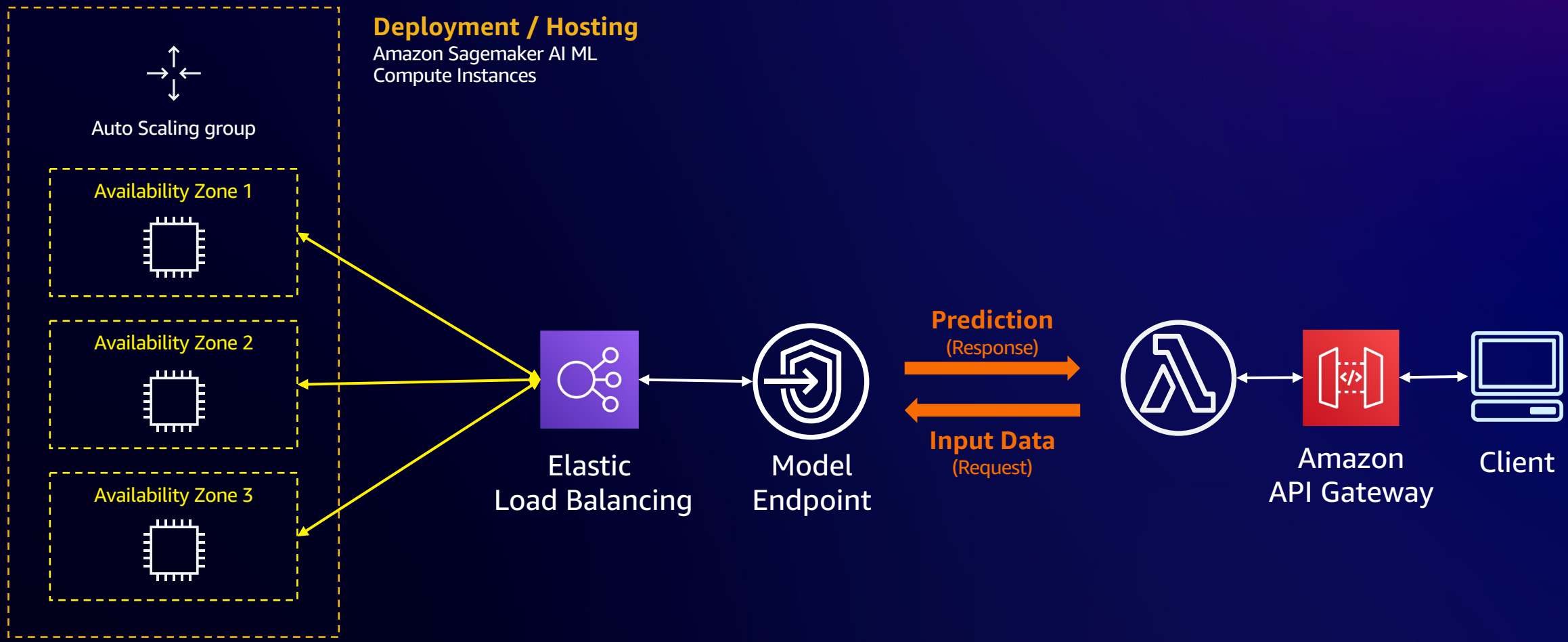
Amazon Sagemaker AI Deployment

SAGEMAKER AI ENDPOINTS (PRIVATE API)



Amazon Sagemaker AI Deployment

SAGEMAKER AI ENDPOINTS (PUBLIC API)



Real time Inference

```
from time import gmtime, strftime
from sagemaker.tensorflow.model import TensorFlowModel

tensorflow_model = TensorFlowModel(model_data=model_path,
                                   role=role,
                                   framework_version="2.3.1")

endpoint_name = "DEMO-tf2-california-housing-model-monitor-" + strftime(
    "%Y-%m-%d-%H-%M-%S", gmtime()
)

predictor = tensorflow_model.deploy(
    initial_instance_count=1,
    instance_type="ml.m5.xlarge",
    endpoint_name=endpoint_name,
)
```

Serverless Inference

Step1 : Create Model

```
from time import gmtime, strftime

model_name = "xgboost-serverless" + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print("Model name: " + model_name)

# dummy environment variables
byo_container_env_vars = {"SAGEMAKER_CONTAINER_LOG_LEVEL": "20", "SOME_ENV_VAR": "myEnvVar"}

create_model_response = client.create_model(
    ModelName=model_name,
    Containers=[
        {
            "Image": image_uri,
            "Mode": "SingleModel",
            "ModelDataUrl": model_artifacts,
            "Environment": byo_container_env_vars,
        }
    ],
    ExecutionRoleArn=role,
)

print("Model Arn: " + create_model_response["ModelArn"])
```

Step2 : Create Endpoint Configuration

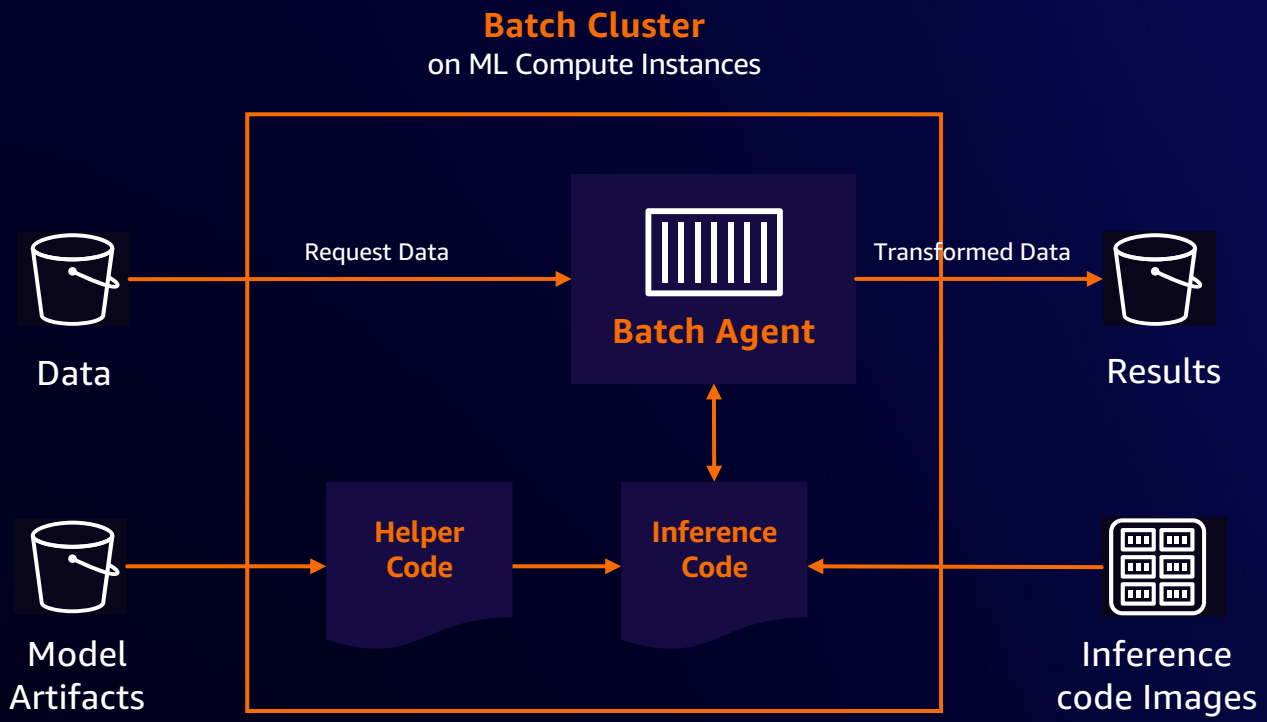
```
xgboost_epc_name = "xgboost-serverless-epc" + strftime("%Y-%m-%d-%H-%M-%S", gmtime())

endpoint_config_response = client.create_endpoint_config(
    EndpointConfigName=xgboost_epc_name,
    ProductionVariants=[
        {
            "VariantName": "byoVariant",
            "ModelName": model_name,
            "ServerlessConfig": {
                "MemorySizeInMB": 4096,
                "MaxConcurrency": 1,
            },
        },
    ],
)

print("Endpoint Configuration Arn: " + endpoint_config_response["EndpointConfigArn"])
```

Step3 : Create Endpoint

```
endpoint_name = "xgboost-serverless-ep" + strftime("%Y-%m-%d-%H-%M-%S", gmtime())

create_endpoint_response = client.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=xgboost_epc_name,
)

print("Endpoint Arn: " + create_endpoint_response["EndpointArn"])
```

Amazon Sagemaker AI Deployment

BATCH HOSTING



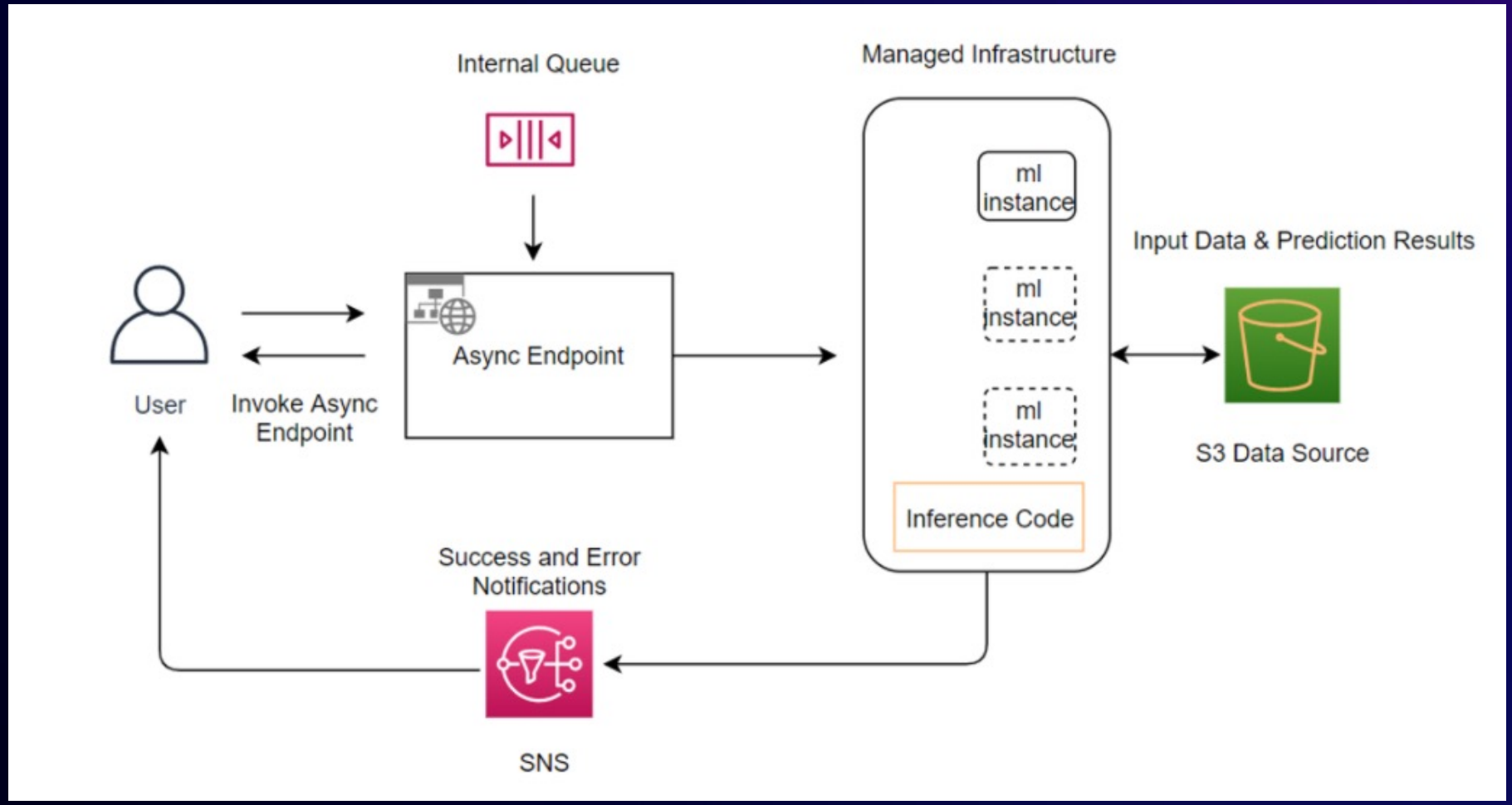
Batch Transform

- Predictions for an entire dataset
- Transient resources (instances provisioned and terminated once job is done)
- No infrastructure to manage
- Can associate prediction results with input
- Supports Built-In/Bring-Your-Own

Batch Inference

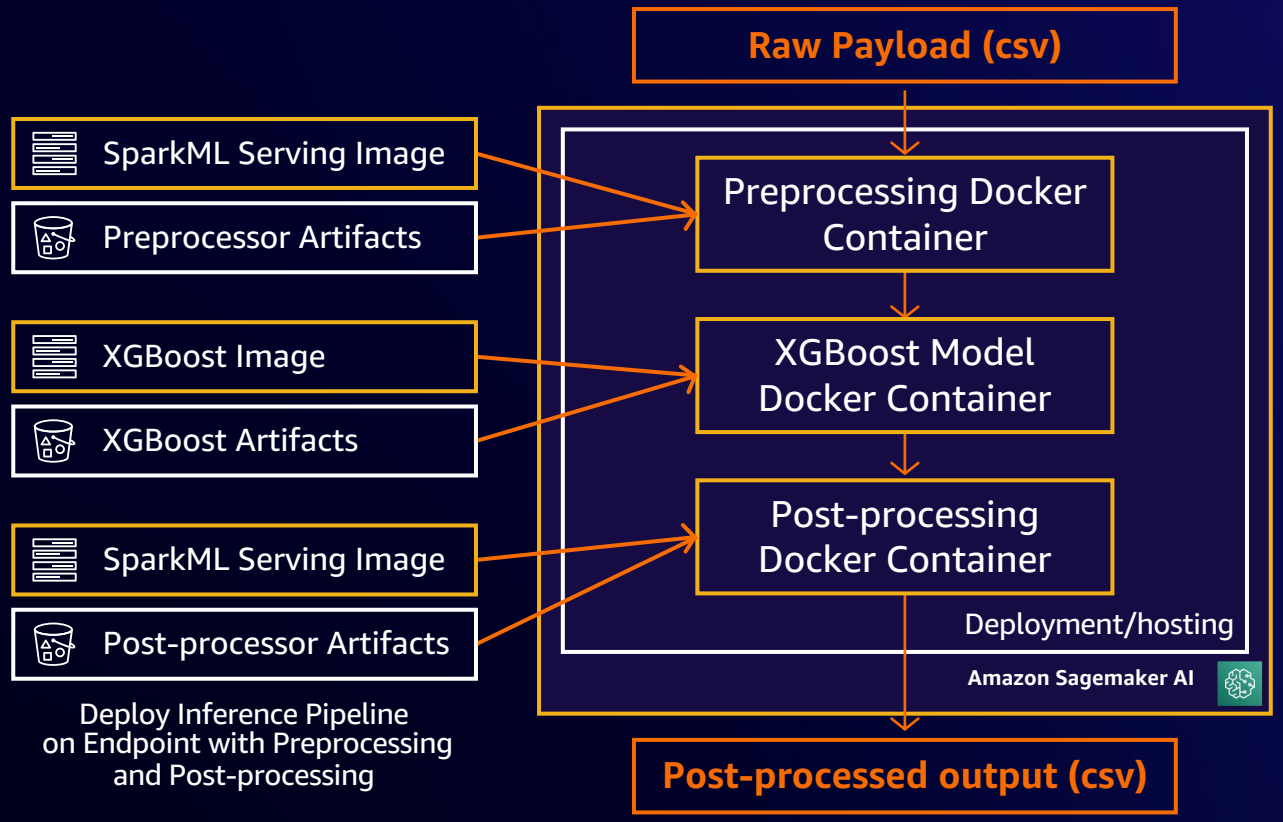
```
from sagemaker.tensorfllow import TensorflowModel

serving = TensorflowModel(
    model_data = 's3 location',
    role=role,
    framework_version="2.3",
    sagemaker_session=sagemaker_session,
    entry_point = 'inference.py',
    source_dir = 'code'
)

input_data_path = "s3://sagemaker-sample-data-{}/tensorflow/california_housing_data/batch.csv".format(
    sagemaker_session.boto_region_name
)
output_data_path = "s3://{}/{}/{}".format(bucket, prefix, "batch-predictions")
batch_instance_count = 2
batch_instance_type = "ml.g4dn.2xlarge"
concurrency = 32
max_payload_in_mb = 1

transformer = serving.transformer(
    instance_count=batch_instance_count,
    instance_type=batch_instance_type,
    max_concurrent_transforms=concurrency,
    max_payload=max_payload_in_mb,
    output_path=output_data_path,
)

transformer.transform(data=input_data_path, content_type="text/csv")
transformer.wait()
```

Async Inference



Inference Pipelines for sequential execution of models

EXECUTE DATA PROCESSING ON INFERENCE REQUESTS,
MAINTAIN SINGLE COPY OF DATA PROCESSING CODE FOR TRAINING AND INFERENCE

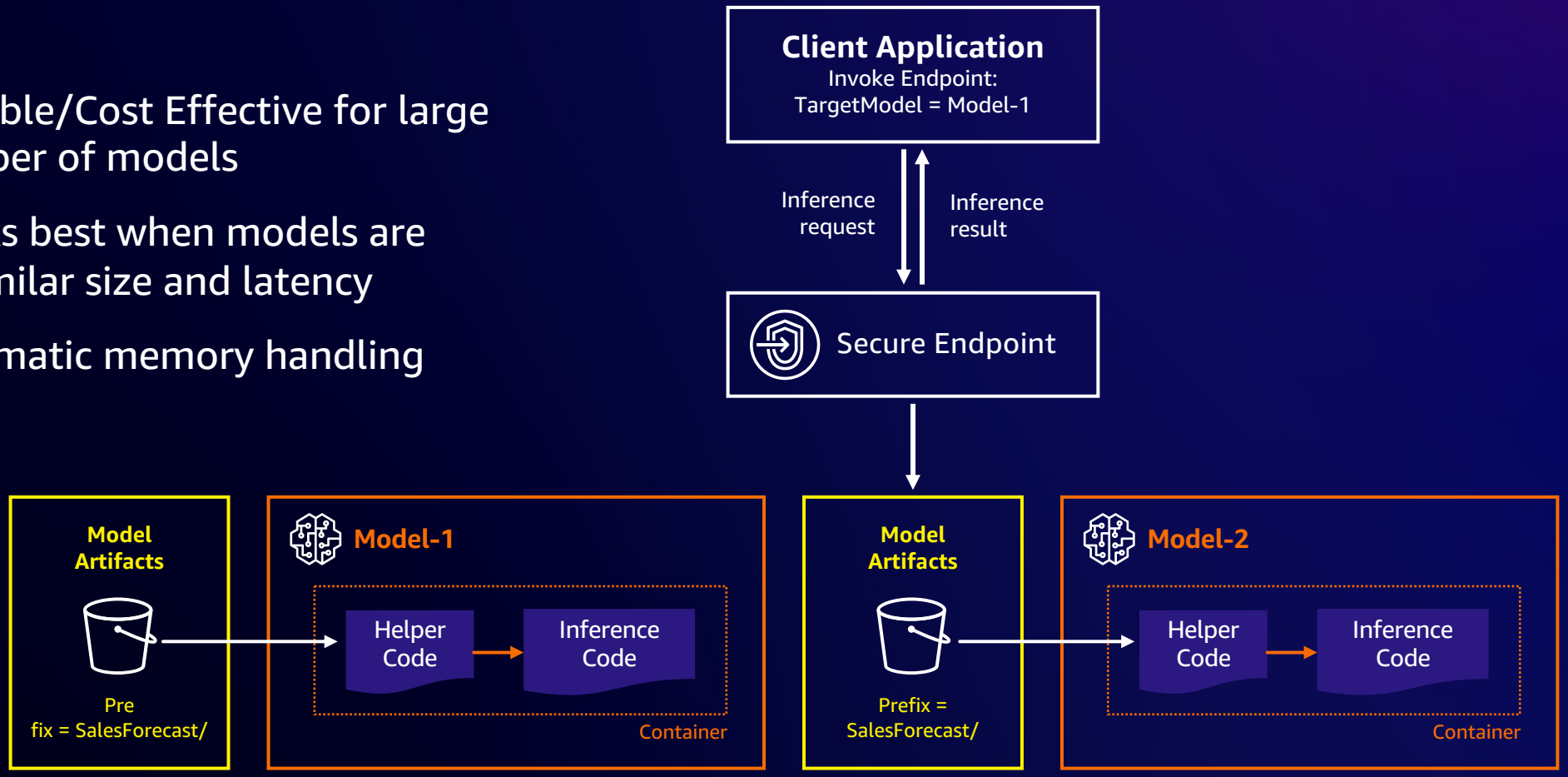


- Built-in containers — Scikit-Learn and Apache Spark MLLib
- Add up to 5 containers; execute sequentially
- Containers co-located on instance for low latency

Amazon Sagemaker AI Deployment

MULTI-MODEL ENDPOINTS

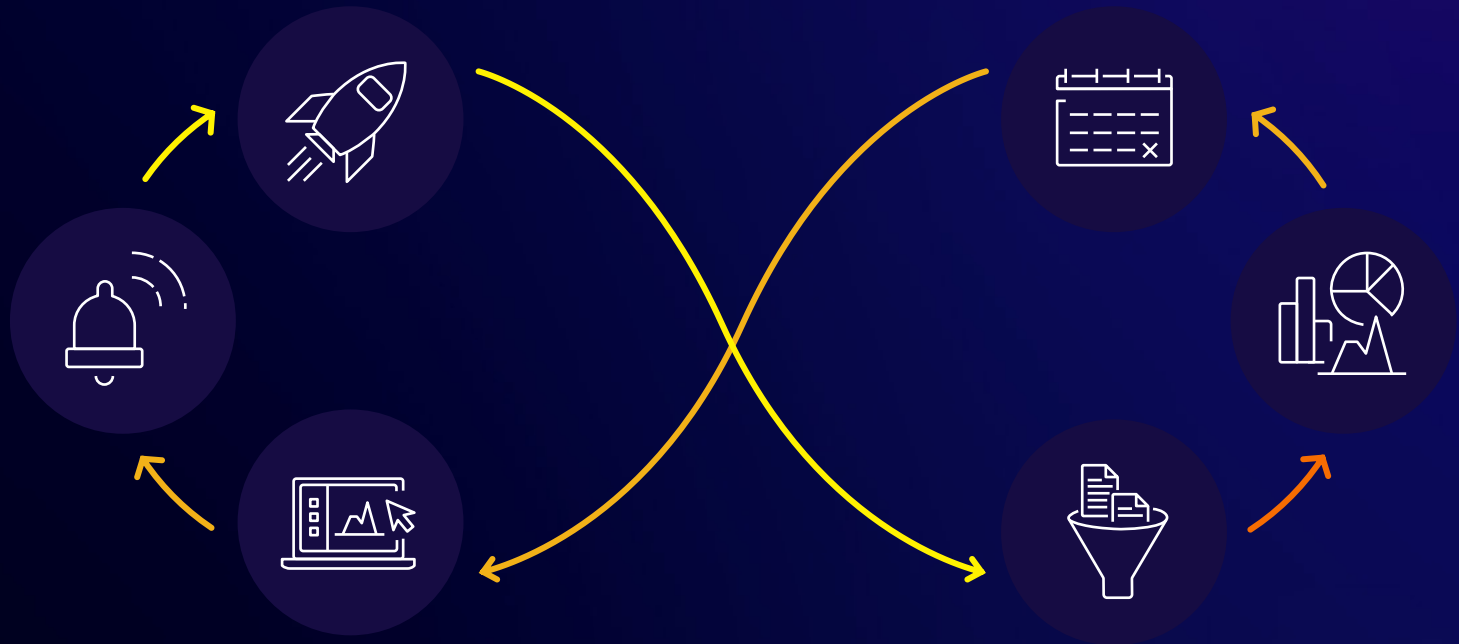
- Scalable/Cost Effective for large number of models
- Works best when models are of similar size and latency
- Automatic memory handling



Model Monitor: how it works

Deploy trained model
Enable data capture on Sagemaker AI Endpoint

Schedule monitoring jobs
Detect and monitor data quality, model accuracy, bias, and explainability drift

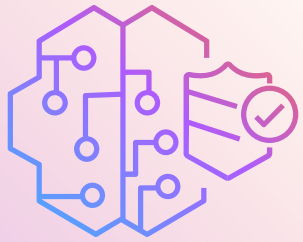


CloudWatch alerts
Retrain models/ update data

Generate baseline
Statistics and constraints

View drift monitoring results
Reports in S3, visualize metrics in Studio, analyze in Notebooks

Collect ground truth
Merge with predictions



Scalable MLOps

Implement reliable MLOps workflows with built-in governance and compliance tools

Ops challenges managing the model lifecycle

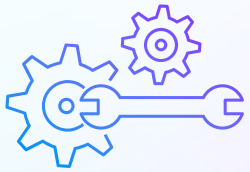
Purpose Built Tools for MLOps and governance

Manual iterative processes slow down ML innovation

Difficult to scale and manage the number of models in production

CI/CD for ML requires writing custom code

Compliance requirements are difficult to meet



Amazon SageMaker MLOps

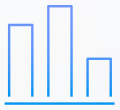
Streamline the ML lifecycle

Automate ML workflows to scale model development

Build CI/CD pipelines for gen AI and ML to improve reliability, quality, and accelerate model deployment

Catalog model versions, metadata, metrics, and approvals for traceability and reusability

Track lineage for traceability and compliance



Maintain accuracy of predictions after models are deployed

NEW

Amazon SageMaker AI now supports fully-managed MLflow 3.0

End-to-end observability and experiment tracking

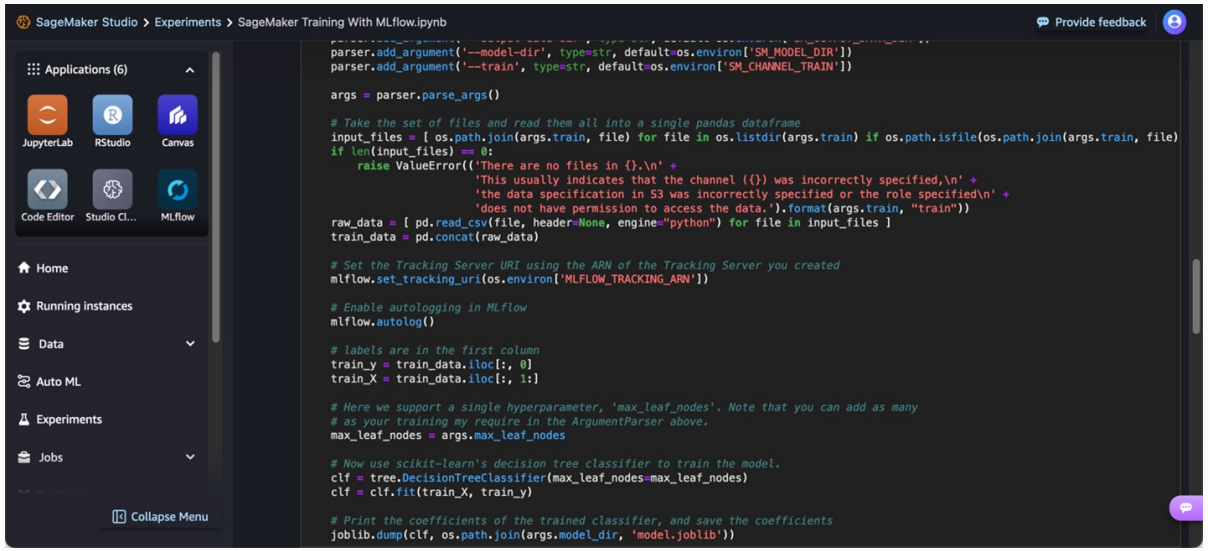
Streamline AI development with unified monitoring from experimentation to production

Advanced tracing capabilities for GenAI applications

Record inputs, outputs, and metadata at every step to quickly identify bugs and unexpected behaviors

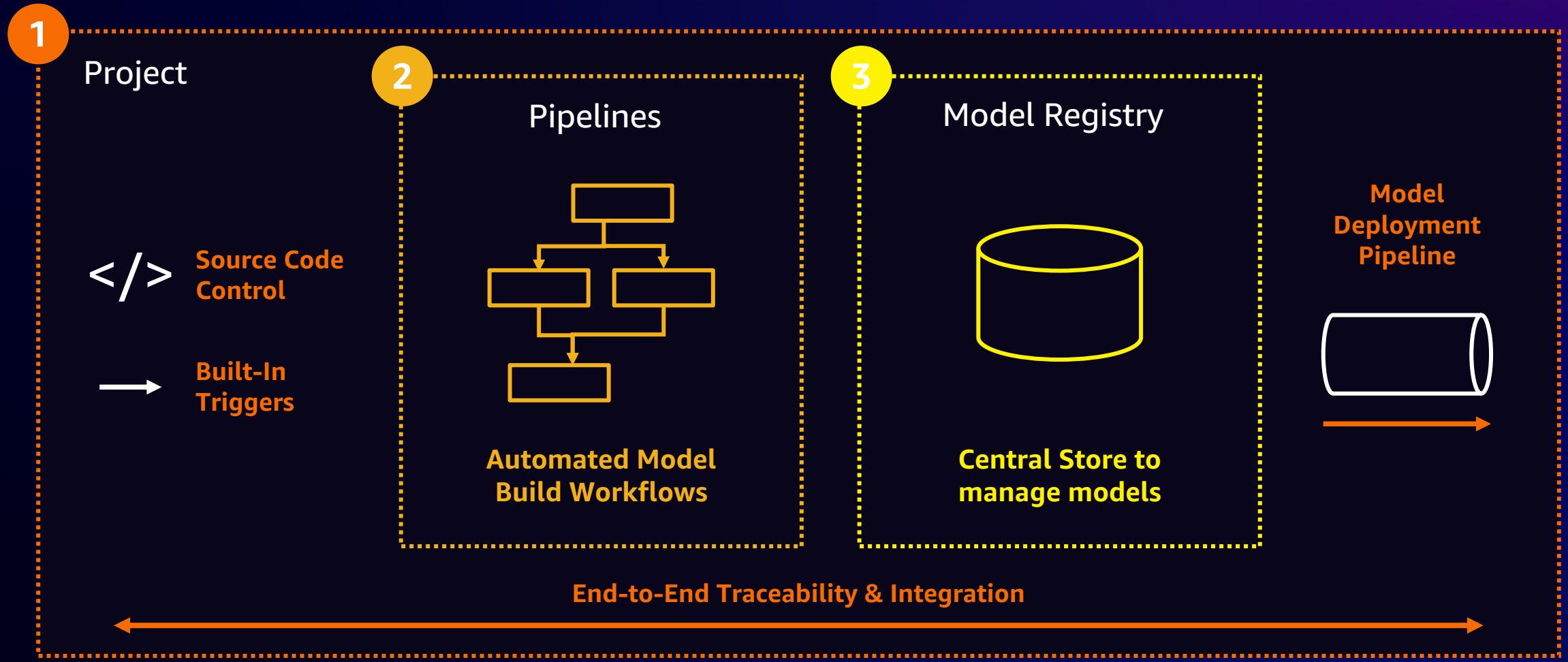
Comprehensive version control and traceability

Connect AI responses to source components for efficient troubleshooting and rapid issue resolution



Amazon Sagemaker MLOps

COMPONENTS



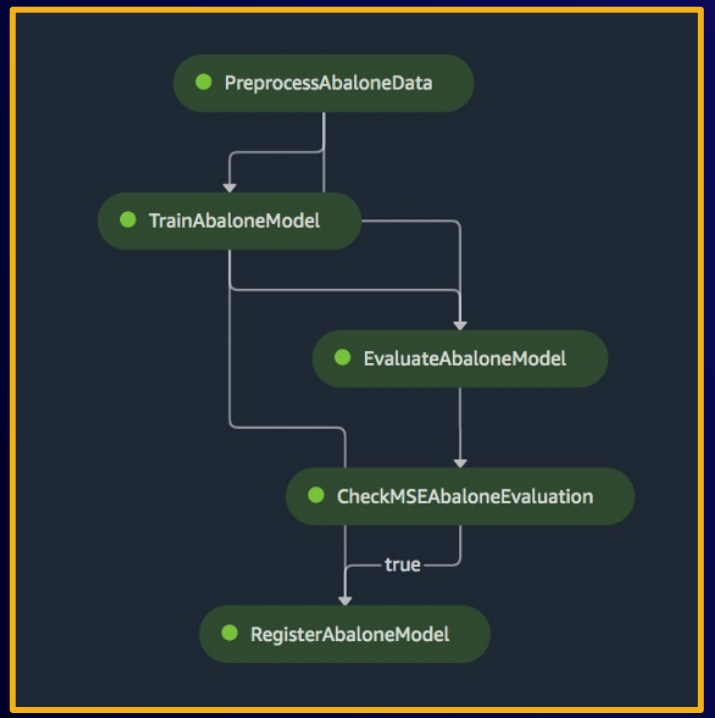
Creating Amazon Sagemaker AI Automated Pipelines

PIPELINE-AS-CODE

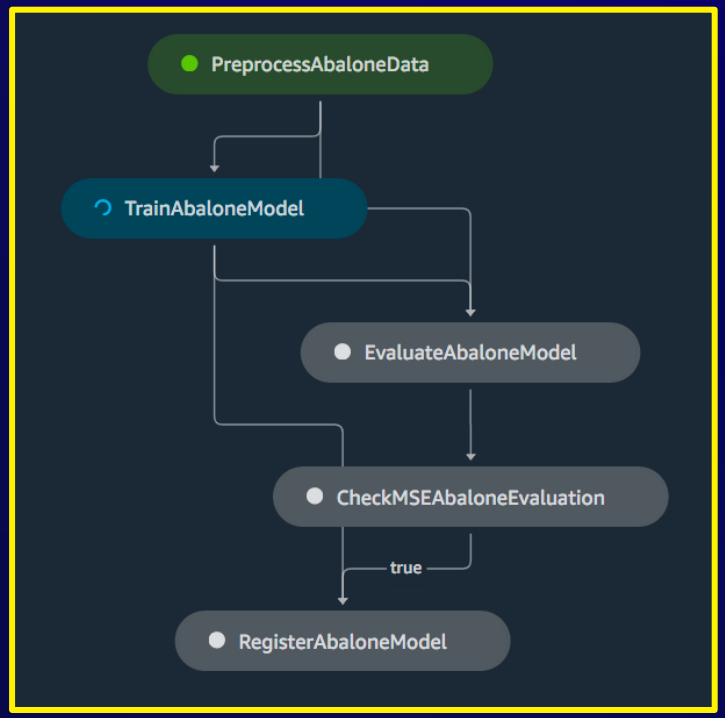
How it Works ...

1. Create Steps
Define & configure each step

2. Define Pipeline
Define & configure the workflow

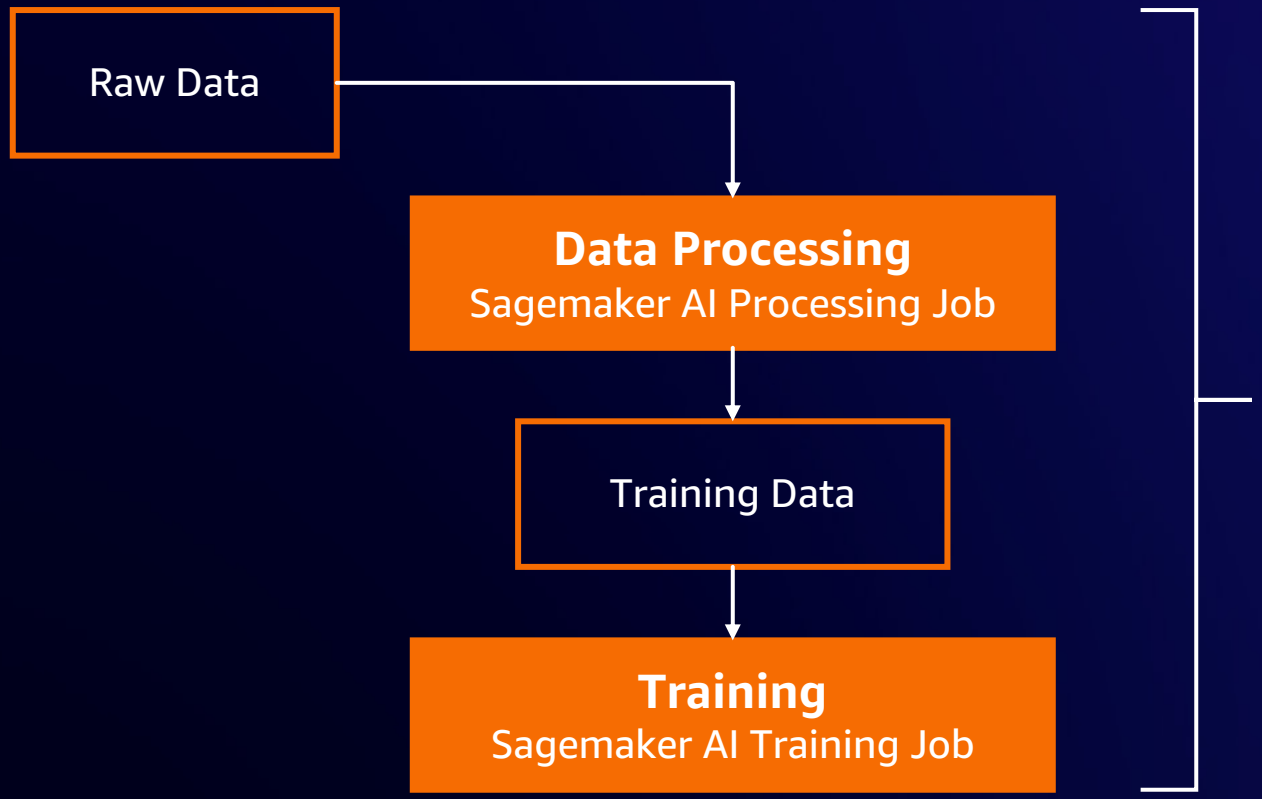


3. Start Pipeline
Execute the pipeline



Amazon Sagemaker AI Pipelines

Support for Step Caching



What if you want to:

- tweak hyperparameters?
- modify training code?

Amazon Sagemaker AI Pipelines

SUPPORT FOR PIPELINE PARAMETERS

1. Configure your parameter

```
from Sagemaker AI.workflow.parameters import (
    ParameterInteger,
    ParameterString,
    ParameterFloat
)

processing_instance_count = ParameterInteger(
    name="ProcessingInstanceCount",
    default_value=1
)
```

2. Pass in parameter on pipeline create

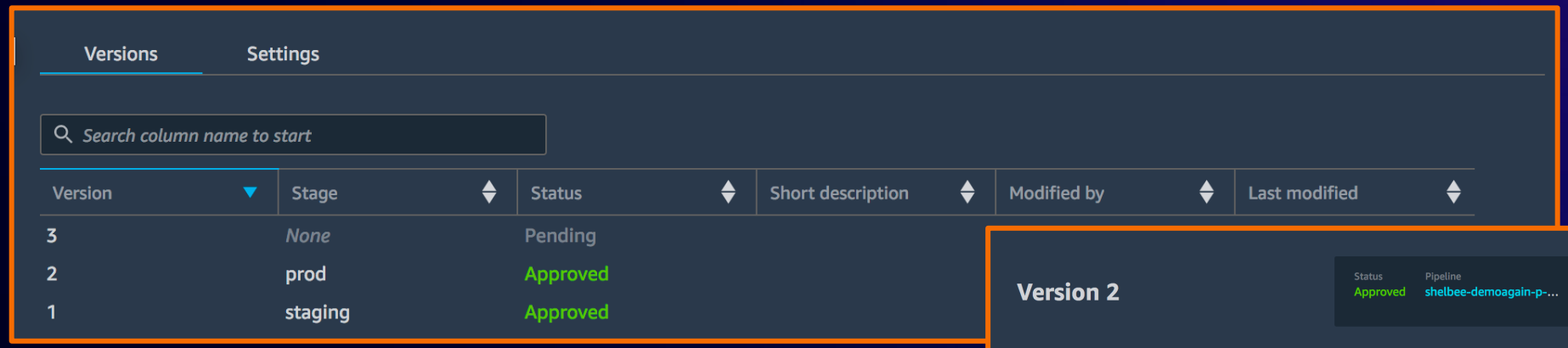
```
pipeline = Pipeline(
  name=pipeline_name,
  parameters=[
    processing_instance_count
  ],
steps=[step_process]
)
```

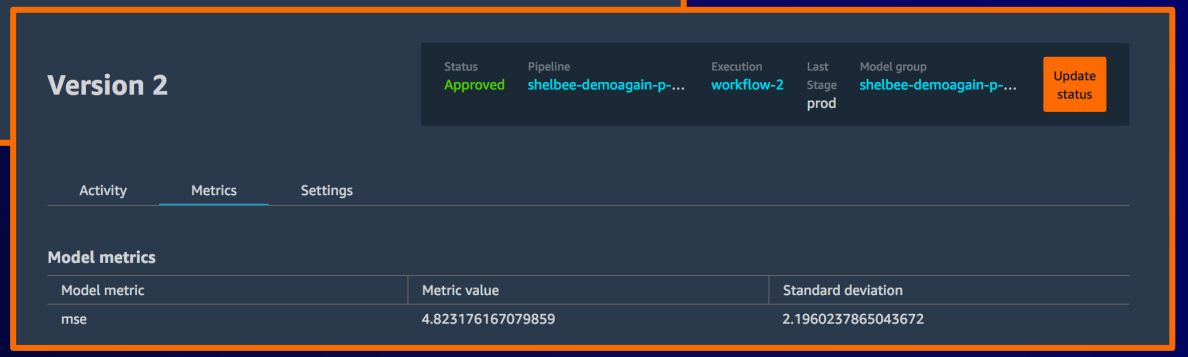
3. Optionally, pass a non-default value for pipeline execution

```
execution = pipeline.start(
   parameters=dict(
           ProcessingInstanceType="ml.c5.xlarge",
           ModelApprovalStatus="Approved”
   )
)
```

Amazon Sagemaker AI Pipelines

MODEL REGISTRY



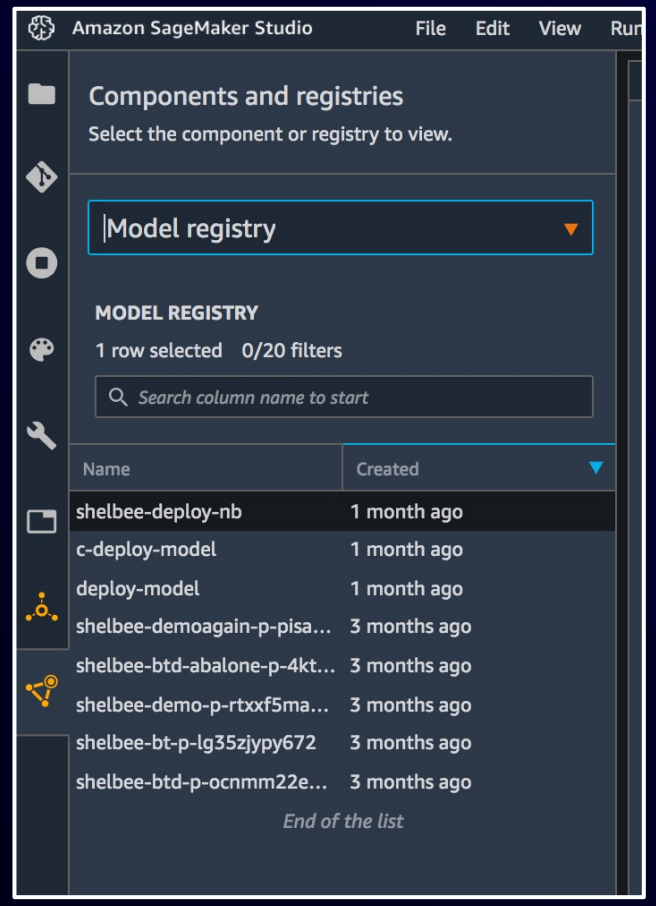


- Catalog models for production
- Manage model versions
- Associate metadata with a model
- Manage the approval status of a model
- Deploy models to production (with Projects)

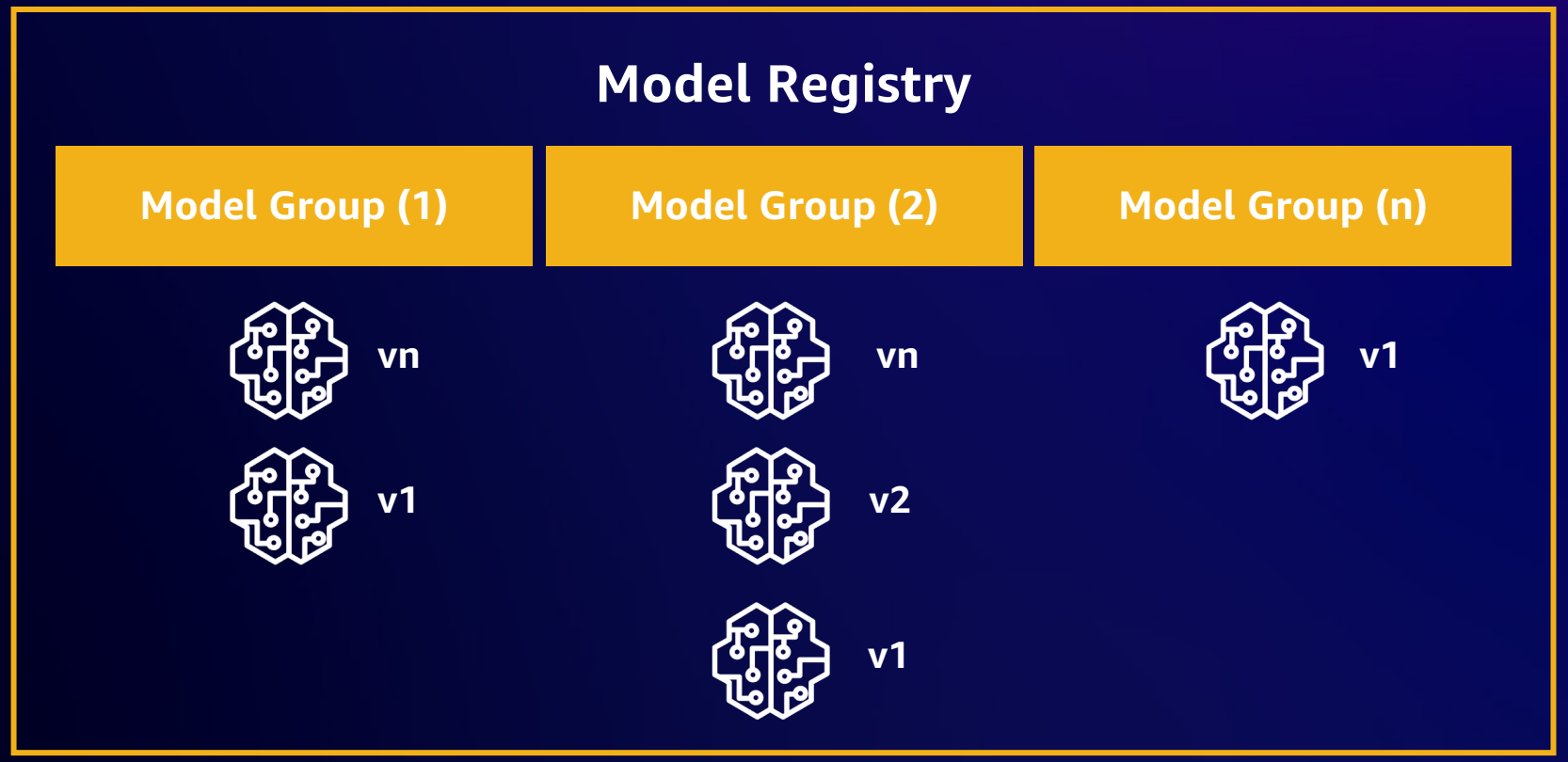
- Track model performance metrics
- Rollback model versions

Amazon Sagemaker AI Pipelines

MODEL REGISTRY



Hierarchy

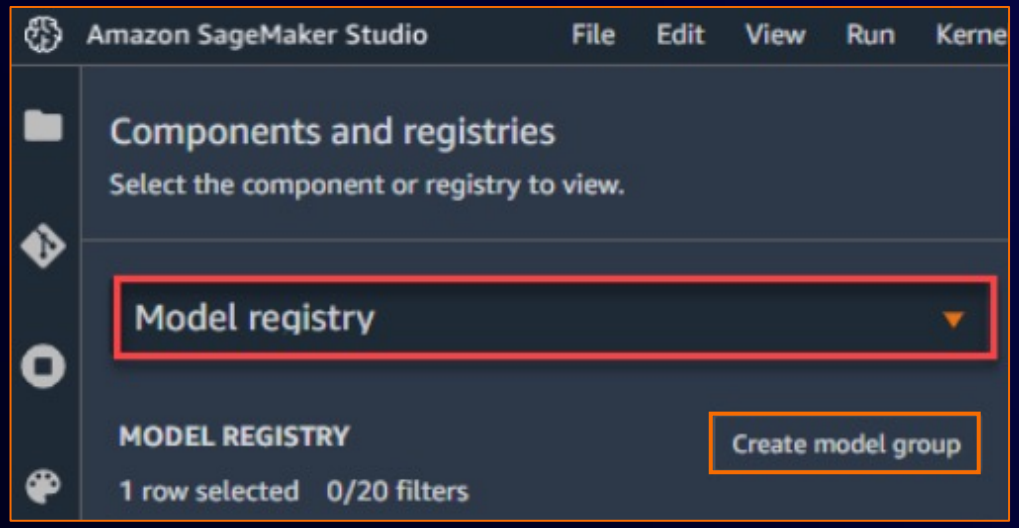


Amazon Sagemaker AI Pipelines

MODEL REGISTRY

How it Works ...

1. Create Model Group



2. Register a Model Version

Within a Pipeline, using RegisterModel Step:

```
step_register = RegisterModel(
    name="RegisterAbaloneModel",
    estimator=xgb_train,
    model_data=step_train.properties.ModelArtifacts.S3ModelArtifacts,
    content_types=["text/csv"],
    response_types=["text/csv"],
    inference_instances=["ml.t2.medium", "ml.m5.large"],
    transform_instances=["ml.m5.large"],
    model_package_group_name=model_package_group_name,
    approval_status=model_approval_status,
    model_metrics=model_metrics,
)
```

=

RegisterAbaloneModel

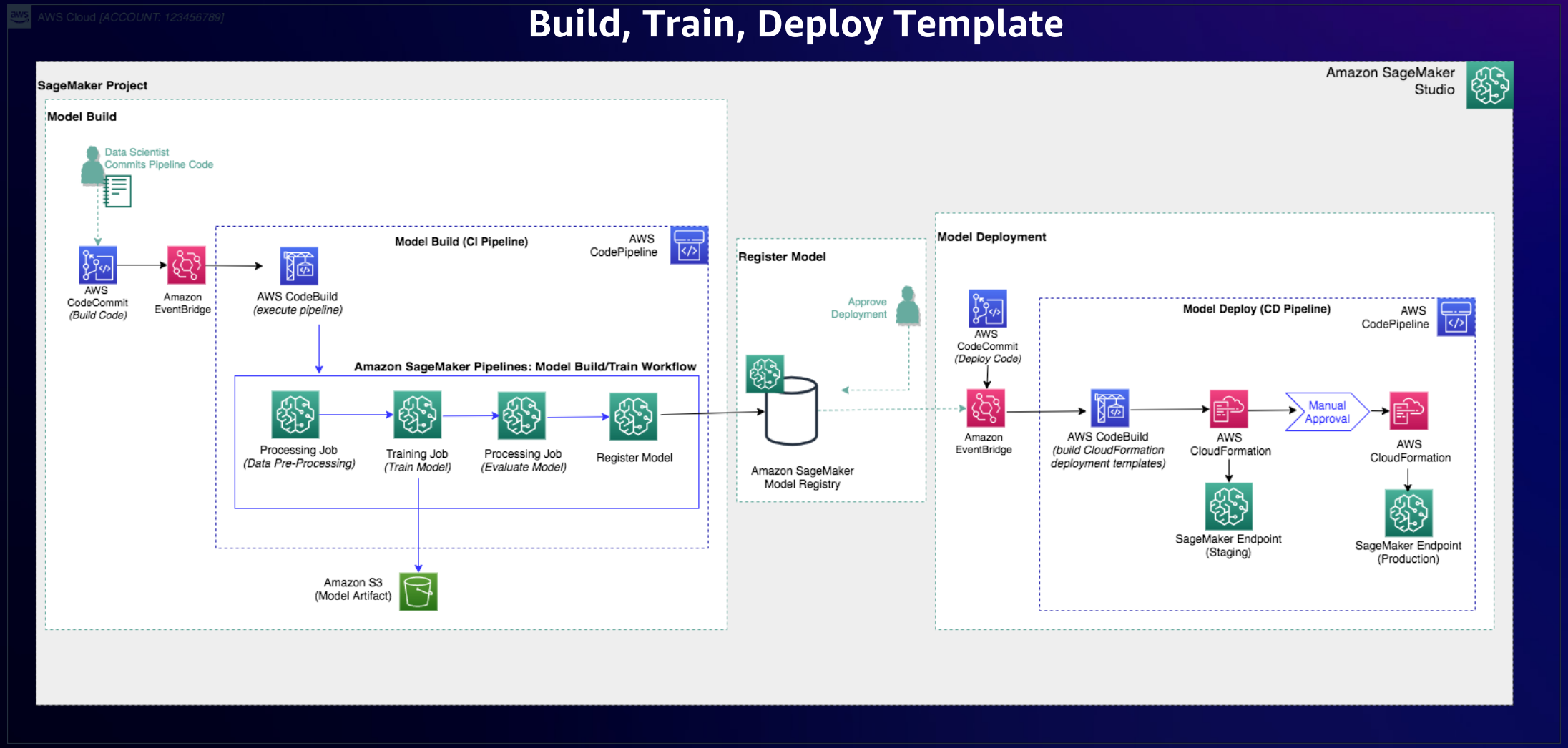
```
create_model_package_response =
sm_client.create_model_package(**create_model_package_input_dict)
model_package_arn = create_mode_package_response["ModelPackageArn"]
print('ModelPackage Version ARN : {}'.format(model_package_arn))
```

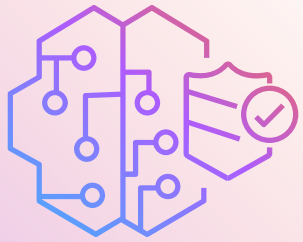
OR using boto3

Amazon Sagemaker AI Pipelines - Projects

HIGH LEVEL SERVICES VIEW

Build, Train, Deploy Template





Built-in Governance

Simplify access control and enhance transparency

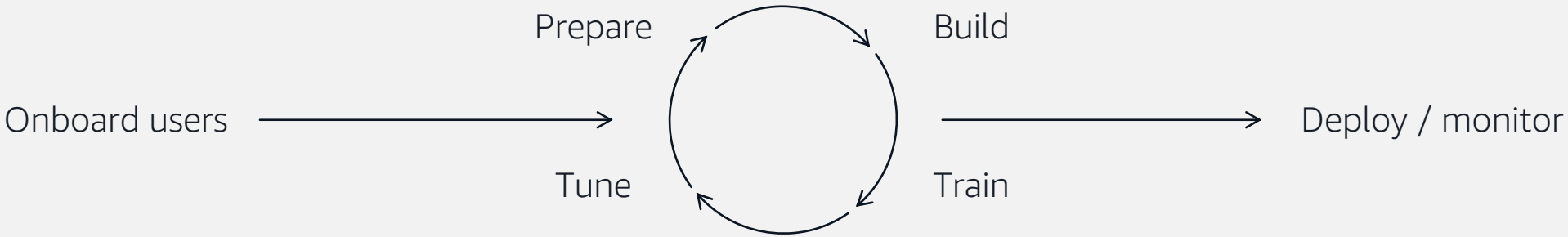
Key challenges for AI and ML governance

Purpose Built Tools for MLOps and governance

Hand-crafting custom policies is time consuming

Using disparate, manual tools to capture and share model information can be error-prone

Custom instrumentation to get visibility into model performance is expensive



Responsible use of AI and ML

Purpose Built Tools for MLOps and governance

Onboard

Setup ML users with custom permissions

Build

Perform bias analysis during exploratory data analysis

Document model information such as intended use and risk ratings

Train

Conduct bias and explainability analysis after training

Capture model training and evaluation observations

Deploy

Explain individual inferences from models in production

Monitor

Validate bias and relative feature importance over time

Audit performance and lineage of all your models, in one place

Governance

Purpose-built governance tools to help implement ML responsibly

Governance tools for AI and ML

Purpose Built Tools for MLOps and governance

Amazon SageMaker Role Manager

Define custom permissions for SageMaker users in minutes

Amazon SageMaker Model Cards

Create a single source of truth for model information

Amazon SageMaker Model Dashboard

Audit all your models, lineage and performance in one place

Comprehensive data protection and privacy

Purpose Built Tools for MLOps and governance

Your data used with Amazon SageMaker AI is not used for service improvement and not shared with third-party model providers

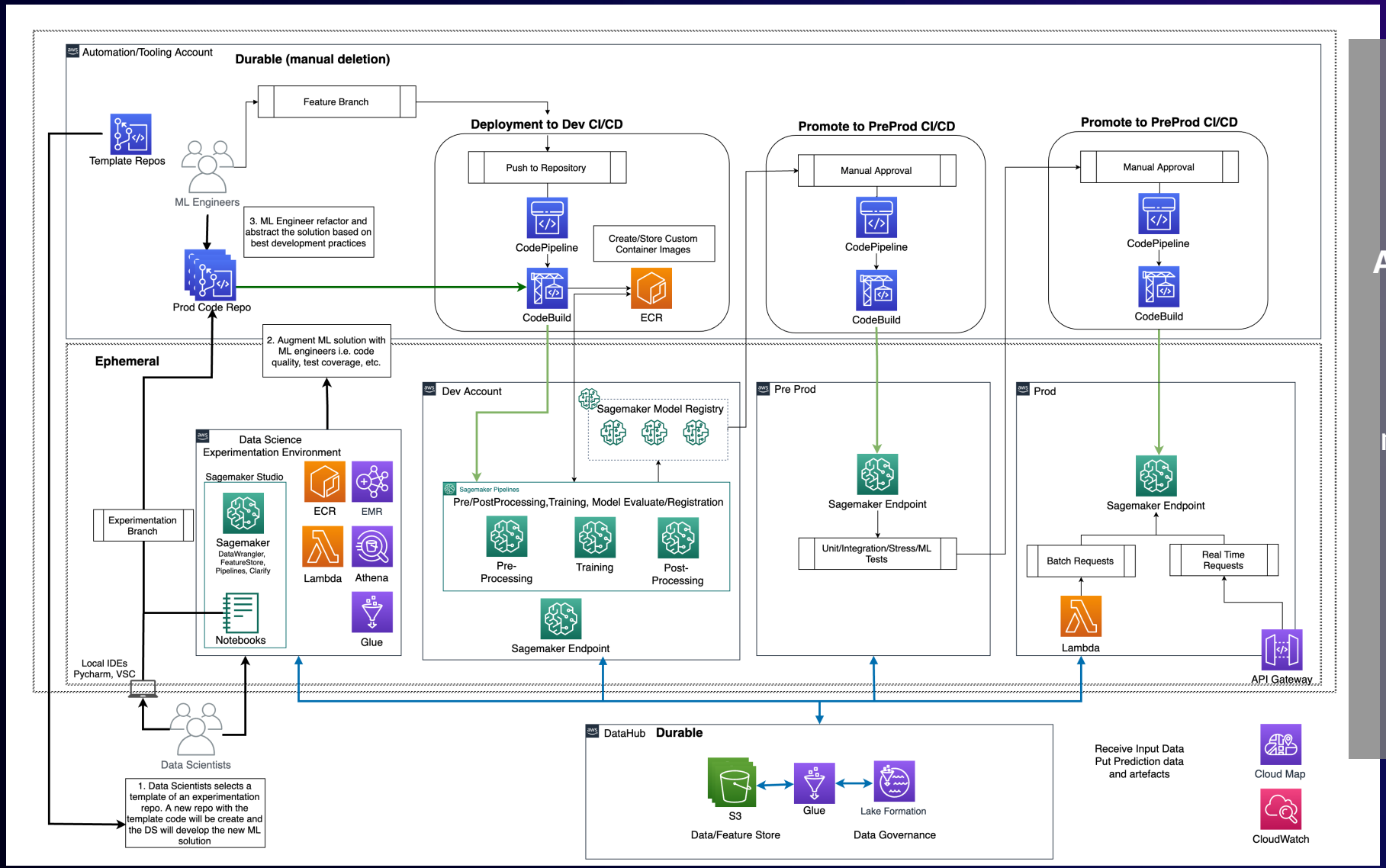
Private connectivity between Amazon SageMaker and your virtual private cloud (VPC)

Your data is encrypted in transit and at rest

Customize FMs privately, retaining control over how your data is used and encrypted

Used and encrypted Deploy FMs on single tenant endpoints provisioned for your inference use

MLOps at scale using SageMaker



SageMaker Reduces Overhead, Gray Areas, & Personas

Key Personas:
Consumers i.e. multiple Business Units

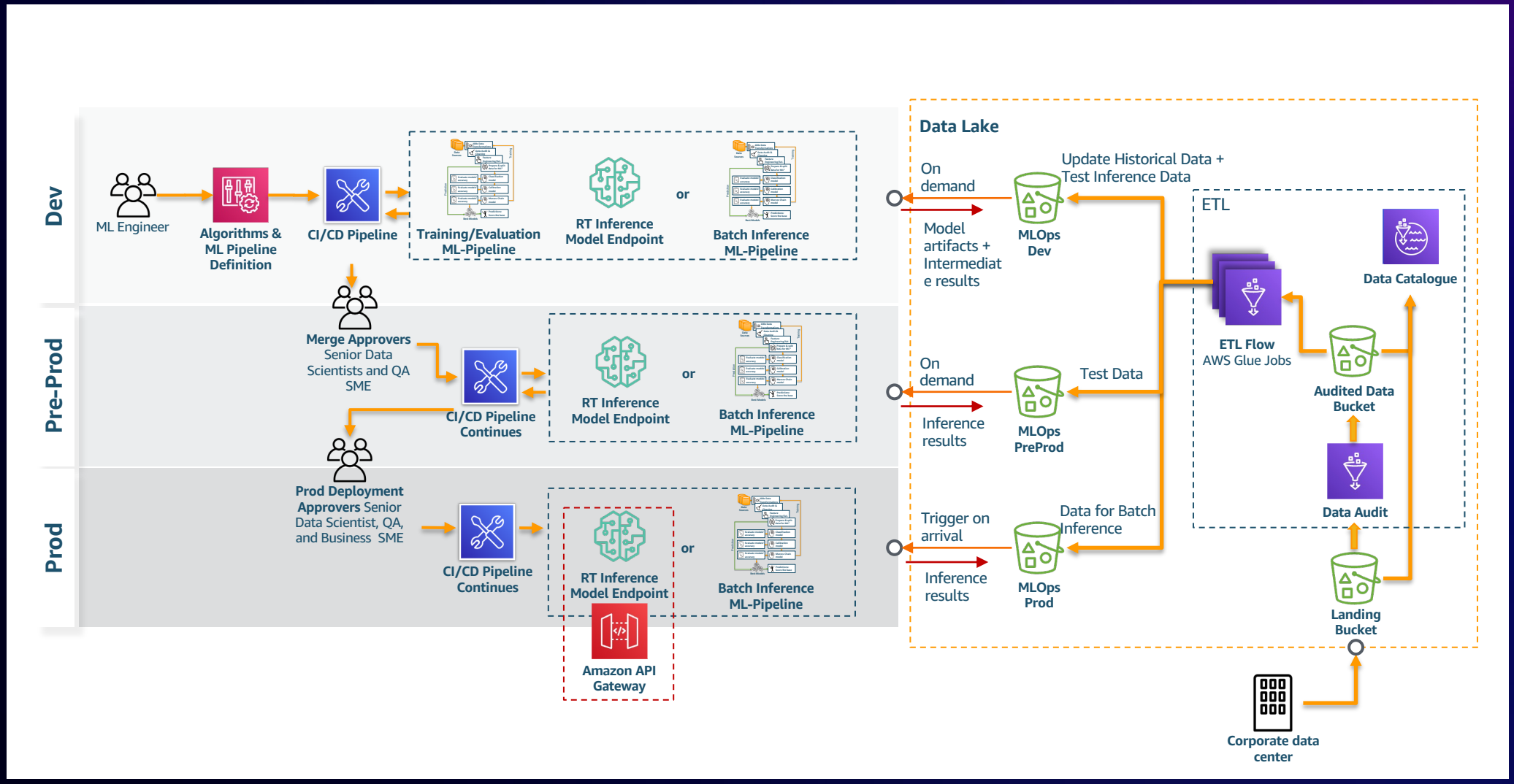
Business SMEs

Product Owners

Data Scientists

MLOps Triggering and Data Ingestion

Activate & Feed the ML Pipelines



Starting your MLOps journey

Documentation

AWS ML whitepapers and guides

https://aws.amazon.com/whitepapers

Classroom training

MLOps Engineering on AWS instructor-led training

https://aws.amazon.com/training/classroom/mlops-engineering-on-aws

MLOps tools

MLOps Workload Orchestrator

https://aws.amazon.com/solutions/implementations/mlops-workload-orchestrator

Continue your learning

AWS Skill Builder

AWS Skill Builder includes the following:

- Access more than **600 free** digital courses and learning plans.
- Explore resources with a variety of skill levels and more than **16** languages to meet your learning needs.
- Deepen your skills with digital learning on demand.

To train now, scan the code or see https://aws.amazon.com/training/digital

AWS Certifications

AWS Certifications include the following:

- Earn an industry-recognized credential.
- Receive foundational, associate, professional, and specialty certifications.
- Join the **AWS Certified Global Community** and get exclusive benefits.

To explore all AWS Certification exams, scan the code or see https://aws.amazon.com/certification/exams

Q&A

Thank you!