MLOps - NLP

Module Overview

MODULE OVERVIEW

- Learn about AWS SageMaker
- Demonstration of an end-to-end pipeline
  - Use case: Ticket classification
  - Introduction to AWS SageMaker
  - Demonstration of SageMaker Studio IDE
  - Build an end-to-end pipeline
- Assignment of an end-to-end pipeline
  - Use case: NER classification on healthcare data
  - Deliverables of the assignment
  - Submission of assignment

SESSION 1: INTRODUCTION TO SAGEMAKER

Recall MLOps principles

Understand different versions of MLOps

Understand different ways to implement MLOps

Get introduced to AWS SageMaker

Set up AWS account

Set up SageMaker

Demonstrate SageMaker Studio and its different services

Differentiate between the previous and the current development and production architectures

Principles of MLOps Projects

PRINCIPLES OF MLOPS PROJECTS

Code, Artifact and Experiment Tracking

Cross Team Collaboration

Reproducible Results

Development/ Production Symmetry

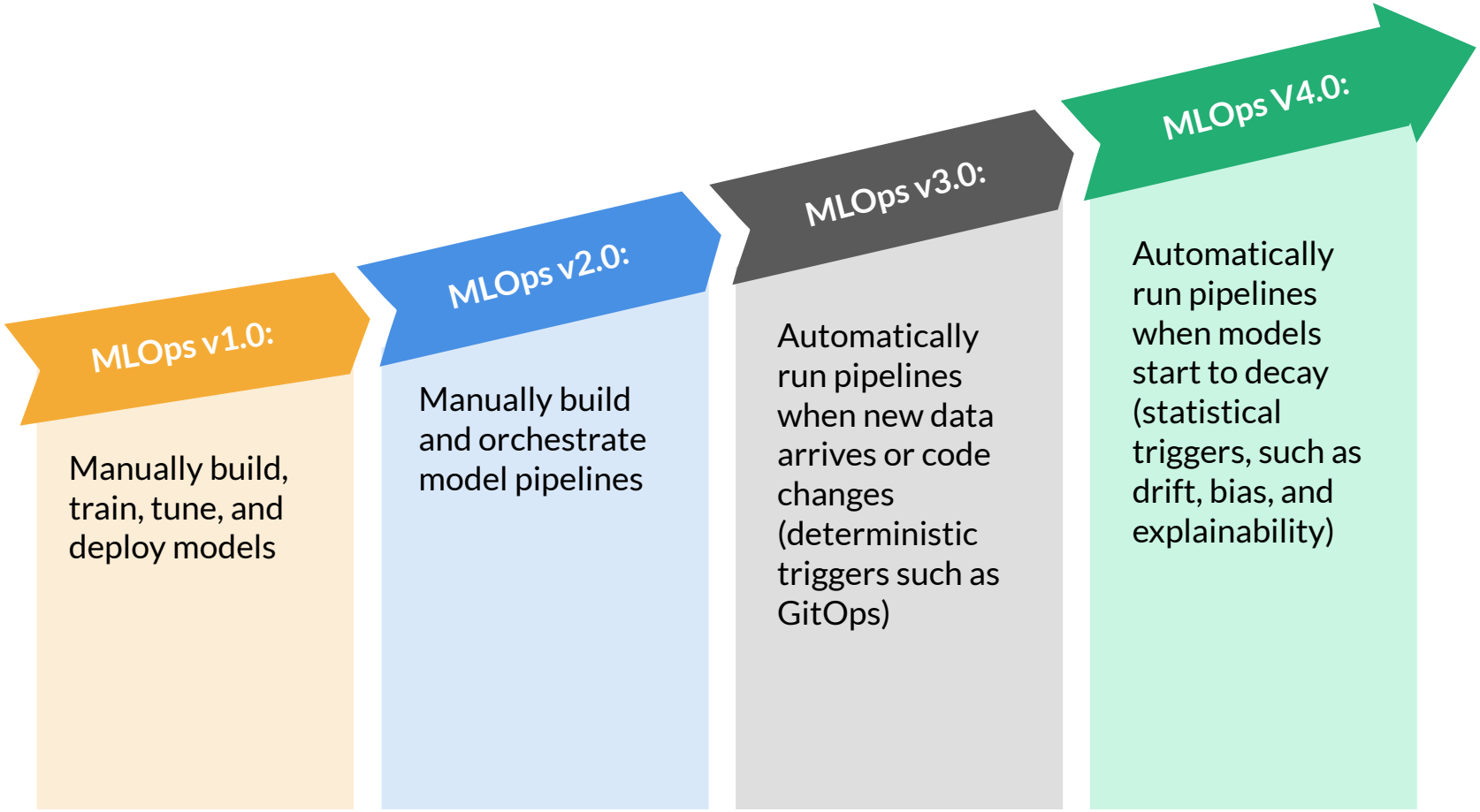
Continuous Integration

Continuous Deployment

Continuous Training

Module Health Check

PIPELINES AND MLOPS



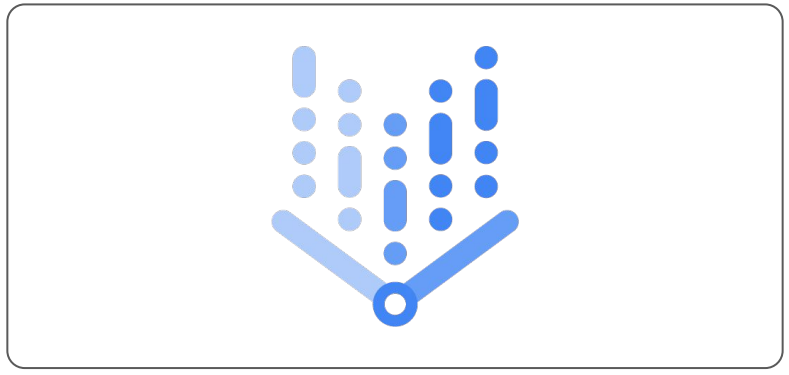
WAYS TO IMPLEMENT MLOPS

Two ways:

- Open source
- Managed services

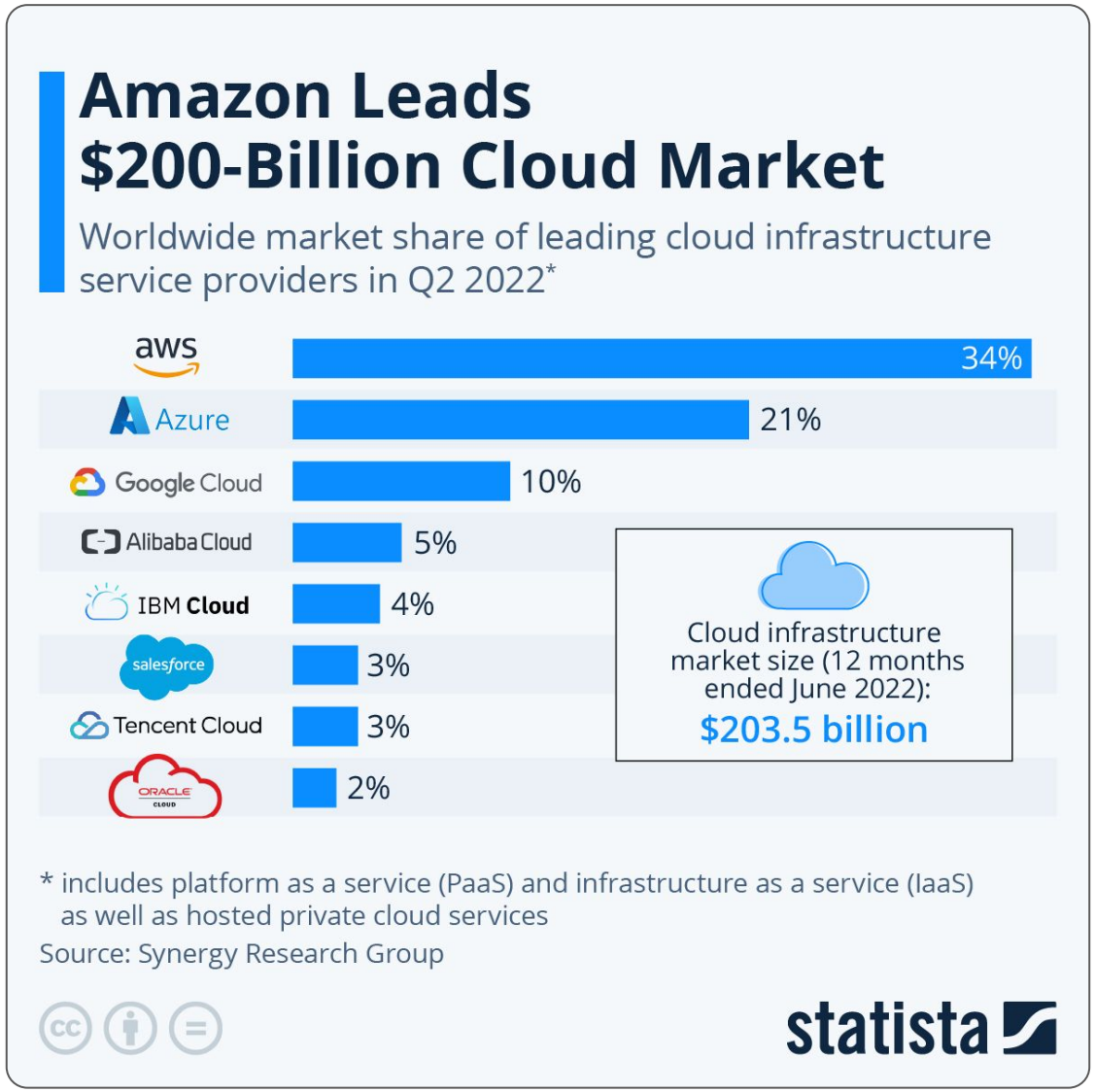
OPEN-SOURCE TOOLS

MANAGED SERVICES



Vertex AI

WHY SAGEMAKER?



COMPANIES USING SAGEMAKER

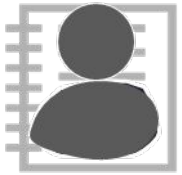
WHAT IS SAGEMAKER?

- Fully managed machine learning service by AWS
- Build and train machine learning models quickly
- Deploy them easily into production environment
- Provides Jupyter notebook instances
- Also provides common machine learning algorithms
- Bills only for the minutes that you use it to train and host

AMAZON SAGEMAKER

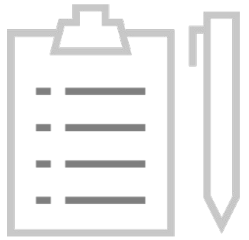
Amazon SageMaker Studio

First fully integrated development environment (IDE) for machine learning



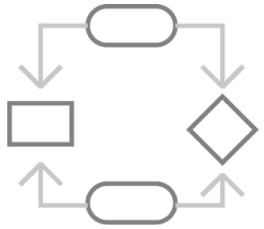
Amazon SageMaker Notebooks

Enhanced notebook experience with quick-start and easy collaboration



Amazon SageMaker Experiments

Experiment management system to organize, track and compare thousands of experiments

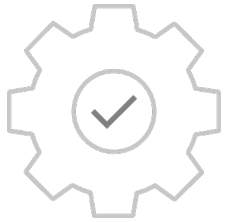


Amazon SageMaker Debugger

Automatic debugging analysis, and alerting

Amazon SageMaker Model Monitor

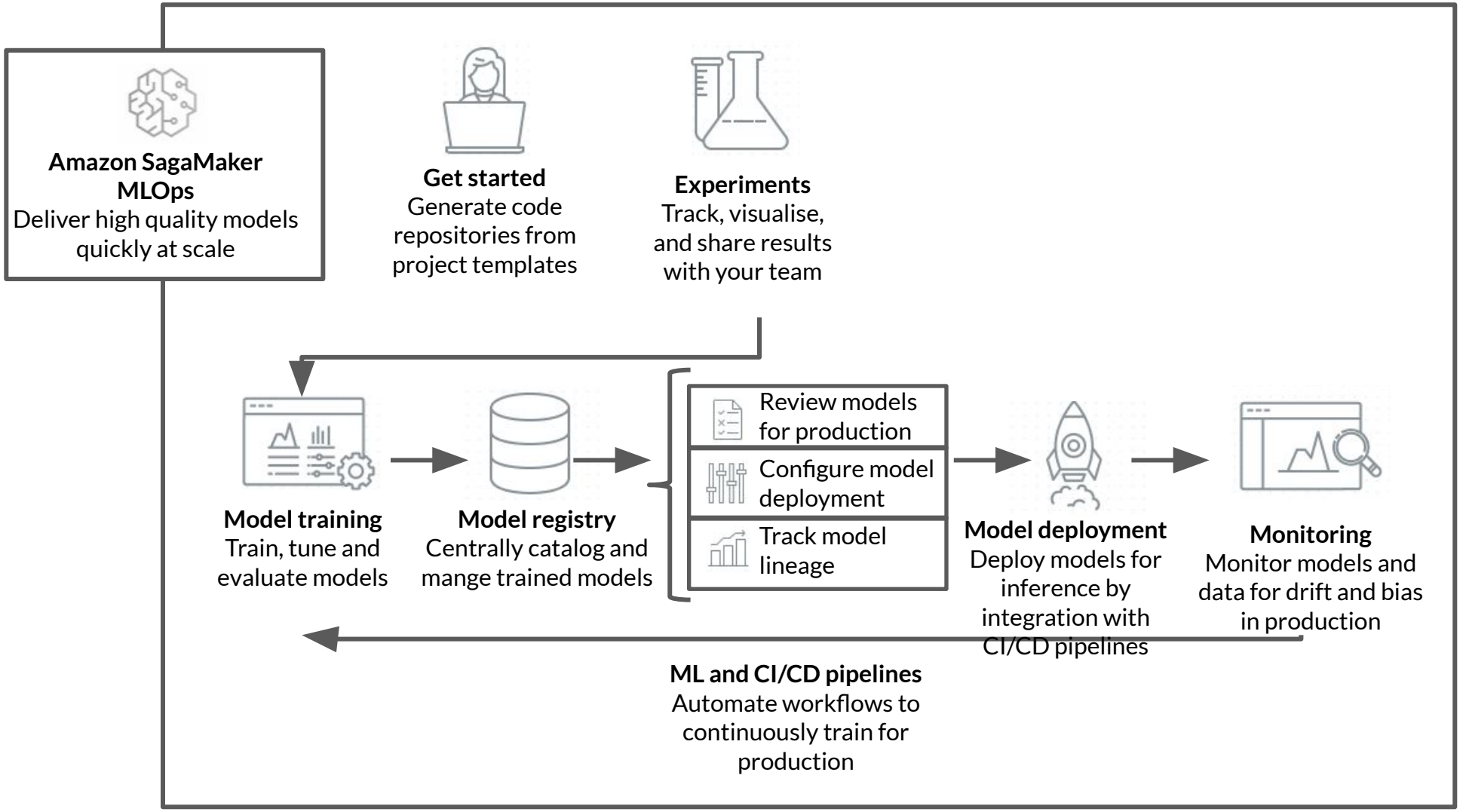
Model monitoring to detect deviation in quality and take corrective actions



Amazon SageMaker Autopilot

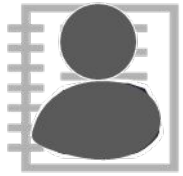
Automatic generation of machine learning models with full visibility and control

OVERVIEW


Amazon SagaMaker MLOps
Deliver high quality models quickly at scale
Get started
Generate code repositories from project templates
Experiments
Track, visualise, and share results with your team
Model training
Train, tune and evaluate models
Model registry
Centrally catalog and mange trained models
Review models for production
Configure model deployment
Track model lineage
Model deployment
Deploy models for inference by integration with CI/CD pipelines
Monitoring
Monitor models and data for drift and bias in production
ML and CI/CD pipelines
Automate workflows to continuously train for production

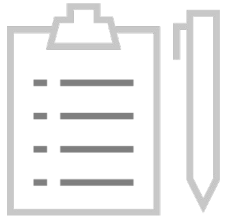
INTRODUCING AMAZON SAGEMAKER STUDIO

Integrated development environment (IDE) for machine learning



Collaboration at scale

Share notebooks without tracking code dependencies



Easy experiment management

Organise, track and compare thousands of experiments

Automatic model generation

Get accurate models with full visibility and control without writing code

Higher quality ML models

Automatically debug errors, monitor models and maintain a high quality

Increased productivity

Code, build, train, deploy and monitor in a unified visual interface

MLOPS PRACTICES AND BENEFITS

Code, Artifact and Experiment Tracking

Challenge: Bridging gap between model building and model deployment tasks

Practice: Lineage tracking and configuration management

Benefit: Repeatable process

Solution: Amazon SageMaker Experiments and Trials

Continuous Integration and Deployment

Challenge: Providing end-to-end traceability

Practice: Auditable ML pipeline

Benefit: Improve time to market

Solution: Amazon SageMaker projects and pipelines

Continuous Training and Model Monitoring

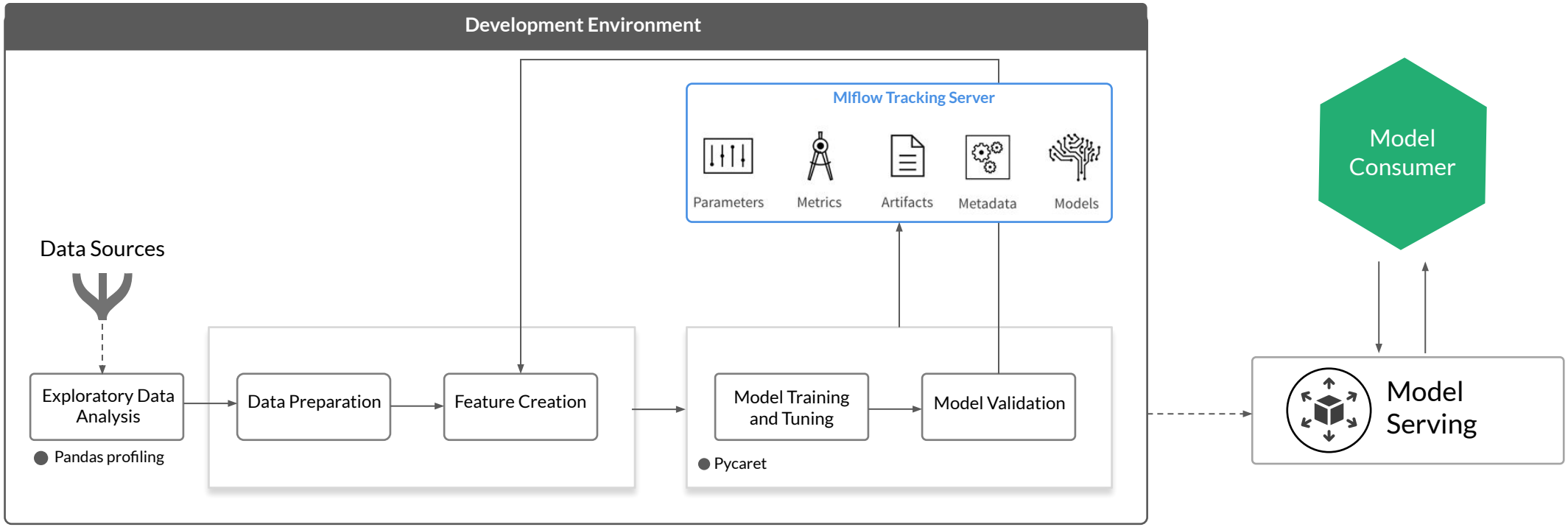
Challenge: Continuous delivery and monitoring

Practice: Maintain model performance over time

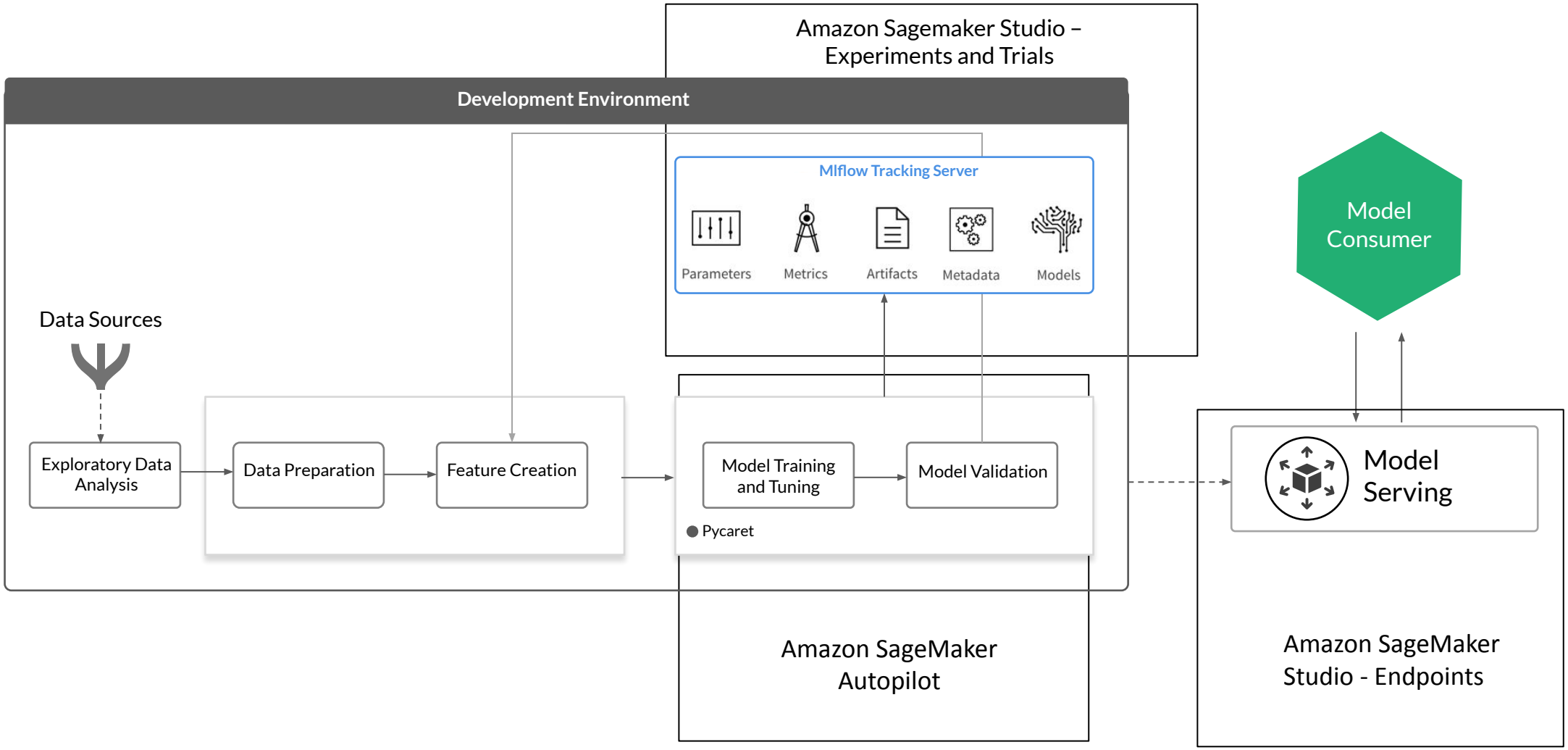
Benefit: Improve time to market

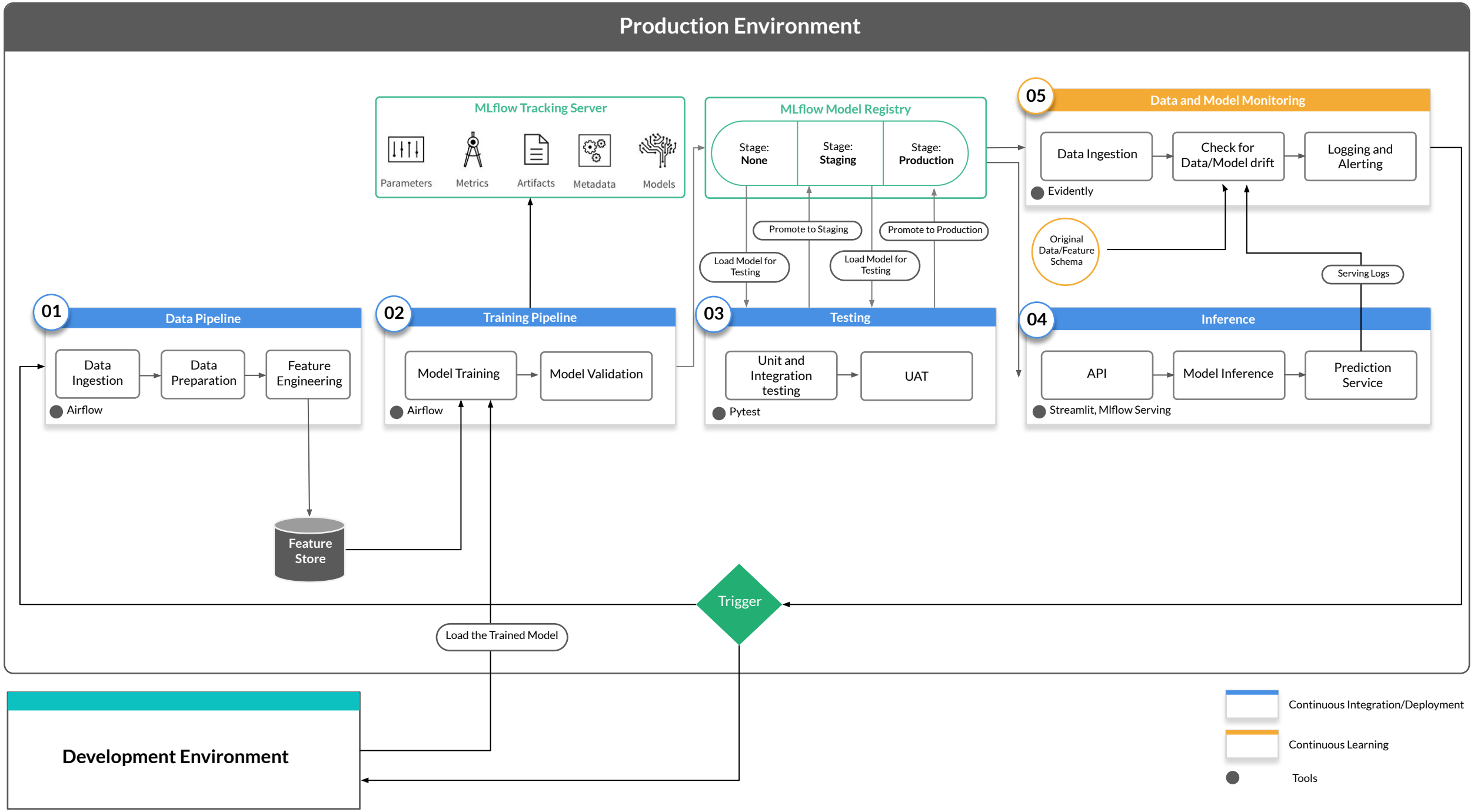
Solution: Amazon SageMaker model monitor, model registry

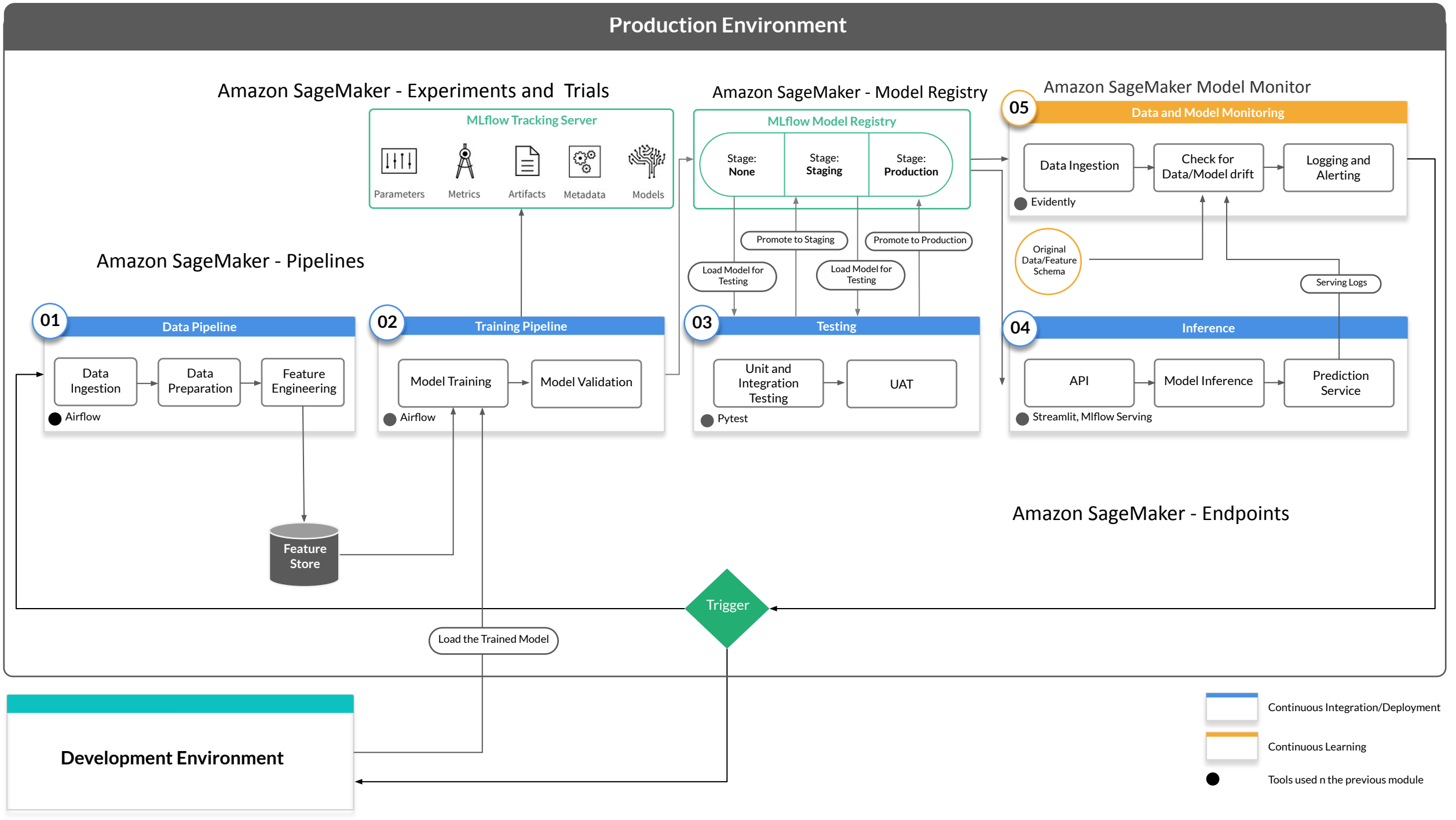
DEVELOPMENT ENVIRONMENT



DEVELOPMENT ENVIRONMENT IN SAGEMAKER




Production Environment
MLflow Tracking Server
Parameters
Metrics
Artifacts
Metadata
Models
MLflow Model Registry
Stage: None
Stage: Staging
Stage: Production
05
Data and Model Monitoring
Data Ingestion
Check for Data/Model drift
Logging and Alerting
Evidently
Original Data/Feature Schema
Promote to Staging
Promote to Production
Load Model for Testing
Load Model for Testing
Serving Logs
01
Data Pipeline
Data Ingestion
Data Preparation
Feature Engineering
Airflow
02
Training Pipeline
Model Training
Model Validation
Airflow
03
Testing
Unit and Integration testing
UAT
Pytest
04
Inference
API
Model Inference
Prediction Service
Streamlit, MlFlow Serving
Feature Store
Trigger
Load the Trained Model
Development Environment
Continuous Integration/Deployment
Continuous Learning
Tools


Production Environment
Amazon SageMaker - Experiments and Trials
MLflow Tracking Server
Parameters
Metrics
Artifacts
Metadata
Models
Amazon SageMaker - Model Registry
MLflow Model Registry
Stage: None
Stage: Staging
Stage: Production
Promote to Staging
Promote to Production
Load Model for Testing
Load Model for Testing
Amazon SageMaker Model Monitor
05
Data and Model Monitoring
Data Ingestion
Check for Data/Model drift
Logging and Alerting
Evidently
Original Data/Feature Schema
Serving Logs
Amazon SageMaker - Pipelines
01
Data Pipeline
Data Ingestion
Data Preparation
Feature Engineering
Airflow
02
Training Pipeline
Model Training
Model Validation
Airflow
03
Testing
Unit and Integration Testing
UAT
Pytest
04
Inference
API
Model Inference
Prediction Service
Streamlit, MlFlow Serving
Amazon SageMaker - Endpoints
Feature Store
Trigger
Load the Trained Model
Development Environment
Continuous Integration/Deployment
Continuous Learning
Tools used n the previous module

CLOUD SERVICES VS OPEN SOURCE

Services	Native Cloud-based approach	Open source tools integration
End2End MLOps	Integrated	Plug and play
Time to set up	Less	High
Maintenance of infrastructure	Low	High
Ease of deployment	High	Medium

CLOUD SERVICES VS OPEN SOURCE

Services	Native Cloud-based approach	Open source tools integration
Learning curve	Low	High
IDE studio support	In built	Need to be configured
Endpoint deployment	Integrated via SDK	Need to be configured
Pre configured MLOps template	Available	Not available
Companies leveraging	Cloud first companies, which have majority of infrastructure on Cloud	Companies that have on premises infrastructure

BUILD VS BUY

- Outsource all your ML use cases to external vendors
  - Vendors manage all infrastructure
  - Only infrastructure needed is to move the data to your vendor
  - Move predictions back from vendor to end-users
- Build and maintain everything in-house
  - In case of sensitive data
  - Need to do everything in-house
- Companies are generally not at either of these extremes

BUILD VS BUY

Factors affecting the build vs buy decision:

- The current stage of your company
- Competitive advantages of your company
- Maturity of the available tools

Session Summary

- MLOps principles and different maturity levels of MLOps project
- Different ways to implement MLOps
- Sagemaker – Introduction and set up
- Demonstration of different services of SageMaker Studio
- Difference between the current and the previous development and production environment