DATA+AI SUMMIT 2022

Survey of Production ML Tech Stacks

Requirements for an ML platform

Conor Murphy & Mary Grace Moesta
Databricks

ORGANIZED BY databricks

Your Tenacious Duo

Conor Murphy

Lead Data Scientist + Manager @ Databricks

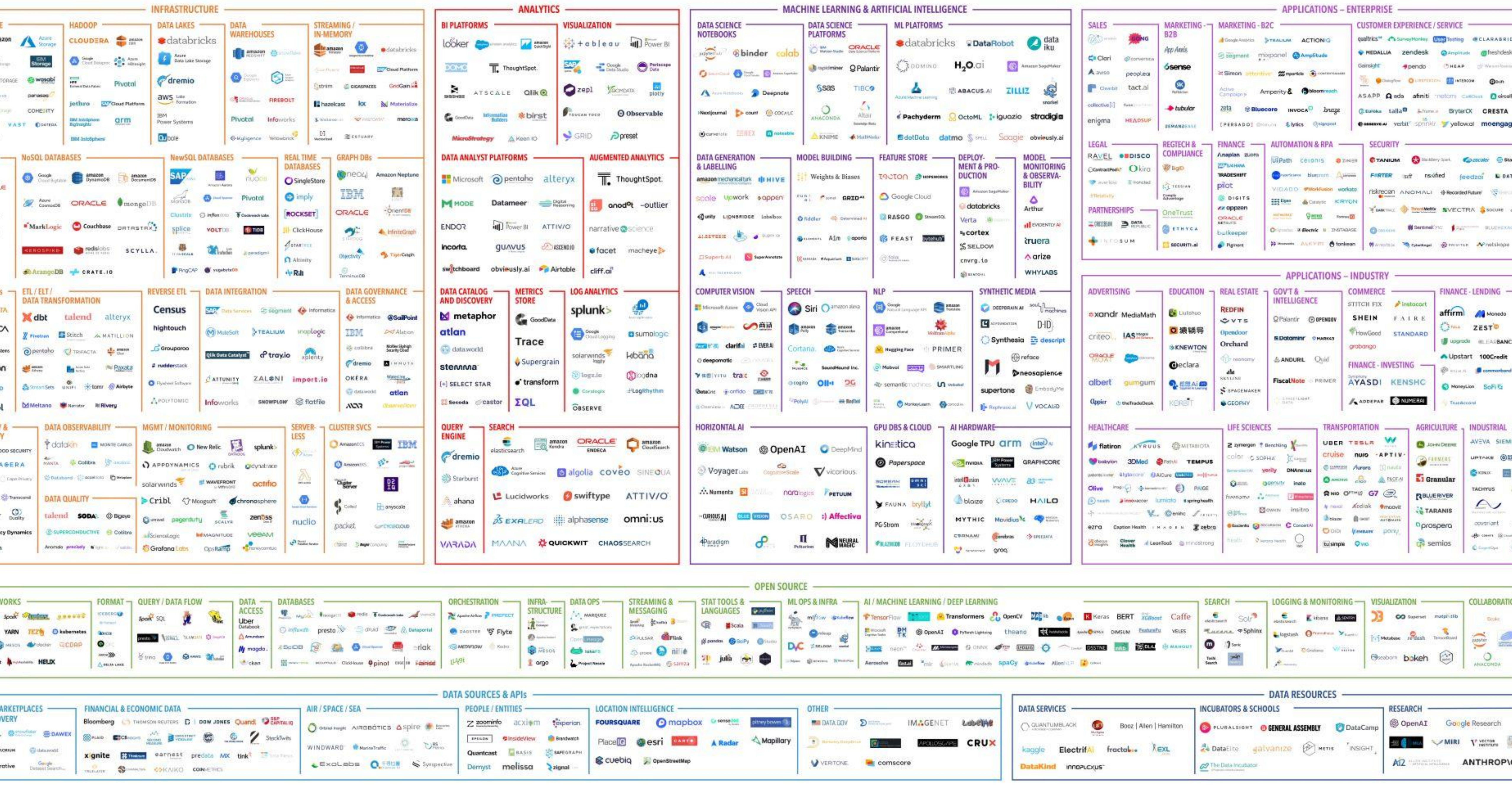
5+ years in distributed ML and production systems

Mary Grace Moesta

Data Scientist @ Databricks

3+ years in distributed ML and production systems

MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, AND DATA (MAD) LANDSCAPE 2021



The Problem

Standardize tech stacks around best practices

- ML platform technology stacks have high build costs
- There are many tools at different levels of maturity and maintenance
- Few end-to-end standards

The Solution

- Standardize tech stack around best practices
- Leverage industry talent by using the most current technologies
- Better enable data teams throughout the stack

Agenda

What to expect

- Introduction
- Organizing data teams
- Features of ML platforms
- Overview of ML tech stacks
  - Language choices
  - Collaboration
  - Python libraries
  - CI / CD
  - ML workflows
  - Deployment

Organizing ML Teams

What Doesn't Work

1. Data science is managed under IT
2. Data scientists manage production models…and then can't develop new models
3. An "MLE" team is created but struggles with handoffs
4. Data pipelining teams struggle to update pipelines using the data warehousing playbook
5. Local development doesn't translate to production systems

Where to put the Data and ML Engineer(s)

- Embedded approach: embedded MLE on each team (or embedded DS on various product teams)
- Centralized MLE approach: separate MLE team that refactors DS code
- Centralized DE approach: monolithic repo for data engineering, looser standards on DS teams

Solution: hand-off checklists with clearly enforced standards

Features of an ML Platform

Defining core components

Core Tech Stack

- Language
- Collaboration
  - Source control
  - Notebooks
  - IDE
  - BI Tools
- Libraries
- Cloud
- ETL Processes

Data + Modeling

- Feature store
- Experiment tracking
- Model registry
- Governance
  - Reproducibility
  - Auditing
- Administration
  - Cost
  - Users
- Security

ML Workflow

- CI/CD
- Orchestration
- Testing
- Retraining Schedules

Deployment

- Modalities
  - Batch
  - Real time
  - Streaming
  - Mobile
- Monitoring
  - Drift
  - Logging
  - Alerting
- A/B Testing

An Opinionated Approach

Downloads last day: 397,503
Downloads last week: 2,599,806
Downloads last month: 11,078,112

- Python (production, maturity, ecosystem)
- Open source
- Focus on traction and unified analytics, not an exhaustive list of newer players

MLflow Components

mlflow™ **Tracking**

Record and query experiments: code, data, config, results

mlflow™ **Projects**

Packaging format for reproducible runs on any platform

mlflow™ **Models**

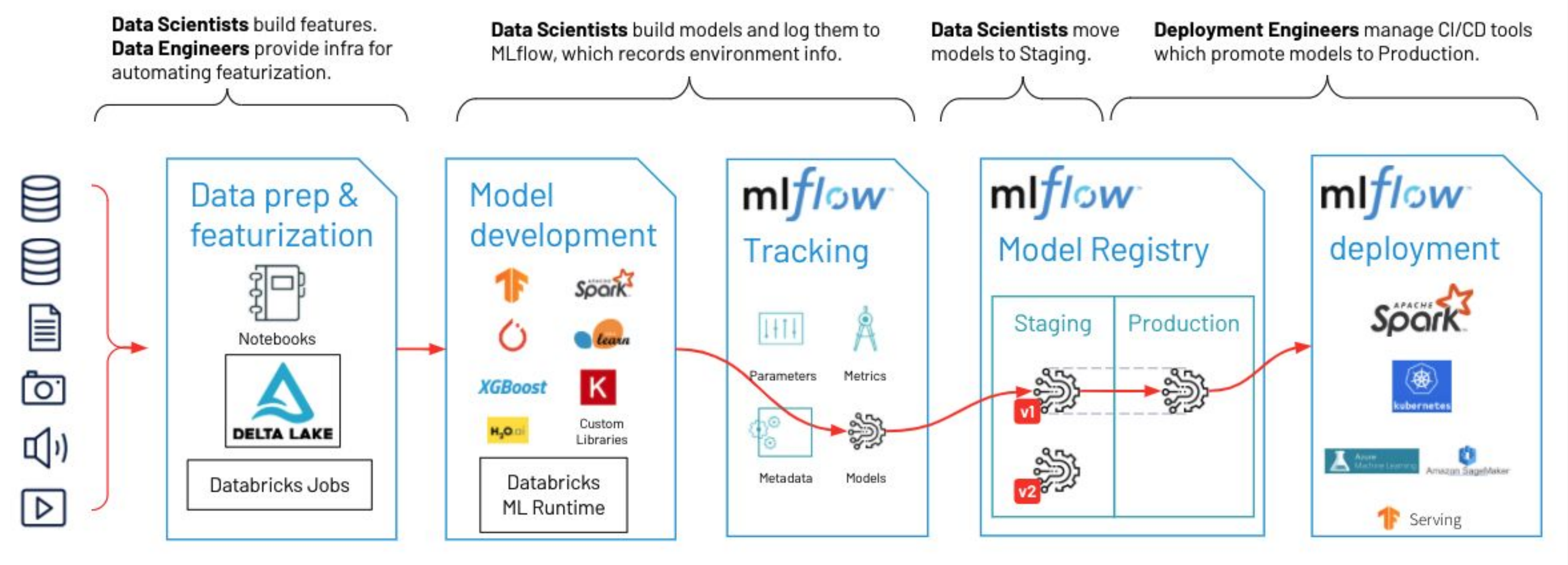
General model format that supports diverse deployment tools

mlflow™ **Model Registry**

Centralized and collaborative model lifecycle management

- APIs: CLI, Python, R, Java, REST

The Full ML Lifecycle



ML Workflow

Tools to support end to end ML workflows

	Open Source	Adoption	Production Tools	Strengths	Limitation	Downloads
MLflow	Yes	High	Yes	Compatibility, multi-cloud	High overhead for OS management	~`10.1M
Weights and Biases	Limited functionality	Medium	Yes	Visualization and hyperparameter tuning	Limited feature set open sourced	~ 2.8M
Neptune	Limited functionality	Medium	Yes	Metadata storage	Limited feature set for OSS	~ 567K
Tensorboard	Yes	Medium	Limited	DL training visualization	Limited model registry	~14.4M
Azure ML	No	Medium	Yes	Azure ecosystem	Proprietary, cloud specific	
Sagemaker	No	Medium	Yes	AWS ecosystem	Proprietary, cloud specific	
Vertex Ai	No	Low	Yes	GCP ecosystem	Proprietary, cloud specific	

Language Choice

	Open Source	Adoption	Production Tools	Industry	Strengths	Limitation
Python	Yes	High	Yes	General	Spark	Limited Statistics, no type safety
R	Yes	Medium	Medium	Academia + Biotech	Statistics	Limited Spark, production, OOP
SQL	Mixed	Medium	Yes	General	Well Known	No ML
Scala	Yes	Medium	Yes	Engineering focus	Data Engineering	Poor ML
Excel	No	Medium	No	General	Interactive	Production + automation
Matlab	No	Low	No	Academic + engineering	Academic standard	Limited production
SAS	No	Low	No	Academic + financial Services	Academic + pipelining	Expensive, proprietary
SPSS	No	Low	No	Academic	Academic standard	UI-based, Limited production

Collaboration

François Chollet @fchollet · 24m

The thing is, applied ML engineers have opposite needs to those of researchers. When you do applied ML, you need a framework that's feature-complete, reasonably prescriptive, high-level, that guides you towards industry best practices. And ofc you want it to be production-ready.

TensorFlow

Downloads last day: 509,503

Downloads last week: 3,724,870

Downloads last month: 17,734,961

PyTorch

Downloads last day: 290,123

Downloads last week: 1,826,962

Downloads last month: 9,017,579

Python Libraries

Python frameworks for ML

	Open Source	Distributed	PyPi Downloads (monthly)	Strengths	Limitation
sklearn	Yes	No	~ 32.8 million	Single node industry standard	Limited by data size
XGBoost	Yes	Yes	~ 7.7 million	Accuracy, speed, distributed, tunable	"Boosters" can be clunky
LightGBM	Yes	Yes	~ 7.2 million	Accuracy, speed, GOSS	Hard to troubleshoot
SparkML	Yes	Yes	N/A	Good for large data	Only a subset of algorithms
Tensorflow	Yes	Partially	~ 14.2 million	Deep learning + production	Distributed can be challenging
Pytorch	Yes	Partially	~ 7.9 million	Deep learning + publications	Poor production tools
Horovod	Yes	Yes	~ 54K	Distribution with Spark	Poor market penetration
Ray	Yes	Yes	~ 1 million	Generalized distribution	Not a step function improvement
Petastorm	Yes	Yes	~92K	Data format for distributed DL	Can be a strange API

CI / CD

Frameworks for orchestration, testing, alerting, and monitoring

	Open Source	Databricks	AWS	Azure	Third Party
Orchestration	Airflow, Jenkins, Terraform	[Databricks Workflows](), Jobs	CodePipeline, Codebuild, CodeDeploy	DevOps, Data Factory	
Git Hooks / Web Hooks		MLflow webhooks	CodeCommit	DevOps	Github Actions, Gitlab, Travis CI
Testing	pytest		Developer Tools	Azure Test Plans	Sonar
Monitoring	Open Telemetry, OpenLineage			Log Analytics	Data Dog, Splunk
Alerting		Jobs	Cloudwatch	Monitor, Teams integrations	PagerDuty, Slack integrations
Artifact Management	Maven, PyPi, Artifactory, TensorHub,		CodeArtifact	Azure Artifacts	Nexus
Environment Management	Conda, Docker, Kubernetes,	MLflow projects	Elastic Container Registry	Container Registry	Docker Hub

Deployment

Toolkit for real time deployment

- For real time deployment there are many options, the most popular being
  - Kserve
  - Cloud-based, real time serving
    - Databricks Model Serving
    - AWS Sagemaker
    - Azure Kubernetes Service
    - Google Vertex.ai

Trends

Continuation of OSS, cloud, data, AI

Multi-cloud (k8, Databricks)

AutoML

CI/CD

> **"By 2021, over 75% of midsize and large organizations will have adopted a multicloud and/or hybrid IT strategy."**
>
> Gartner Predicts

Wrapping up

- Think critically about team member dedicated to production ML
- Choose “unified” tech stacks over single tools
- MLflow can manage many features of an ML stack–and has some exciting announcements to come!

Abstract

Production machine learning demands stitching together many tools ranging from open source standards to cloud-specific and third party solutions. This session surveys the current ML deployment technology landscape to contextualize which tools solve for which features of production ML systems such as CI/CD, REST endpoints, and monitoring. It'll help answer the questions: what tools are out there? Where do I start with the MLops tech stack for my application? What are the pros and cons of open source versus managed solutions? This talk takes a features-driven approach to tool selection for MLops stacks to provide best practices in the most rapidly evolving field of data science.

CI/CD and Other Deployment Considerations

Packaging up a subset of surveyed tools for deployment

Code & Libraries

Data and Artifact Management

Orchestration

Deployment

Requirements

Tool Selection

Let's Look at an Example: Core Tech Stack

Say we want to build an Open Source centric stack

Requirement: Our system involves training an image classifier in a development environment, promoting the training code to a production environment, retraining the model in production environment, deploying as a REST endpoint, and daily monitoring of model performance

Core Tech Stack

→

Because we need an **OS deep learning library** with **transfer learning**

→

- **Python** and **tensoflow** for pre-processing, data augmentation, and modeling
- **Local IDE** for easy sharing of code across the team

Let’s Look at an Example: Core Tech Stack

Say we want to build an Open Source centric stack

Requirement: Our system involves training an image classifier in a development environment, promoting the training code to a production environment, retraining the model in production environment, deploying as a REST endpoint, and daily monitoring of model performance

Language

**Collabo
ration**

Libraries

Cloud

Data Storage

Let’s Look at an Example: Data

Say we want to build an Open Source centric stack

Requirement: Our system involves training an image classifier in a development environment, promoting the training code to a production environment, retraining the model in production environment, deploying as a REST endpoint, and daily monitoring of model performance

Data

→

Because we need an **OS tools** to **consistently manage** artifacts (in this case the code) across **dev and prod** environments

→

- **MLflow** to manage the lifecycle of the code and model through dev and prod
- Cheap **cloud storage** with **Delta** to manage feature engineered tables

Let's Look at an Example: ML Workflow

Say we want to build an Open Source centric stack

Requirement: Our system involves training an image classifier in a development environment, promoting the training code to a production environment, retraining the model in production environment, deploying as a REST endpoint, and daily monitoring of model performance

ML Workflow

Because we need an **OS tools** to **manage our code**, automatically trigged based on actions, and

- **MLflow webhooks** to automatically trigger code based on model changes
- **Jenkins** to managing automation across the promotion from development to production

Let's Look at an Example: Deployment

Say we want to build an Open Source centric stack

Requirement: Our system involves training an image classifier in a development environment, promoting the training code to a production environment, retraining the model in production environment, deploying as a REST endpoint, and daily monitoring of model performance

Deployment

Because we need an **real time** serving, **A/B testing** for promotion of new models, and **monitoring** for feature drift and label drift

- **Selden** for serving in real time and **A/B** testing
- Writing our own drift tests using **python** and managing the notifications through **Slack integrations**

Resources

AI Landscape