MLOPs with DataBricks

Introduction

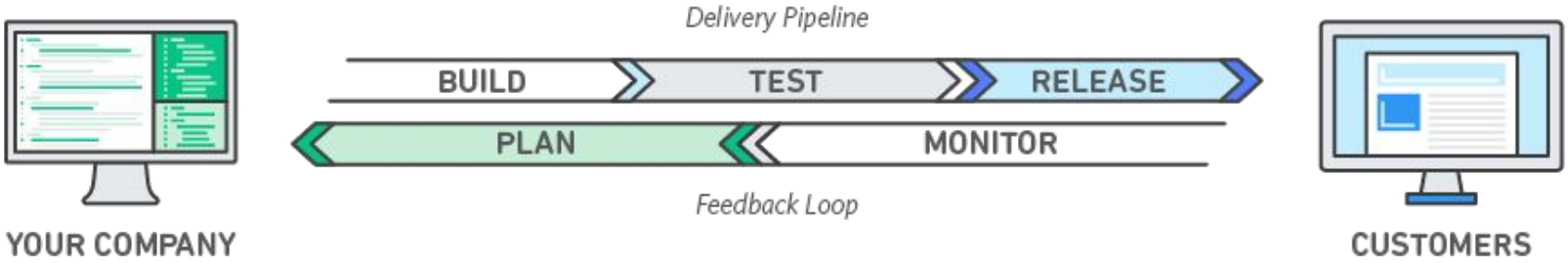
Introduction to DevOps, MLOps and Databricks

Agenda

1. Intro to Devops
2. Intro to MLOps
3. Databricks
4. Hands on Azure DataBricks

What is Devops?

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



Benefits of DevOps

Speed

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker.

Rapid Delivery

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage. Continuous integration and continuous delivery are practices that automate the software release process, from build to deploy.

Reliability

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.

Scale

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

Improved Collaboration

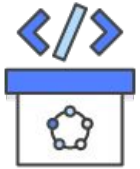
Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

Security

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

Continuous integration and continuous delivery (CI/CD)

Continuous Integration (CI) is the practice used by development teams to automate, merge, and test code. CI helps to catch bugs early in the development cycle, which makes them less expensive to fix. Automated tests execute as part of the CI process to ensure quality.



Continuous Delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production environments. Deploying and testing in multiple environments increases quality. CD systems produce deployable artifacts, including infrastructure and apps.

Continuous integration and continuous delivery (CI/CD) Tools

Continuous Integration (CI) -

Jenkins: Jenkins is one of the most widely used open-source CI/CD automation servers. It allows you to automate building, testing, and deployment processes and can integrate with a wide range of plugins and tools.

Travis CI: Travis CI is a cloud-based CI service that's popular for open-source projects. It provides easy integration with GitHub repositories and supports multiple programming languages.

CircleCI: CircleCI is a cloud-based CI/CD platform that offers powerful configuration options for building, testing, and deploying applications. It's known for its speed and ease of use.

GitLab CI/CD: GitLab offers built-in CI/CD capabilities as part of its DevOps platform. It provides a single interface for managing source code and CI/CD pipelines.

TeamCity: TeamCity is a CI/CD server developed by JetBrains. It's known for its user-friendly interface and powerful build configuration options.

Azure DevOps

Version Control

Version control is the practice of managing code in versions—tracking revisions and change history to make code easy to review and recover. This practice is usually implemented using version control systems such as Git, which allow multiple developers to collaborate in authoring code. These systems provide a clear process to merge code changes that happen in the same files, handle conflicts, and roll back changes to earlier states.

Version Control

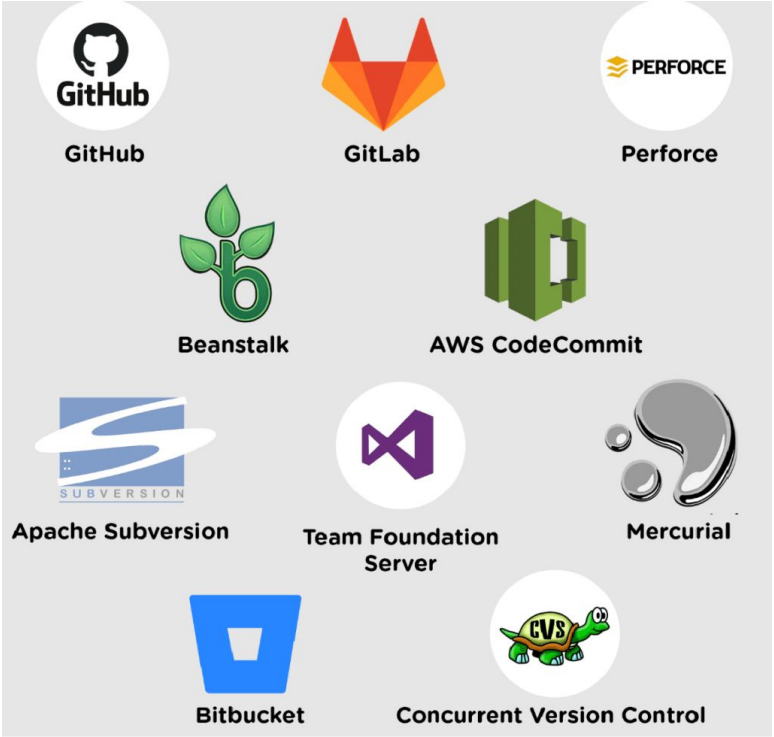
Git: Git is one of the most widely used distributed version control systems (DVCS). It's known for its speed, flexibility, and strong branching and merging capabilities. GitHub, GitLab, and Bitbucket are popular hosting platforms for Git repositories.

Subversion (SVN): Subversion is a centralized version control system that has been in use for many years. It's known for its simplicity and ease of use, especially in scenarios where distributed workflows are not a requirement.

Mercurial: Mercurial is another distributed version control system that is easy to learn and use. It's well-suited for projects of all sizes and offers a straightforward and consistent interface.

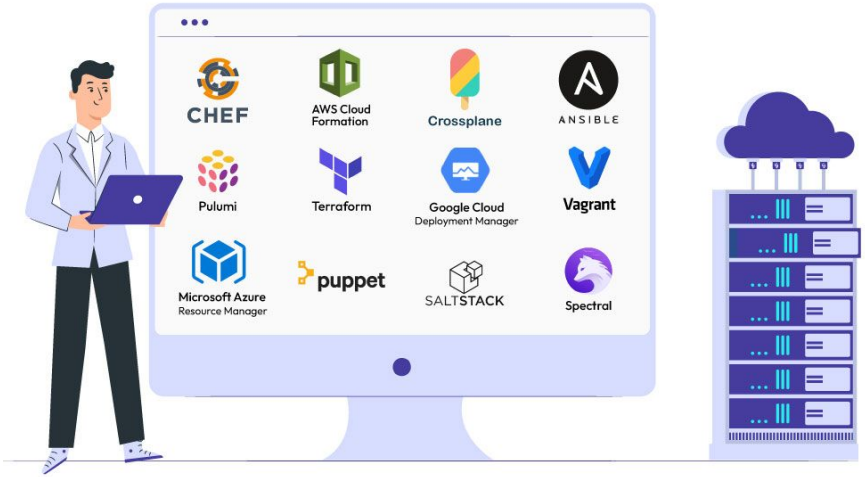
AWS CodeCommit: AWS CodeCommit is a managed Git service provided by Amazon Web Services (AWS). It integrates seamlessly with other AWS services and is suitable for cloud-native development.

Microsoft Team Foundation Version Control (TFVC): TFVC is a centralized version control system that is part of Microsoft's Azure DevOps Services (formerly known as Visual Studio Team Services). It integrates well with the Microsoft development ecosystem.



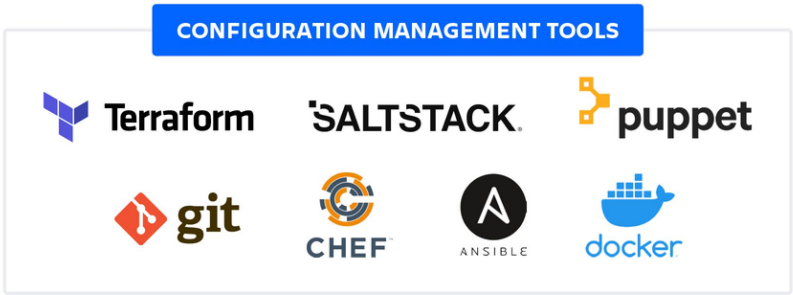
Infrastructure as code

Infrastructure as code defines system resources and topologies in a descriptive manner that allows teams to manage those resources as they would code. Those definitions can also be stored and versioned in version control systems, where they can be reviewed and reverted—again like code.



Configuration management

Configuration management refers to managing the state of resources in a system including servers, virtual machines, and databases. Using configuration management tools, teams can roll out changes in a controlled, systematic way, reducing the risks of modifying system configuration. Teams use configuration management tools to track system state and help avoid configuration drift, which is how a system resource's configuration deviates over time from the desired state defined for it.



Infrastructure as code

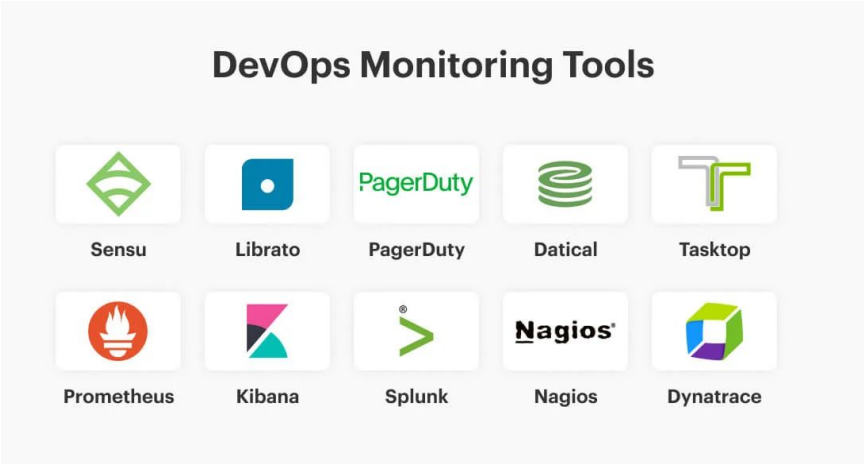
Infrastructure as code (IAC) is the managing and provisioning of infrastructure through code instead of using a manual process to configure devices or systems.

Configuration Management as code

Configuration as code is the practice of managing configuration files in a repository. Config files establish the parameters and settings for applications, server processing, operating systems.

Continuous monitoring

Continuous monitoring means having full, real-time visibility into the performance and health of the entire application stack. This visibility ranges from the underlying infrastructure running the application to higher-level software components. Visibility is accomplished through the collection of telemetry and metadata and setting of alerts for predefined conditions that warrant attention from an operator.



Planning

In the planning phase, DevOps teams ideate, define, and describe the features and capabilities of the applications and systems they plan to build. Teams track task progress at low and high levels of granularity, from single products to multiple product portfolios. Teams use the following DevOps practices to plan with agility and visibility:

- Create backlogs.
- Track bugs.
- Manage Agile software development with Scrum.
- Use Kanban boards.
- Visualize progress with dashboards.

Development

The development phase includes all aspects of developing software code. In this phase, DevOps teams do the following tasks:

Select a development environment.

- Write, test, review, and integrate the code.
- Build the code into artifacts to deploy into various environments.
- Use version control, usually Git, to collaborate on code and work in parallel.
- To innovate rapidly without sacrificing quality, stability, and productivity, DevOps teams

Use highly productive tools.

- Automate mundane and manual steps.
- Iterate in small increments through automated testing and continuous integration (CI).

Deliver

Delivery is the process of consistently and reliably deploying applications into production environments, ideally via continuous delivery (CD).

In the delivery phase, DevOps teams:

- Define a release management process with clear manual approval stages.
- Set automated gates to move applications between stages until final release to customers.
- Automate delivery processes to make them scalable, repeatable, controlled, and well-tested.
- Delivery also includes deploying and configuring the delivery environment's foundational infrastructure.

DevOps teams use technologies like **infrastructure as code (IaC), containers, and microservices** to deliver fully governed infrastructure environments.

Operations

The operations phase involves maintaining, monitoring, and troubleshooting applications in production environments, including hybrid or public clouds like Azure.

DevOps teams aim for system reliability, high availability, strong security, and zero downtime.

Automated delivery and safe deployment practices help teams identify and mitigate issues quickly when they occur. Maintaining vigilance requires rich telemetry, actionable alerting, and full visibility into applications and underlying systems.

DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

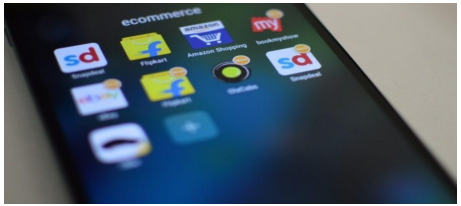
Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Container Solutions

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

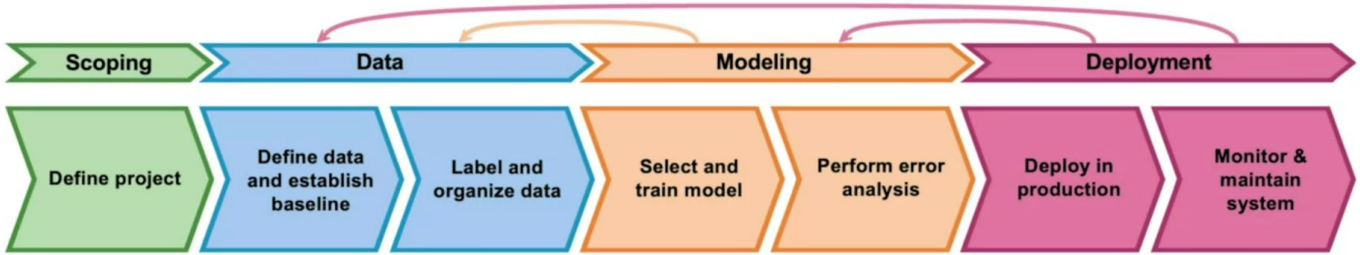
AI is everywhere



ChatGPT

What is MLOPs?

Data Science Lifecycle

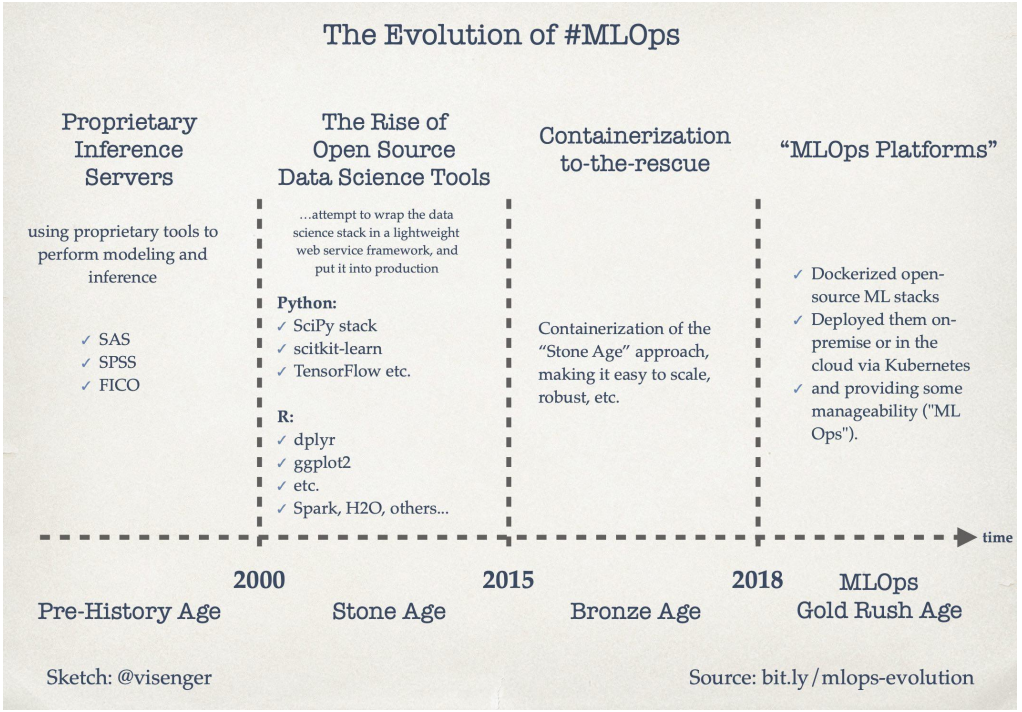


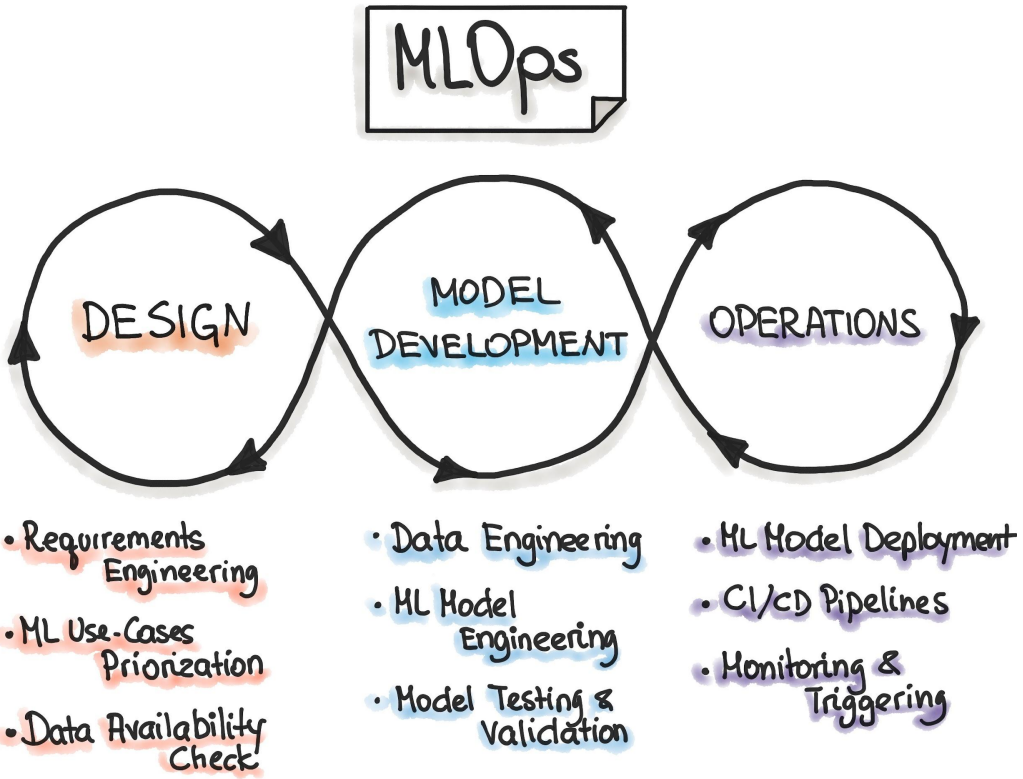
01	Data Extraction	• You select and integrate the relevant data from various data sources for the ML task.
02	Data Analysis	• You perform exploratory data analysis (EDA) to understand the available data for building the ML model.
03	Data preparation	• This preparation involves data cleaning, where you split the data into training, validation, and test sets.
04	Model Training	• The data scientist implements different algorithms with the prepared data to train various ML models and tune them.
05	Model evaluation	• The model is evaluated on a holdout test set to evaluate the model quality. The output of this step is a set of metrics to assess the quality of the model.
06	Model serving / Deployment	• The validated model is deployed to a target environment to serve predictions
07	Model Monitoring	• The model predictive performance is monitored to potentially invoke a new iteration in the ML process.

MLOPS

The definition of MLOps (machine learning operations) includes the culmination of **people, processes, practices and underpinning technologies that automate the deployment, monitoring, and management of machine learning (ML) models** into production in a scalable and fully governed way to finally provide measurable business value from machine learning.

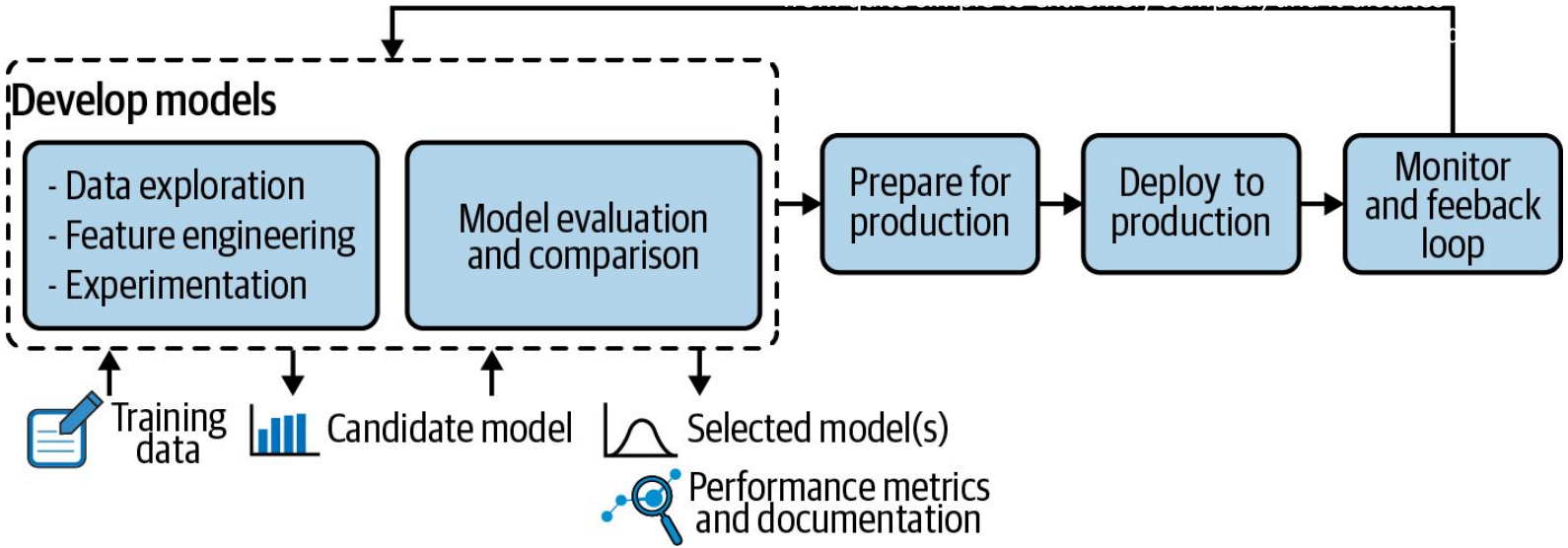
Laying an MLOps foundation allows data, development, and production teams to work collaboratively and leverage automation to deploy, monitor, and govern machine learning services and initiatives within an organization.





The term MLOps is defined as “the extension of the DevOps methodology to include Machine Learning and Data Science assets as first-class citizens within the DevOps ecology

Developing Models



DevOps vs MLOps

DEVOPS vs MLOPS

DevOps is a set of practices in the traditional software development world that enables faster, more reliable software development into production. DevOps relies on automation, tools, and workflows to abstract accidental complexity away and allow developers to focus on more critical issues.

The reason that DevOps is not simply applied to ML is that ML is not merely code, but **code and data**.

A data scientist creates an ML model that is eventually placed in production by applying an algorithm to training data. This huge amount of data affects the model's behavior in production. The behavior of the model also hinges on the input data that it receives at the time of prediction.

DEVOPS vs MLOPS

ML systems are similar to other software systems in many ways, but there are a few important differences:

CI includes testing and validating data, data schemas, and models, rather than being limited to testing and validating code and components.

CD refers to an ML training pipeline, a system that should deploy a model prediction service automatically, rather than a single software package or service.

In addition, ML systems feature **continuous testing (CT), a property that deals with retraining and serving models automatically** to ensure that the team can better assess business risks using immediate feedback throughout the process.

DEVOPS vs MLOPS

Teams: The team working on an ML project typically includes data scientists who focus on model development, exploratory data analysis, research, and experimentation. In contrast to team members on the DevOps side, these team members might not be capable of building production-class services as experienced software engineers are.

Development. To rapidly identify what best addresses the problem, ML is by necessity experimental. Team members test and tweak various algorithms, features, modeling techniques, and parameter configurations in this vein, but this creates challenges. Maximizing the reusability of code and maintaining reproducibility while tracking which changes worked and which failed are chief among them.

Testing. Compared to other software systems, testing an ML system is much more involved. Both kinds of systems require typical integration and unit tests, but ML systems also demand model validation, data validation, and quality evaluation of the trained model.

Deployment. ML models trained offline are not simply deployed in ML systems in real-world conditions for reliable predictions and results. Many situations demand a multi-step pipeline to automatically retrain the model before deployment. To create and deploy this kind of machine learning project pipeline, it is necessary to automate ML steps that data scientists complete manually before deployment as they validate and train the new models—and this is very complex.

Production. ML models are subject to more sources of decay than are conventional software systems, such as data profiles that are constantly changing and suboptimal coding, and it is essential to consider this degradation and reduced performance. Tracking summary data statistics and monitoring online model performance is critical, and the system should be set to catch values that deviate from expectations and either send notifications or roll back when they occur.

ML in research vs. in production

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting

Data

Research	Production
● Clean ● Static ● Mostly historical data	● Messy ● Constantly shifting ● Historical + streaming data ● Biased, and you don't know how biased ● Privacy + regulatory concerns

THE COGNITIVE CODER

By Armand Ruiz, Contributor, InfoWorld | SEP 26, 2017 7:22 AM PDT

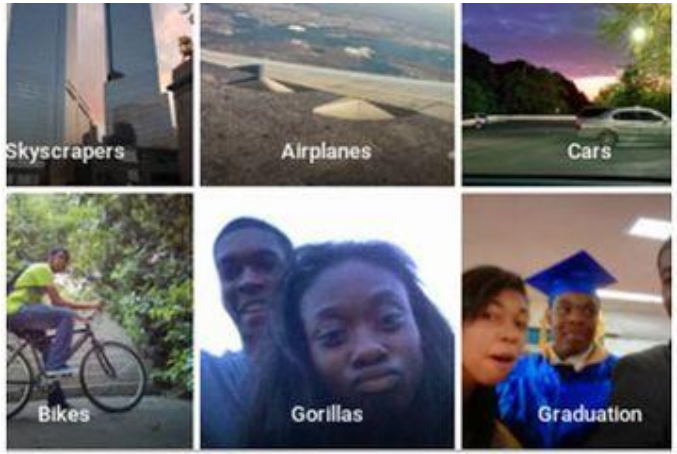
The 80/20 data science dilemma

Most data scientists spend only 20 percent of their time on actual data analysis and 80 percent of their time finding, cleaning, and reorganizing huge amounts of data, which is an inefficient data strategy

ML in research vs. in production

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important

Fairness

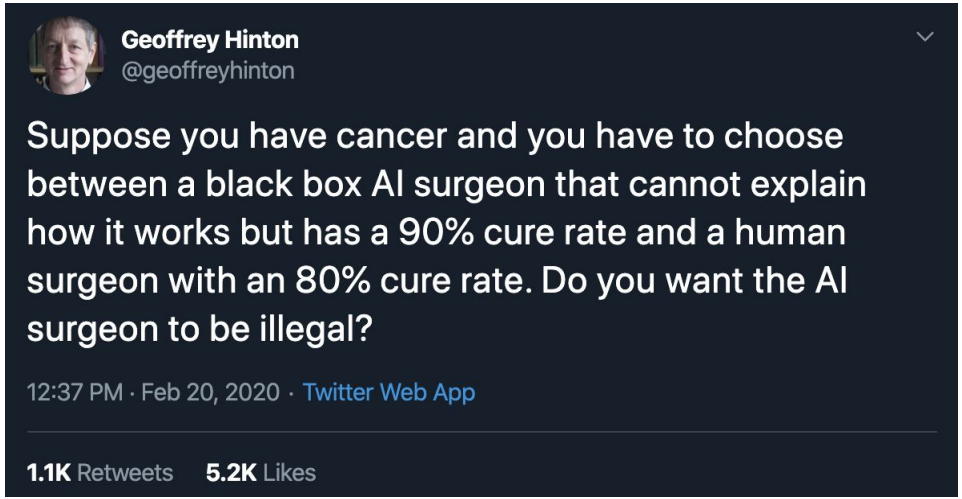


The Berkeley study found that both face-to-face and online lenders rejected a total of 1.3 million creditworthy black and Latino applicants between 2008 and 2015. Researchers said they believe the applicants "would have been accepted had the applicant not been in these minority groups." That's because when they used the income and credit scores of the rejected applications but deleted the race identifiers, the mortgage application was accepted.

ML in research vs. in production

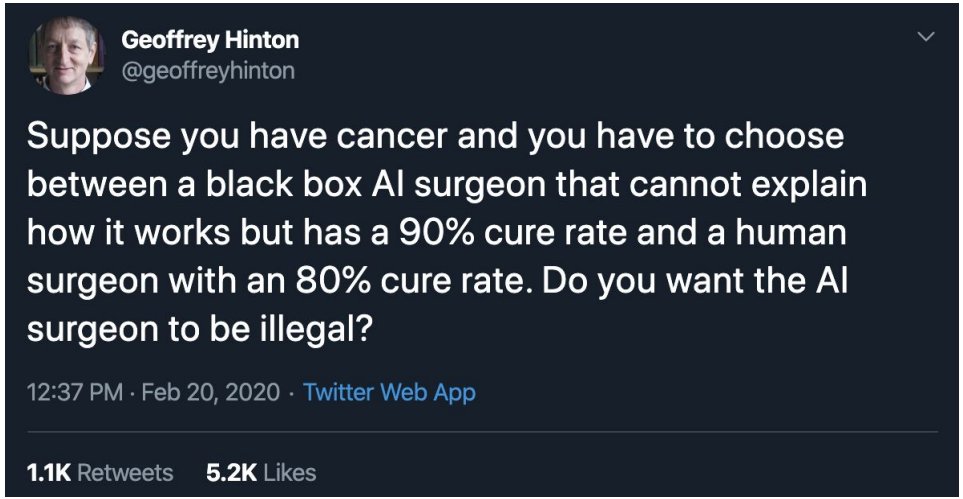
	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important
Interpretability*	Good to have	Important

Interpretability



Zoom poll: which one would you want as your surgeon?

Interpretability



ML in research vs. in production

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important
Interpretability	Good to have	Important

Myth #1: Deploying is hard

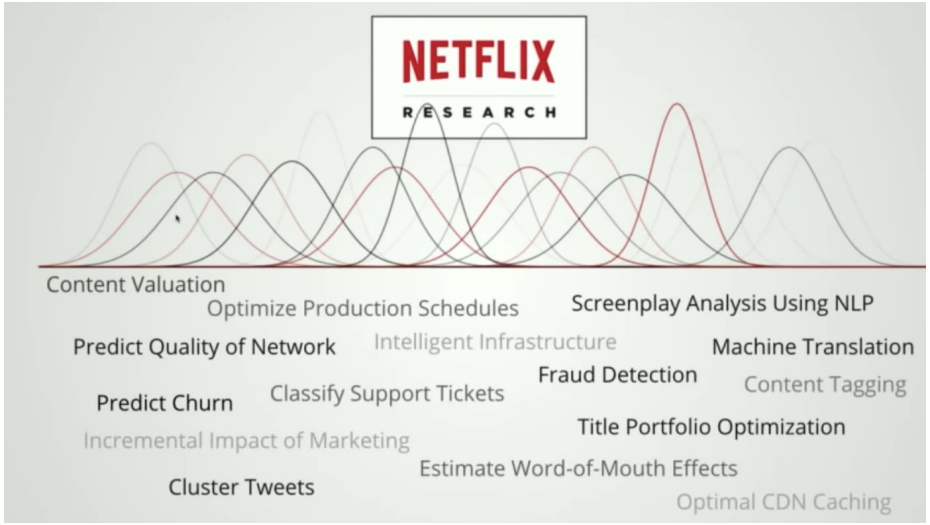
Myth #1: Deploying is hard

Deploying is easy. Deploying reliably is hard

Myth #2: You only deploy one or two ML models at a time

Myth #2: You only deploy one or two ML models at a time

Booking.com: 150+ models, Uber: thousands



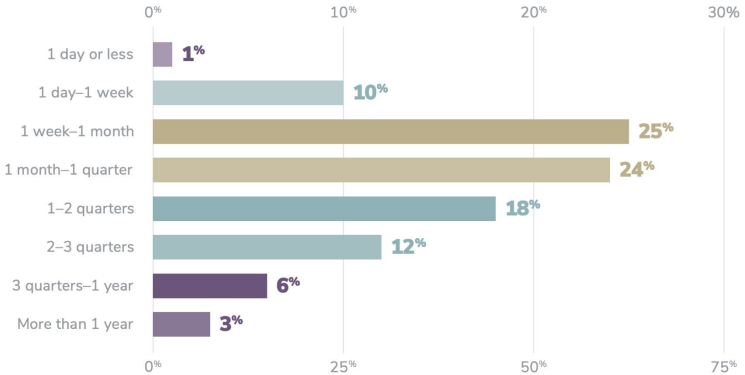
Myth #3: You won’t need to update your models as much

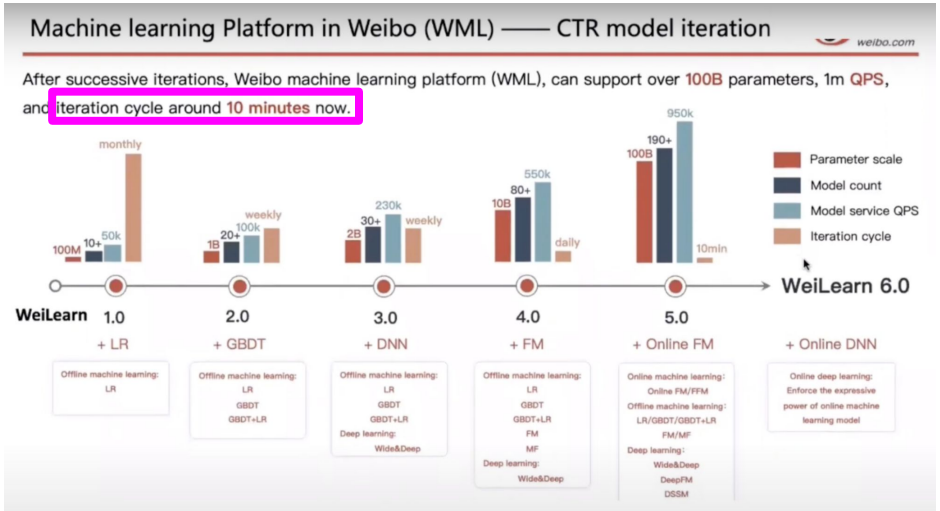
DevOps: Pace of software delivery is accelerating

- Elite performers deploy **973x** more frequently with **6570x** faster lead time to deploy (Google DevOps Report, 2021)
- DevOps standard (2015)
  - Etsy deployed 50 times/day
  - Netflix 1000s times/day
  - AWS every 11.7 seconds

DevOps to MLOps: Slow vs. Fast

Only 11% of organizations can put a model into production within a week, and 64% take a month or longer





Left image from Algorithmia | Right image: Machine learning with Flink in Weibo (Qian Yu, QCon 2019)

Accelerating ML Delivery

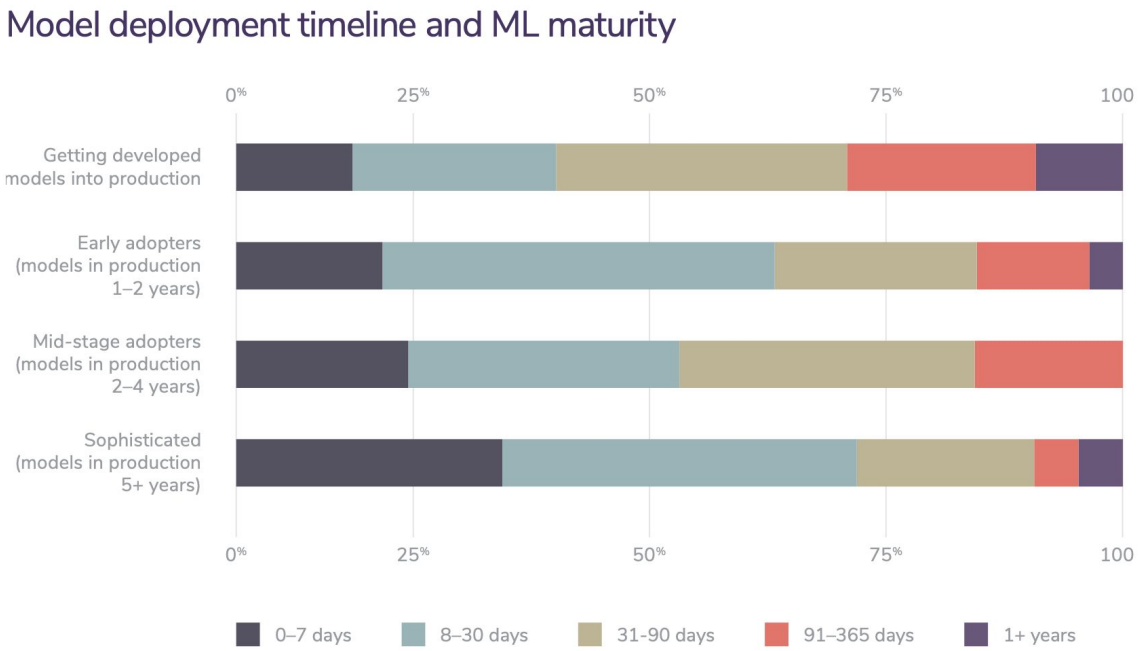
ML + DevOps = 

Myth #4: ML can magically transform your business overnight

Myth #4: ML can magically transform your business overnight

Magically: possible
Overnight: no

Efficiency improves with maturity



2020 state of enterprise machine learning (Algorithmia, 2020)

ML engineering is more engineering than ML

MLEs might spend most of their time:

- wrangling data
- understanding data
- setting up infrastructure
- deploying models

instead of training ML models

Key Features of MLOPS

- ❑ Collaboration and Communication
- ❑ Version Control
- ❑ Automation
- ❑ Monitoring
- ❑ Scalability and Resource Management
- ❑ Security and Compliance
- ❑ Feedback Loops and Iterative Development
- ❑ Documentation and Reproducibility

ML Monitoring

Operational - Is it working?

- Latencies
- Memory size
- CPU usage

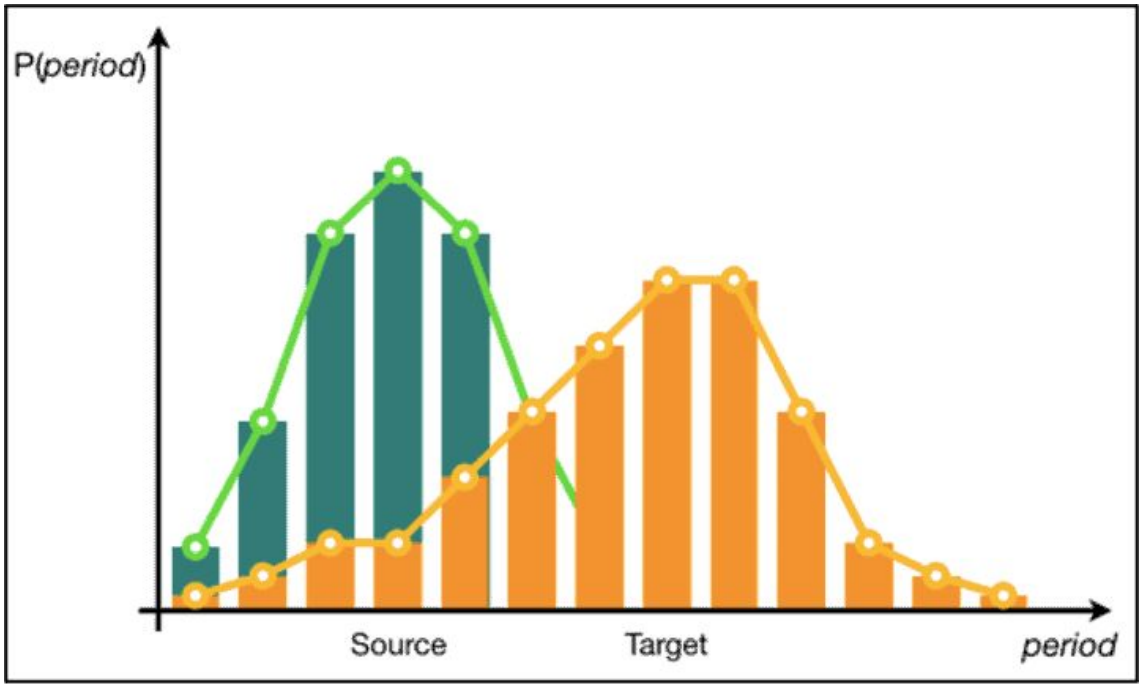
Are the predictions accurate?

- Model Outputs

Is the data what is expected?

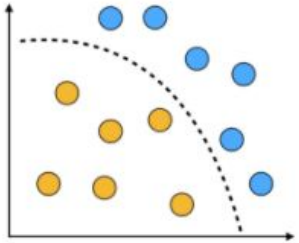
- Model Inputs

Data Drift



Data drift refers to the situation where the statistical properties of the incoming data used for model prediction change over time.
This change can be due to various factors such as shifts in the data distribution, changes in user behavior, or external events.
When data drift occurs, the data that the model encounters in production may differ significantly from the data it was trained on.

Concept Drift



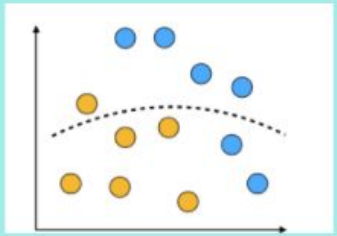
Training data with decision boundary

$P(Y|X)$

Probability of y output given x input

Concept Drift

$P(Y|X)$ Changes

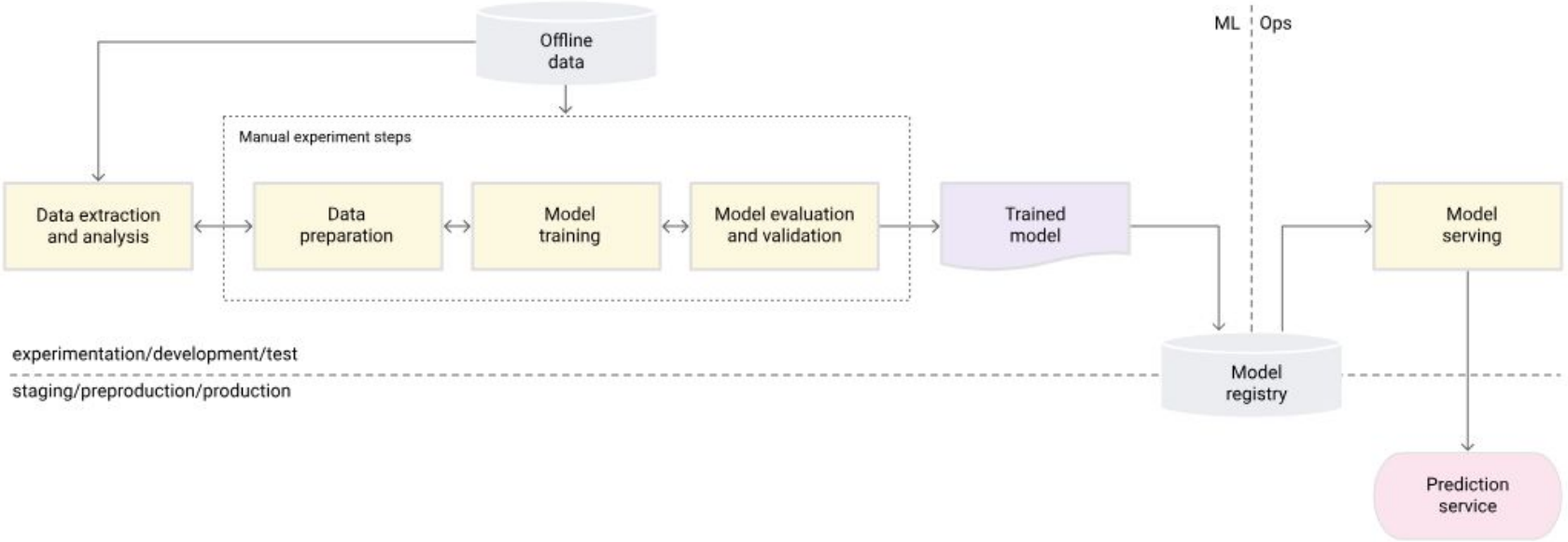


- Reality/behavioral change
- Relationships change, not the input

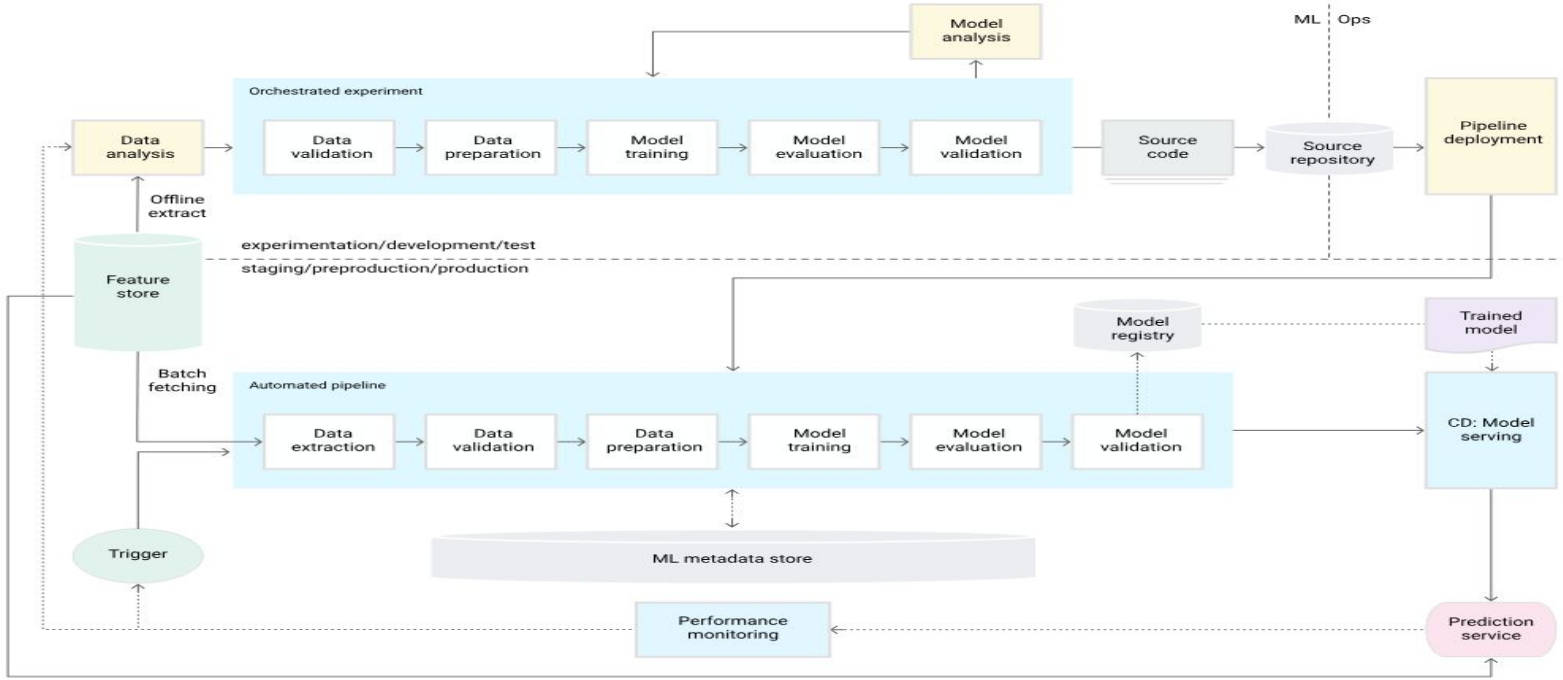
Concept drift is closely related to data drift but focuses on the underlying relationships between features and labels rather than just statistical properties. It occurs when the relationships between input features and the target variable change over time.
This means that the assumptions the model was built upon are no longer valid. Concept drift can be more subtle and challenging to detect than data drift. Continuous monitoring is necessary to identify concept drift and to retrain or update the model as needed to maintain its

MLOPS PIPELINE

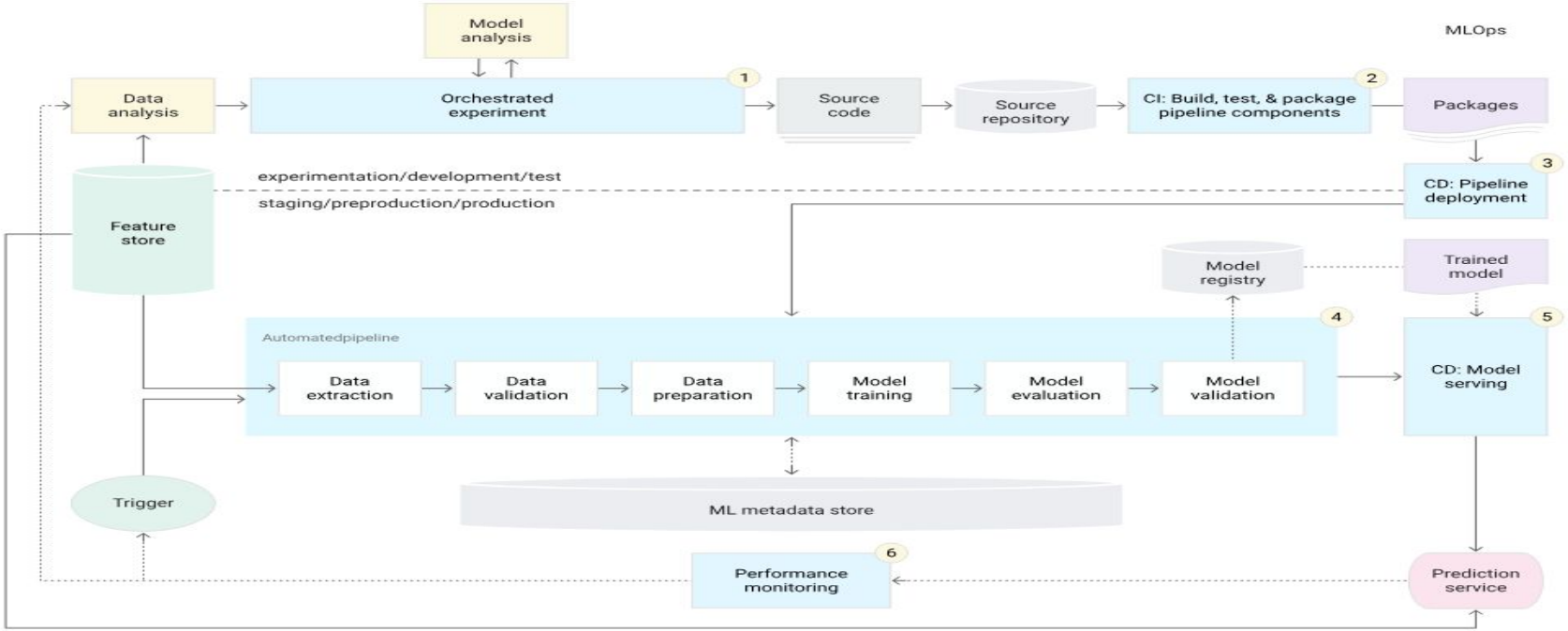
MLOps level 0: Manual process



MLOps level 1: ML pipeline automation



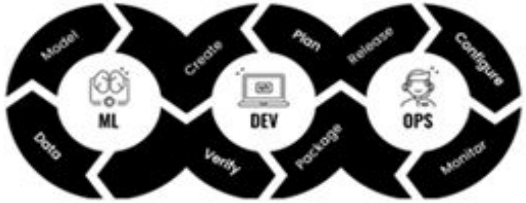
MLOps level 2: CI/CD pipeline automation



Designed by author Shanthababu

Machine Learning Operations (MLOps)

MLOps = ML + Dev + Ops



Line of Business

Business requirement

Machine Learning Lifecycle

Data Collection
- Collecting data from various data sources

SMEs
Data Scientists

Data Pipeline
- Data Pipeline Prep
- Data Clean up
- Data Transformation
- Data management Data drift monitoring
- Data support automation scripts/Tools

Data Engineer

Model Building
- Model selection
- Model training
- Evaluation of Scoring
- Experimentation on accuracy
- Model validation

ML Engineer

Model Deployment
- Scaling deployment model
- Model monitoring
- Model drift monitoring
- Model validation
- Assess the streaming and batch process

ML Engineer

DevOps
- CI/CD pipeline
- Testing model
- Monitoring
  - Logging
  - Notification
  - Scaling deployment model
- Versioning
  - Model
  - Code
  - Data
- Model drift monitoring

MLOps Engineers

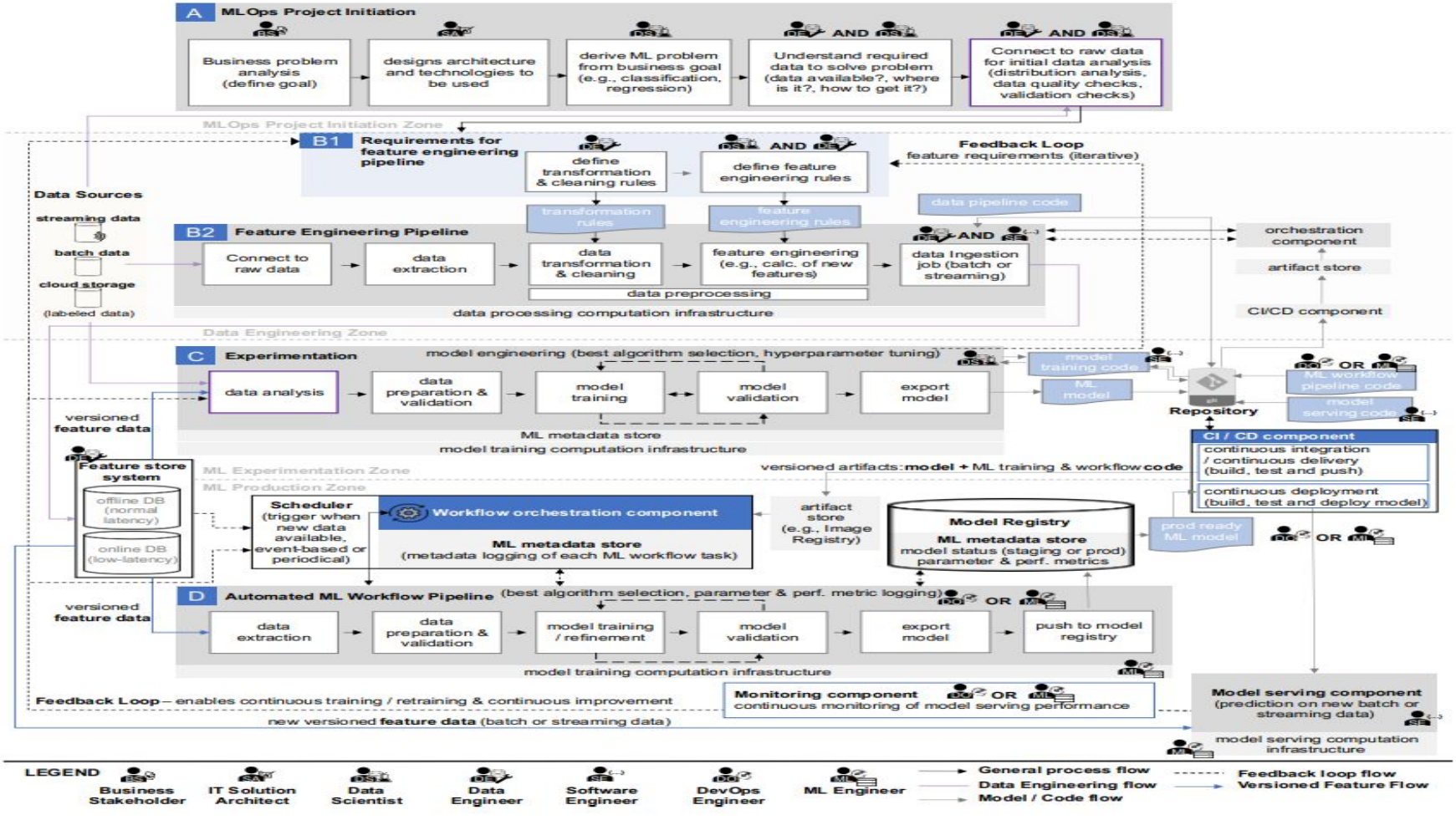
Catalogs
- Dataset
- Feature
- Model

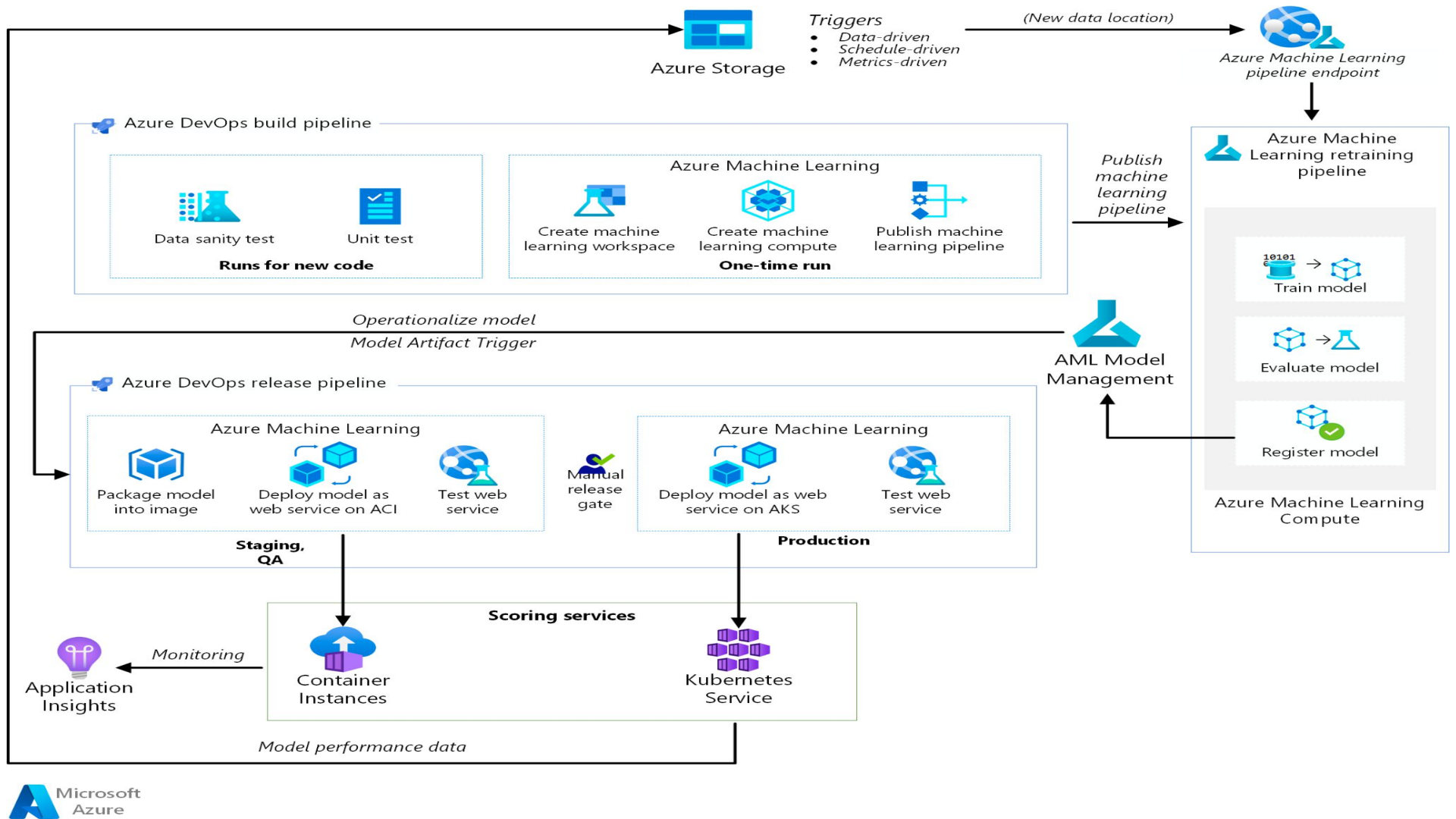
ML Archives

Governance
- Access control
- Security
- GDPR/CCPA

ML Governance

ML Architect


A MLOps Project Initiation
Business problem analysis (define goal)
designs architecture and technologies to be used
derive ML problem from business goal (e.g., classification, regression)
Understand required data to solve problem (data available?, where is it?, how to get it?)
Connect to raw data for initial data analysis (distribution analysis, data quality checks, validation checks)
AND
MLOps Project Initiation Zone
B1 Requirements for feature engineering pipeline
define transformation & cleaning rules
define feature engineering rules
transformation rules
feature engineering rules
Feedback Loop
feature requirements (iterative)
data pipeline code
Data Sources
streaming data
batch data
cloud storage
(labeled data)
B2 Feature Engineering Pipeline
Connect to raw data
data extraction
data transformation & cleaning
feature engineering (e.g., calc. of new features)
data Ingestion job (batch or streaming)
data preprocessing
data processing computation infrastructure
orchestration component
artifact store
CI/CD component
Data Engineering Zone
C Experimentation
model engineering (best algorithm selection, hyperparameter tuning)
data analysis
data preparation & validation
model training
model validation
export model
model training code
ML model
OR
ML workflow pipeline code
model serving code
Repository
versioned feature data
ML metadata store
model training computation infrastructure
CI / CD component
continuous integration / continuous delivery (build, test and push)
continuous deployment (build, test and deploy model)
Feature store system
offline DB (normal latency)
online DB (low-latency)
ML Experimentation Zone
ML Production Zone
versioned artifacts: model + ML training & workflow code
Scheduler (trigger when new data available, event-based or periodical)
Workflow orchestration component
ML metadata store (metadata logging of each ML workflow task)
artifact store (e.g., Image Registry)
Model Registry
ML metadata store
model status (staging or prod)
parameter & perf. metrics
prod ready ML model
D Automated ML Workflow Pipeline
(best algorithm selection, parameter & perf. metric logging)
data extraction
data preparation & validation
model training / refinement
model validation
export model
push to model registry
model training computation infrastructure
Feedback Loop – enables continuous training / retraining & continuous improvement
Monitoring component
continuous monitoring of model serving performance
new versioned feature data (batch or streaming data)
Model serving component
(prediction on new batch or streaming data)
model serving computation infrastructure
LEGEND
Business Stakeholder
IT Solution Architect
Data Scientist
Data Engineer
Software Engineer
DevOps Engineer
ML Engineer
General process flow
Data Engineering flow
Model / Code flow
Feedback loop flow
Versioned Feature Flow


Azure Storage
Triggers
Data-driven
Schedule-driven
Metrics-driven
(New data location)
Azure Machine Learning pipeline endpoint
Azure DevOps build pipeline
Data sanity test
Unit test
Runs for new code
Azure Machine Learning
Create machine learning workspace
Create machine learning compute
Publish machine learning pipeline
One-time run
Publish machine learning pipeline
Azure Machine Learning retraining pipeline
Train model
Evaluate model
Register model
Azure Machine Learning Compute
Operationalize model
Model Artifact Trigger
AML Model Management
Azure DevOps release pipeline
Azure Machine Learning
Package model into image
Deploy model as web service on ACI
Test web service
Staging, QA
Manual release gate
Azure Machine Learning
Deploy model as web service on AKS
Test web service
Production
Scoring services
Container Instances
Kubernetes Service
Monitoring
Application Insights
Model performance data
Microsoft Azure

DataBricks

Hardest Part of AI isn't AI, it's Data

"Hidden Technical Debt in Machine Learning Systems," Google NIPS 2015

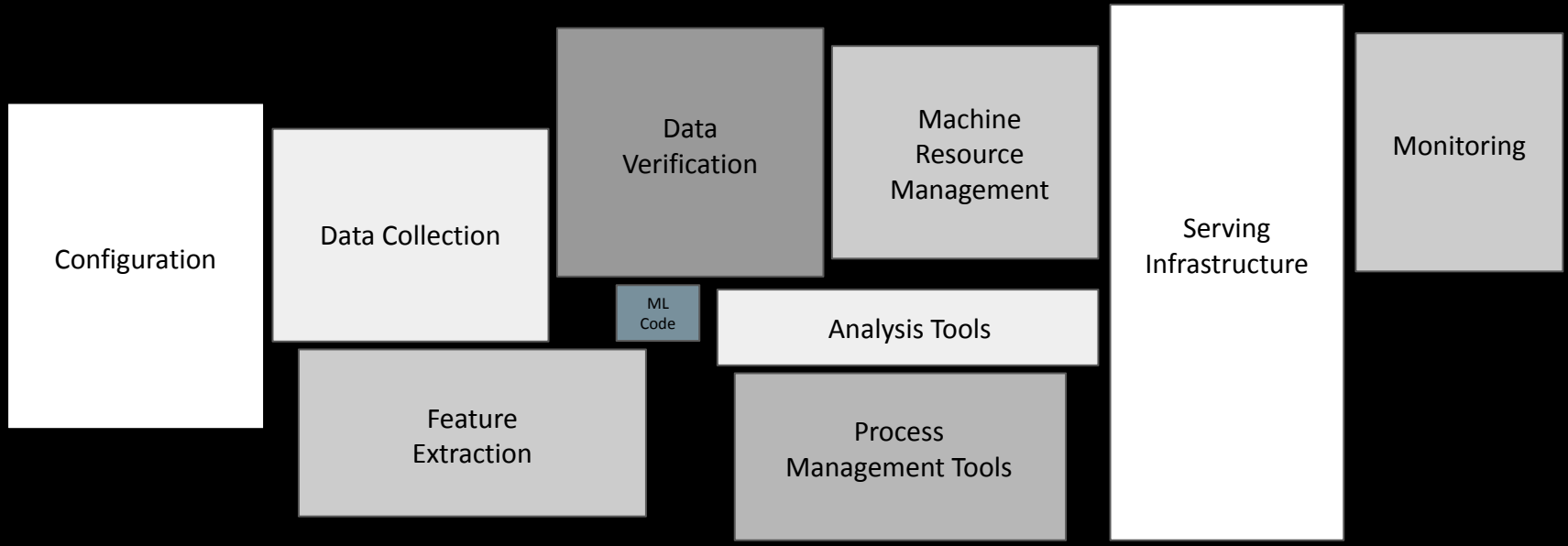
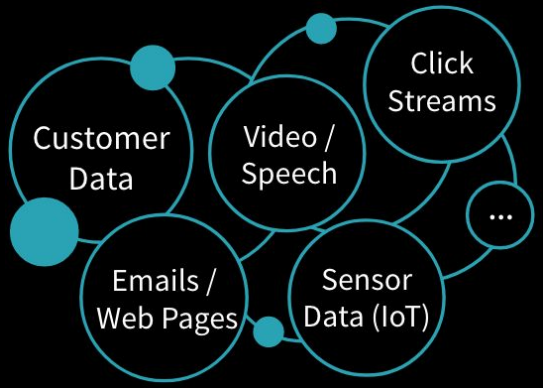
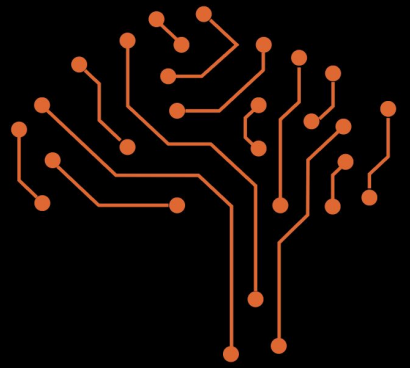


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small green box in the middle. The required surrounding infrastructure is vast and complex.

Data & AI Technologies are in Silos



X



SQL hadoop mongoDB Parquet

kafka ORACLE TERADATA

Great for Data, but not AI

K Keras scikit learn PYTORCH

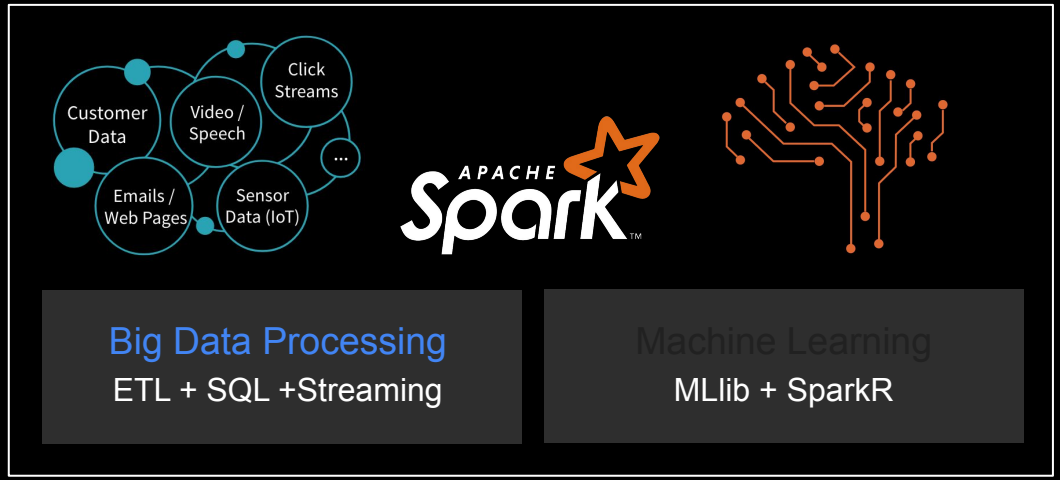
TensorFlow mxnet

Great for AI, but not for data

databricks

Apache Spark: The First Unified Analytics Engine

Uniquely combines Data & AI technologies



S3 Parquet hadoop kafka Azure

databricks

Enterprises face challenges beyond Apache Spark

Engineers

Disconnect

Scientists

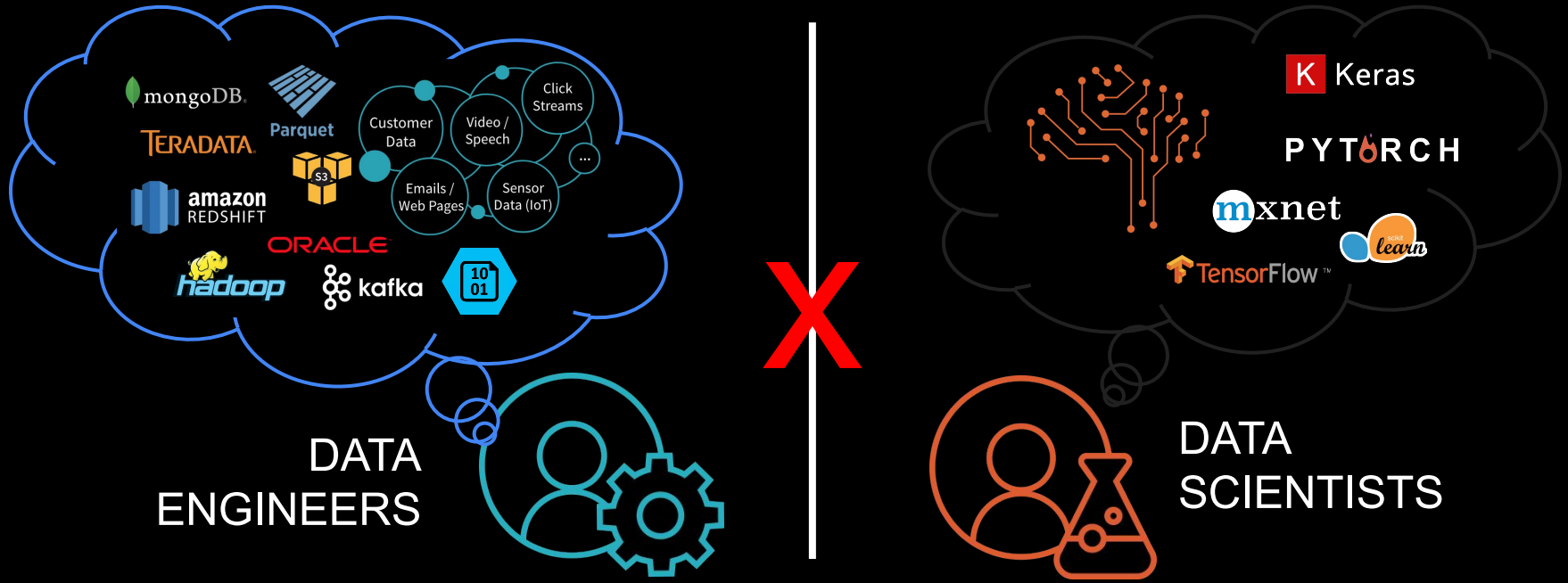
Complex data pipelines and infrastructure

Unified Analytics Engine

APACHE Spark™

databricks

Data & AI People are in Silos



Azure Databricks

AZURE DATA SOURCES

Blob Storage

Data Lake Store

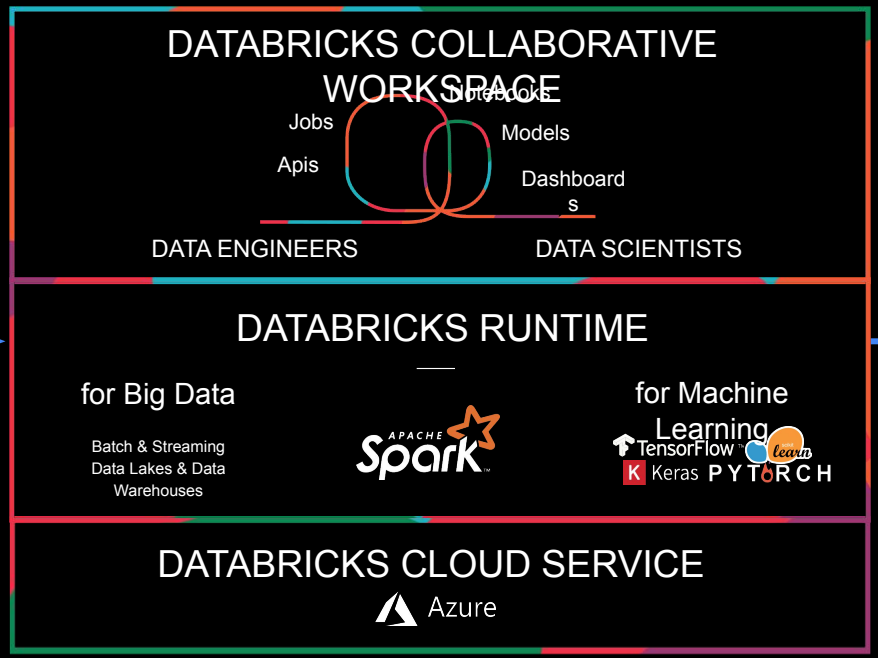
SQL Data Warehouse

Cosmos DB

Event Hub

IoT Hub

Azure Data Factory



Power BI

BI Reporting
Dashboards

Active Directory

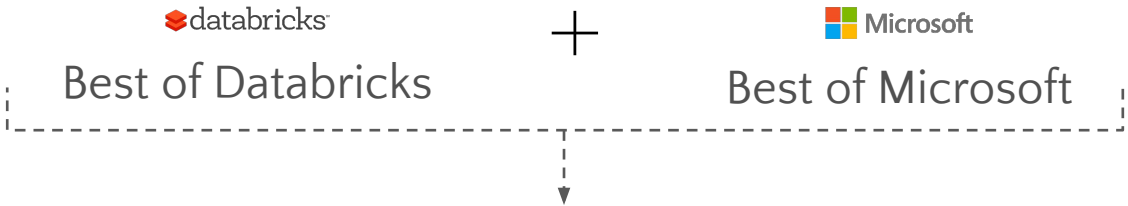
Security Integration

Azure Portal
One-Click setup
Unified Billing

databricks

What is Azure Databricks?

A fast, easy and collaborative Apache® Spark™ based analytics platform optimized for Azure



- Designed in collaboration with the founders of Apache Spark
- One-click set up; streamlined workflows
- Interactive workspace that enables collaboration between data scientists, data engineers, and business analysts.
- Native integration with Azure services (Power BI, SQL DW, Cosmos DB, Blob Storage)
- Enterprise-grade Azure security (Active Directory integration, compliance, enterprise-grade SLAs)

Differentiated experience on Azure

ENHANCE PRODUCTIVITY

Get started quickly by launching your new Spark environment with one click.

Share your insights in powerful ways through rich integration with Power BI.

Improve collaboration amongst your analytics team through a unified workspace.

Innovate faster with native integration with rest of Azure platform

BUILD ON THE MOST COMPLIANT CLOUD

Simplify security and identity control with built-in integration with Active Directory.

Regulate access with fine-grained user permissions to Azure Databricks' notebooks, clusters, jobs and data.

Build with confidence on the trusted cloud backed by unmatched support, compliance and SLAs.

SCALE WITHOUT LIMITS

Operate at massive scale without limits globally.

Accelerate data processing with the fastest Spark engine.

What is Azure Databricks?

Azure Databricks is a unified, open analytics platform for building, deploying, sharing, and maintaining enterprise-grade data, analytics, and AI solutions at scale. The Databricks Data Intelligence Platform integrates with cloud storage and security in your cloud account, and manages and deploys cloud infrastructure on your behalf.

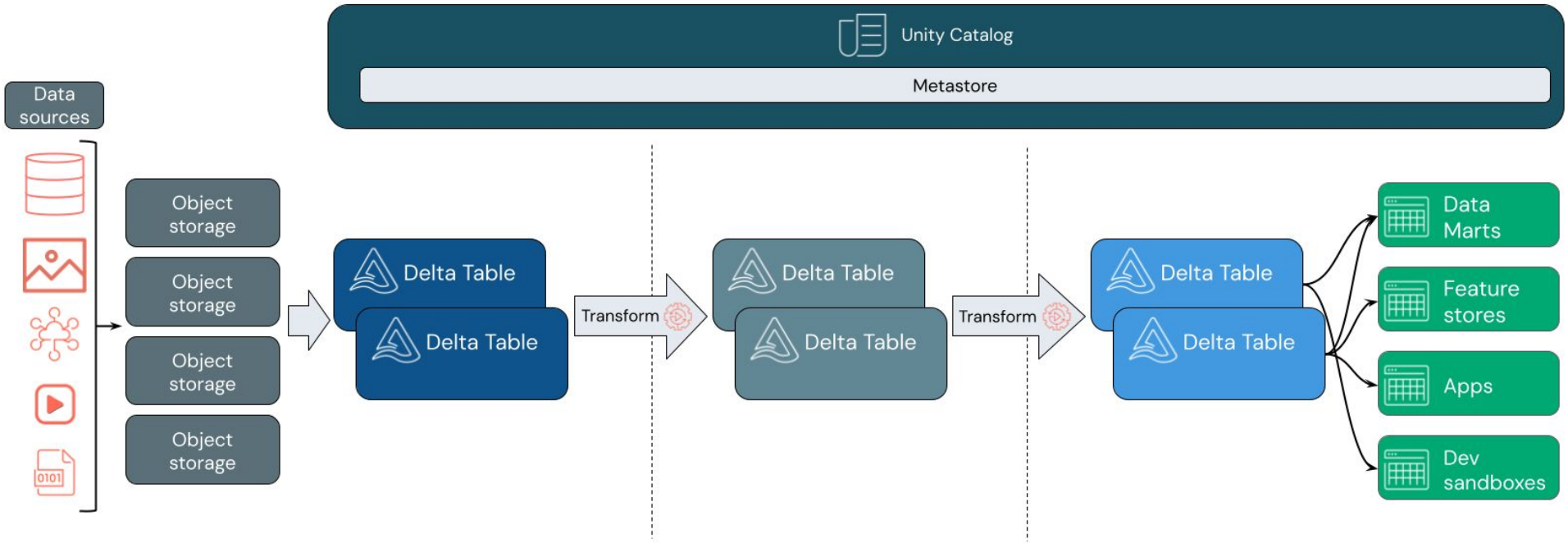
The Azure Databricks workspace provides a unified interface and tools for most data tasks, including:

- Data processing workflows scheduling and management
- Generating dashboards and visualizations
- Managing security, governance, high availability, and disaster recovery
- Data discovery, annotation, and exploration
- Machine learning (ML) modeling, tracking, and model serving
- Generative AI solutions

Databricks has a strong commitment to the open source community. Databricks manages updates of open source integrations in the Databricks Runtime releases. The following technologies are open source projects founded by Databricks employees:

- Delta Lake and Delta Sharing
- MLflow
- Apache Spark and Structured Streaming
- Redash

What is a data Lakehouse?



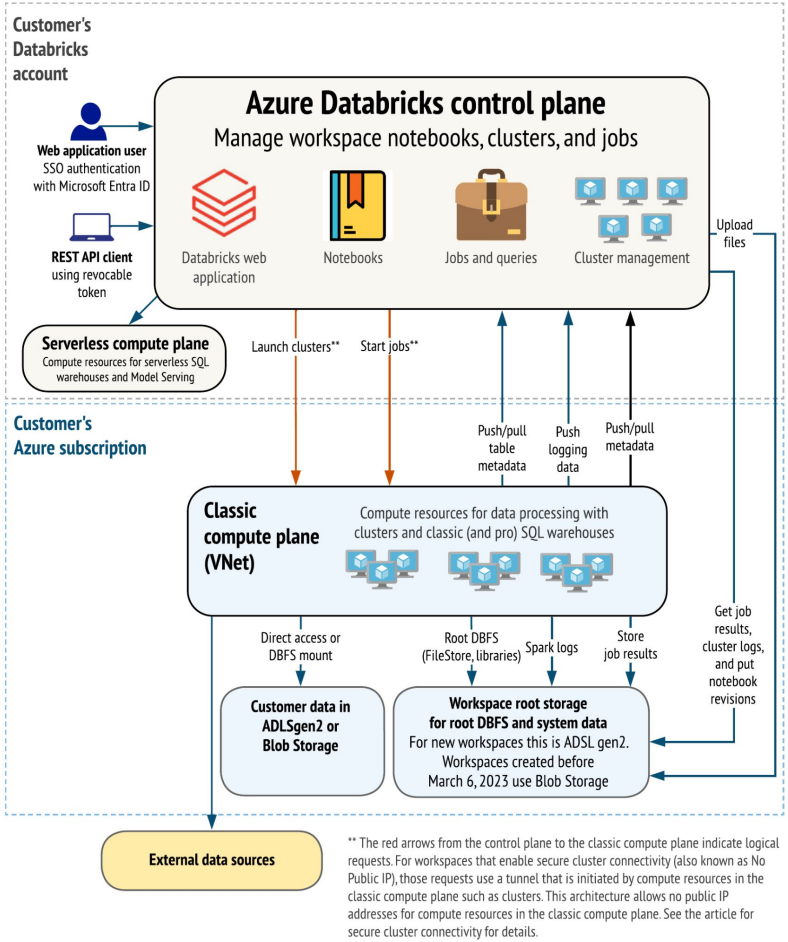
What is a Data Lakehouse used for?

A data Lakehouse provides scalable storage and processing capabilites for modern organizations who want to avoid a isolated systems for processing different workloads, like machine learning (ML) and business intelligence (BI).

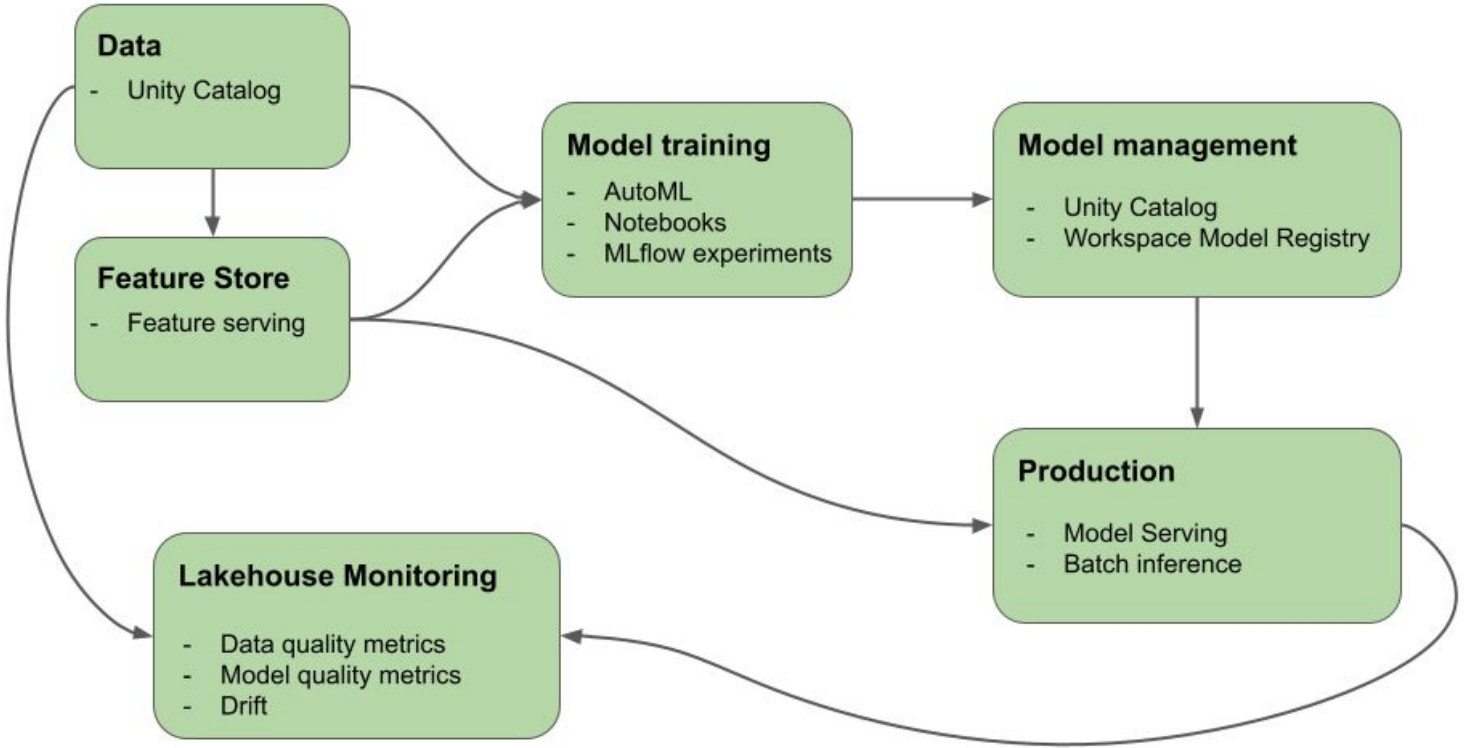
A Data Lakehouse can help establish a single source of truth, eliminate redundant costs, and ensure data freshness.

Delta Lake: an optimized storage layer that supports ACID transactions and schema enforcement.
Unity Catalog: a unified, fine-grained governance solution for data and AI.

Azure Databricks architecture overview



AI and Machine Learning on Databricks



Why use Databricks for machine learning and deep learning?

- Unity Catalog for governance, discovery, versioning, and access control for data, features, models, and functions.
- Lakehouse Monitoring for data monitoring.
- Feature engineering and serving.
- Support for the model lifecycle:
    - Databricks AutoML for automated model training.
    - MLflow for model development tracking.
    - Unity Catalog for model management.
    - Databricks Model Serving for high-availability, low-latency model serving.
    - Lakehouse Monitoring to track model prediction quality and drift.
- Databricks Workflows for automated workflows and production-ready ETL pipelines.
- Databricks Repos for code management and Git integration.

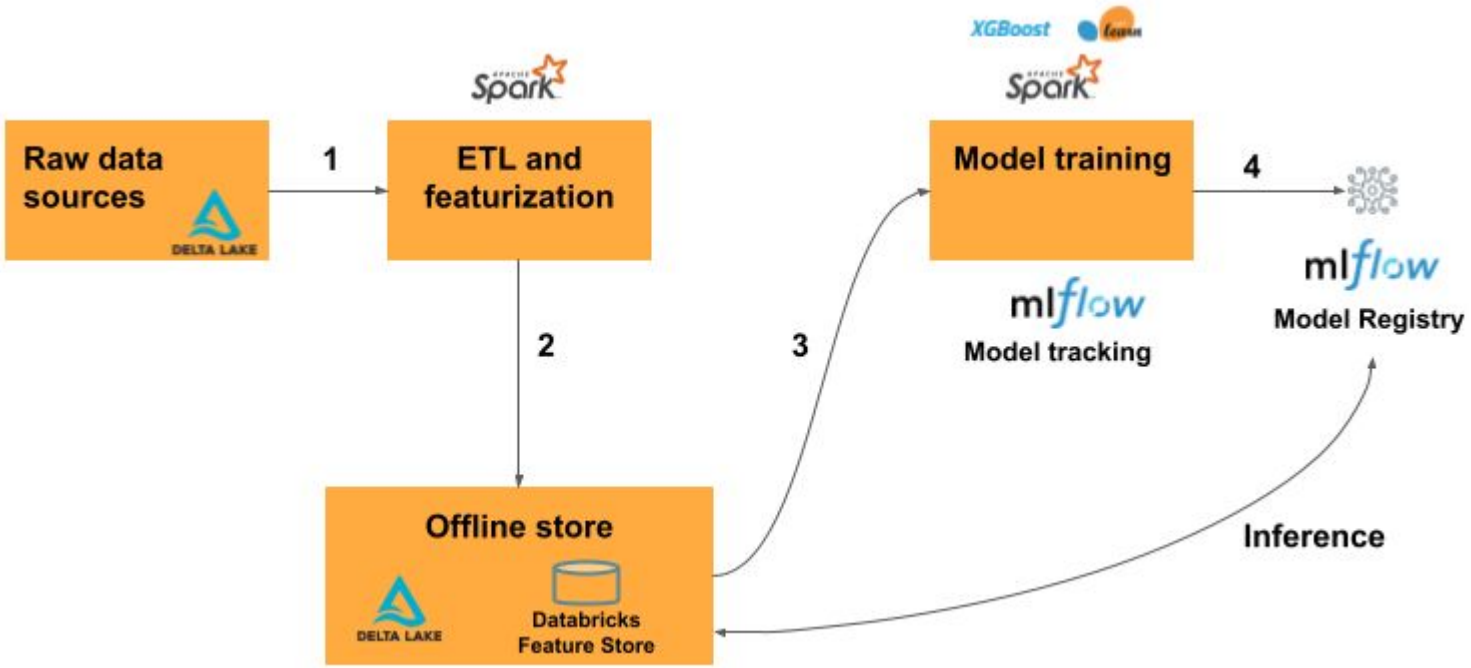
What is a feature store?

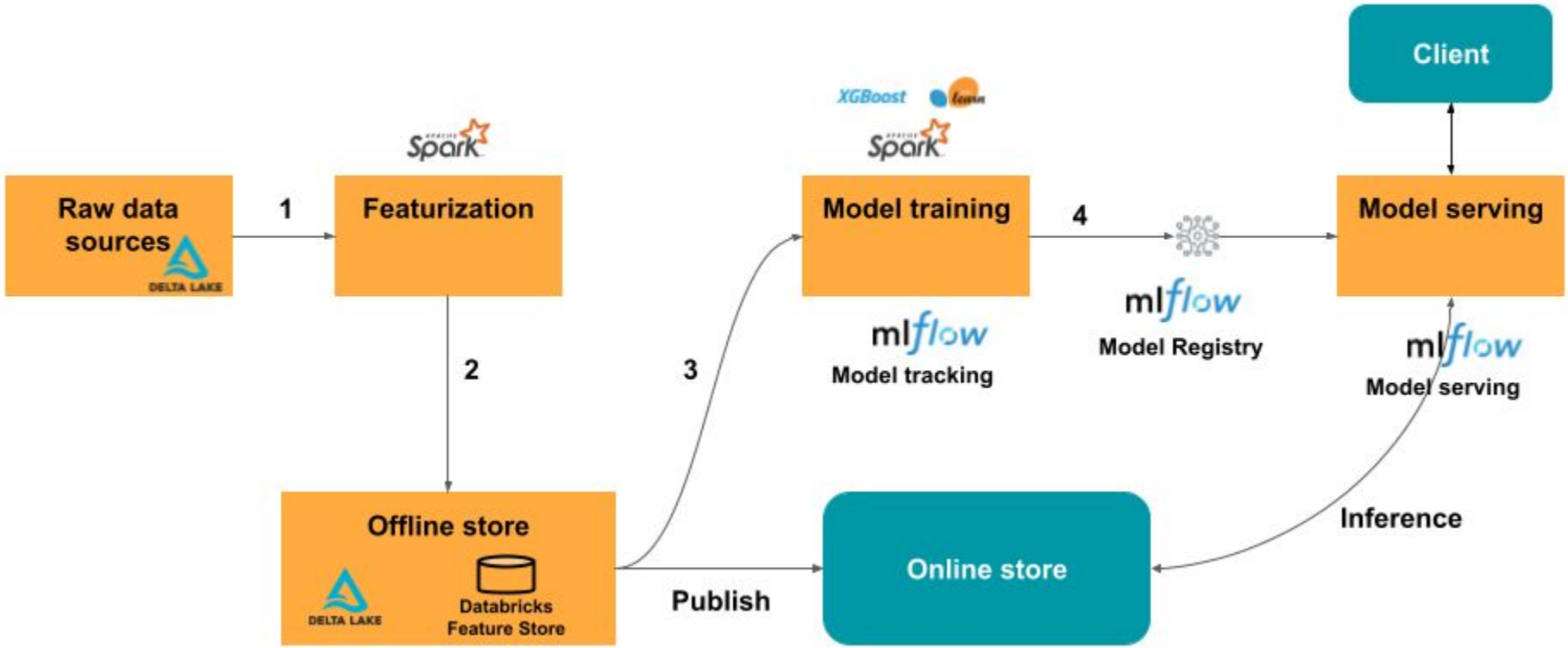
A feature store is a **centralized repository** that enables data scientists to find and share features and also ensures that the same code used to compute the feature values is used for model training and inference.

Machine learning uses existing data to build a model to predict future outcomes. In almost all cases, the raw data requires preprocessing and transformation before it can be used to build a model.

This process is called feature engineering, and the outputs of this process are called features - the building blocks of the model.

How does Databricks Feature Store work?




Client
Raw data sources
DELTA LAKE
1
Featurization
Spark
2
Offline store
DELTA LAKE
Databricks Feature Store
3
Model training
XGBoost
Spark
mlflow
Model tracking
4
mlflow
Model Registry
Model serving
mlflow
Model serving
Publish
Online store
Inference

HANDS ON

END