MLOps

Introduction

Outline

- Implementation and Automation
  - Continuous Integration (CI)
  - Continuous Delivery (CD)
  - Continuous Monitoring & Training (CT)

DevOps [CI | CD] + CM/CT ⇒ MLOps

DevOps:

1. Design & Requirements
2. Coding → Collaboration
   Pull / Change
3. Testing
4. Release / Deployment
5. Monitoring → Operations
6. Security

7. Integration to Cross Platform
8. Infra Provisioning
9.

Data Science & ML

- Capable of solving complex real-world problems, transforming businesses, and delivering values in all domains
- In forefront due to the access to
  - Large datasets
  - Inexpensive on-demand compute resources, cloud solutions
  - Advanced easy to use ML tools
  - Rapid advances in ML methods and applications
  - Many businesses investing in developing their own AI team

Recent advances in Data Science & ML

- Demand for Data Engineers, Data Scientists and ML Engineers
- Emergence of MLOps
  - Unifies AI system developments and AI system operations
  - Automates and monitors all steps in AI system construction, integration, testing, releasing, deployment, data collection, cleaning, feature engineering, feature selection, retraining and infrastructure management, etc.

DataOps

MLOps :- machine learning operations

Challenges

- AI deeply depend on Cloud Computing
- Raw ingredients of AI requires massive compute, extensive data and specialized hardware

H100 → 520

H200 [illegible]/hr ≈ 200/hr

AI/ML Systems

- AI systems mainly focus on data engineering, data processing, problem feasibility and business alignment
- Primary focus is on business with ML rather than code “Highest Paid Person's Opinion” (HIPPO) effect
- Most AI systems are not code native, use academic software packages that do not scale for large-scale problems

DNN / Imm ady

What is MLOps?

- The process of automating AI system using DevOps methodologies
- Share

Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.

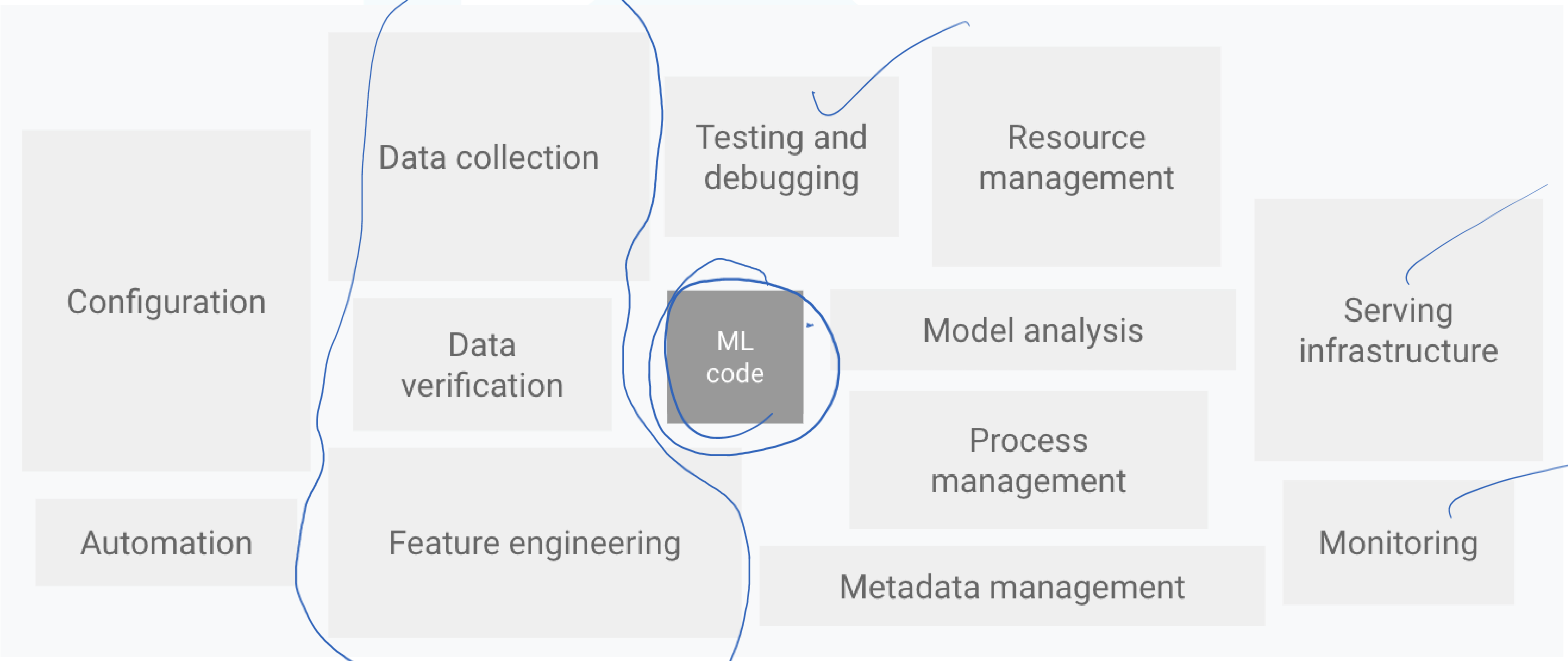
Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison
{ebner,vchaudhary,mwyoung,jfcrespo,dennison}@google.com
Google, Inc.

Source: Google Cloud

What is MLOps?

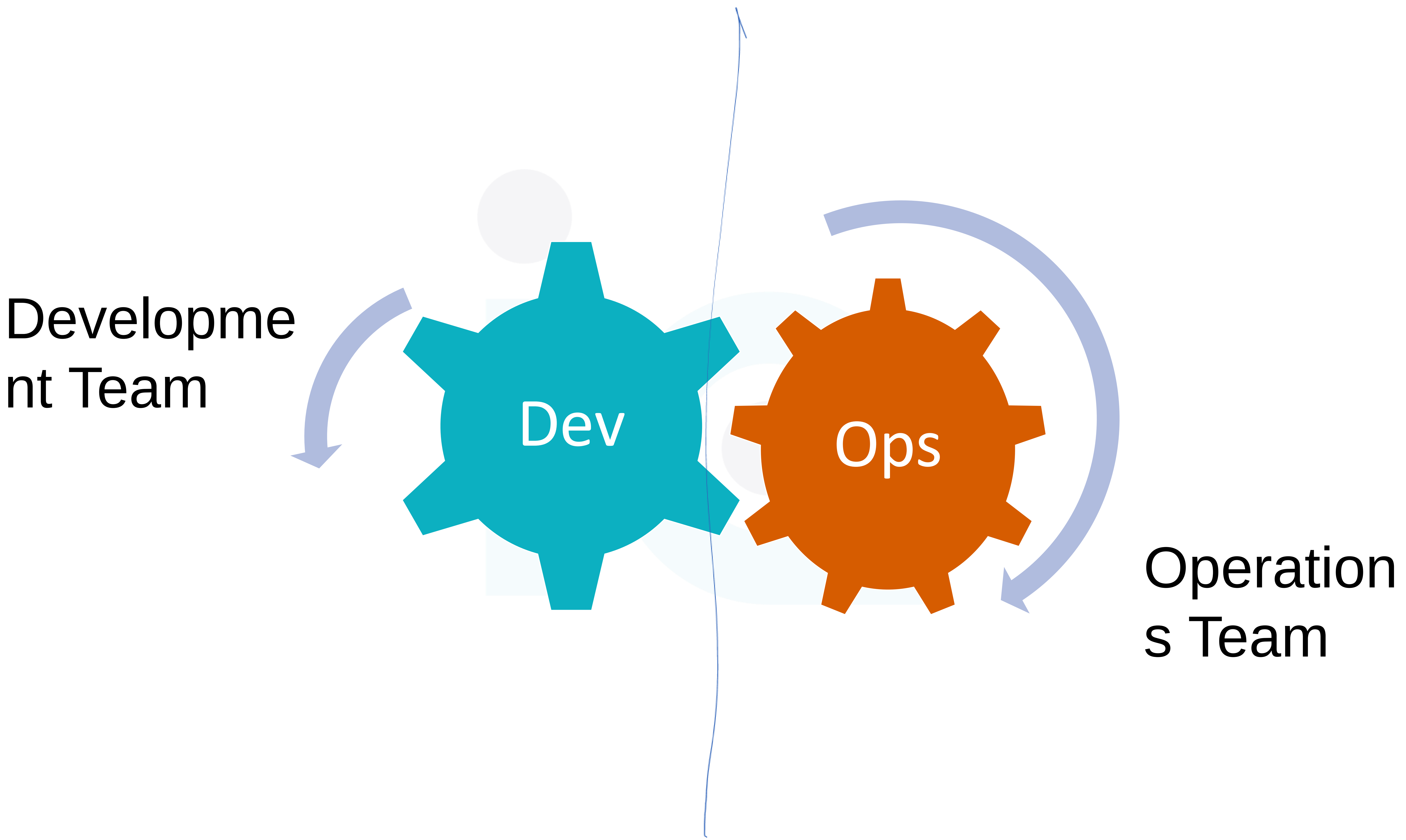
- The process of automating ML systems using DevOps methodologies
- Shares lineage with DevOps, which demands automation

Elements of MLOps

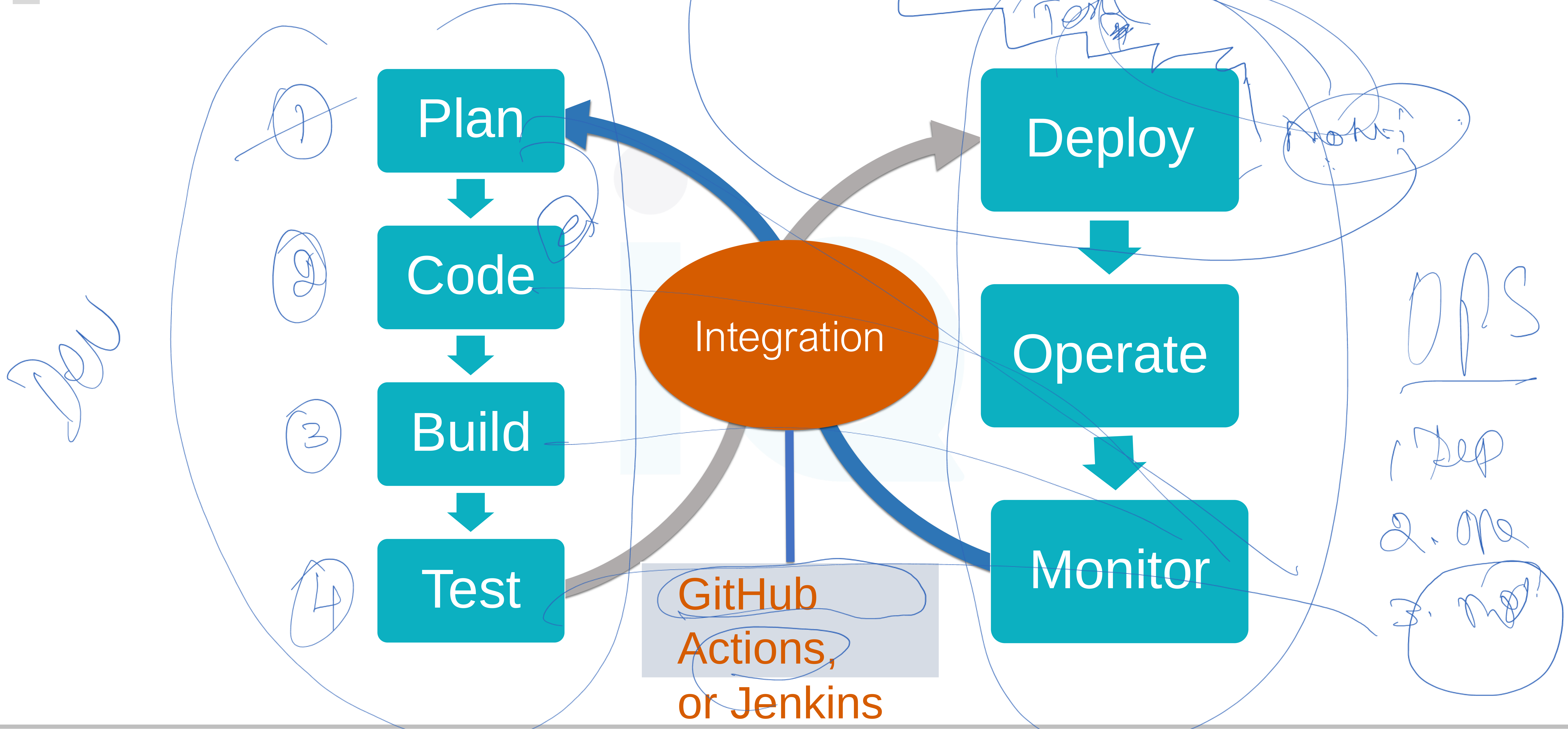


Source: Google Cloud

DevOps Methodology



DevOps Methodology



Stages in DevOps

yaml

Version Control
- source code management,
- GitHub

Continuous Integration
- compile, validate, code review, integration

Continuous Integration
- transform the application to the test servers, performing unit test

Continuous Deployment
- Deploy the tested package on the production server for release

Plan

monitor

MLOps versus DevOps

VC

- With MLOps, not only the software engineering process needs full automation, but so do the data and ML models
- Continuous Training
  - Model training and deployment is added to DevOps
- Monitor new things that break automation
  - Data-drift – the delta changes between the data from last time the model training occurred.

Today

X_i	Y_i
⋮	⋮
x_{100}	y_{100}
x_{500}	y_{500}

Future

$$y_{ML}(x) = w_{opt}\, x + b_{opt}$$

$$\approx w_{opt,1}\, x + b_{opt,1}$$

ML Artefacts

VC

MLOps differs from DevOps!

Team Skill

Data scientists: Focusing EDA, model development, & experimentatio, no necessarily software engineers

Development

Experiment different features, algorithms, models, hyperparameter selection, reproducibility

Testing

Not software but ML system. In addition to unit and integration tests, data validation, trained model quality, ad model validation

Deployment

Involves multi-step pipeline to automate retraining and deploy model

Production

model decays in many ways than conventional software

MLOps differ from software systems!

- CI is not only just testing and validation of software, but also data, data schemas and models
- CD is not only for single software or service but a system including ML deploying another system
- CM/CT is a new methodology, unique to MLOps, that retrains the ML model

Eight Steps in AI/ML system Development

Refer module 5

1. Data Extraction
    - Select and integrate relevant data from various data source
2. Data Analysis
    - Perform Exploratory data analysis (EDA) t understand the data
3. Data Preparation
    - Prepare data (cleaning, split into training, validation & testing) for ML task
4. ML Model training
    - Train appropriate ML model, tune hyperparameters

DataOps for ML System

Overview

- Principles, best practices, and tools for scalable DataOps pipelines in ML.
- **Importance:** Handling large data volumes, real-time processing, resource optimization, system reliability.

- **Data Ingestion:** Tools - Apache Kafka, Apache NiFi, Amazon Kinesis.
- **Data Processing:** ETL vs. ELT, Tools - Apache Spark, Dask.
- **Data Storage:** Solutions - HDFS, S3, Google Cloud Storage, Delta Lake.
- **Data Quality:** Tools - Great Expectations, Apache Griffin.

- **Architecture:** Decoupling, modularity, fault tolerance.
- **Orchestration:** Tools - Apache Airflow, Prefect, Apache NiFi.
- **Monitoring:** Tools - Prometheus, Grafana, ELK stack.
- **Case Study:** Real-world scalable DataOps pipeline example.

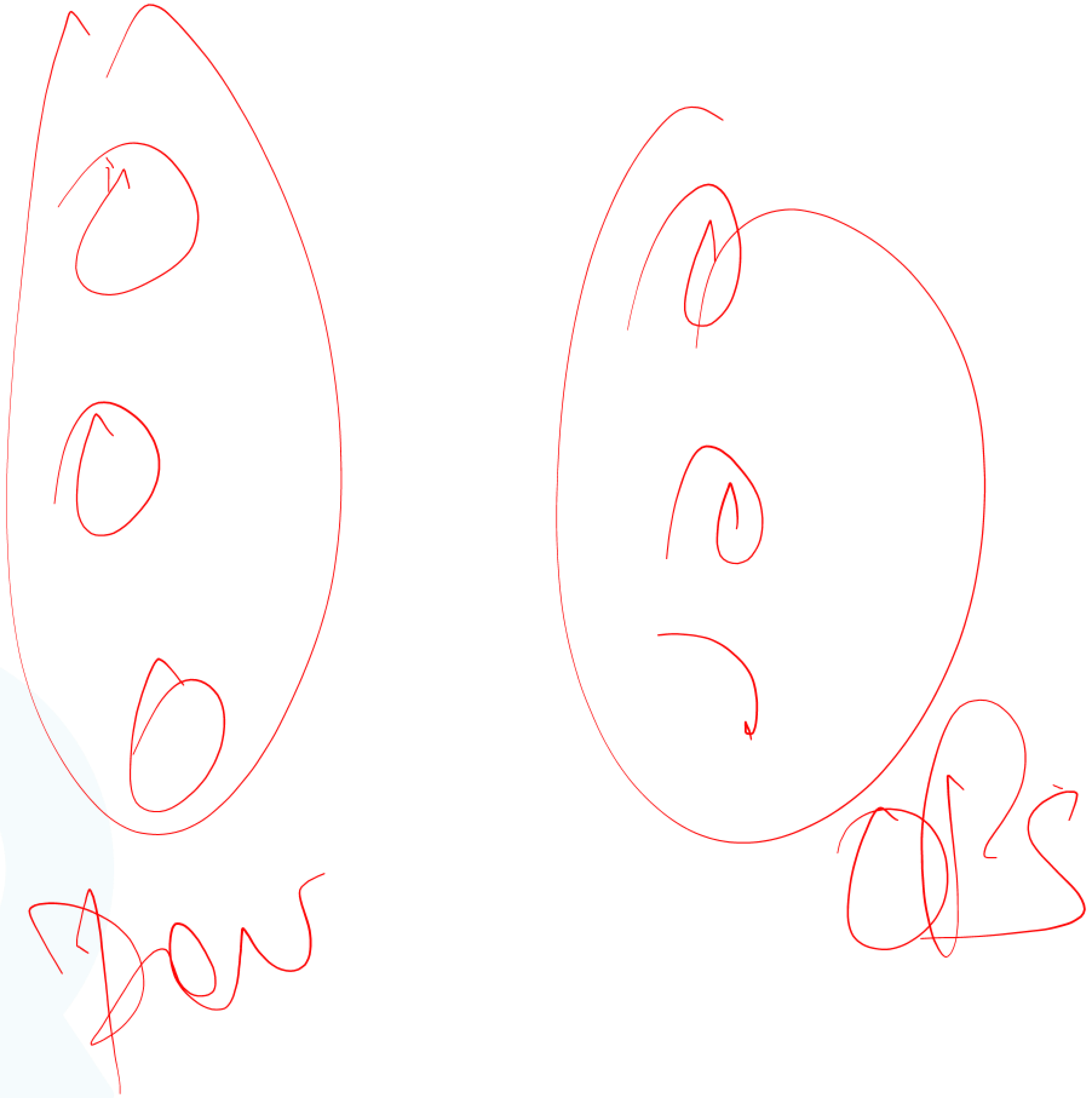
Eight Steps in AI/ML system Development

5. Model evaluation
    - Evaluate model on “Holdout test set” and assess the model
6. Model validation
    - Is the trained model adequate for deployment?
    - Predictive performance is better than a baseline?
7. AI system serving
    - Deployment to a target environment (Microservices, Edge, etc)
8. AI system monitoring
    - Monitor for new iteration/retraining

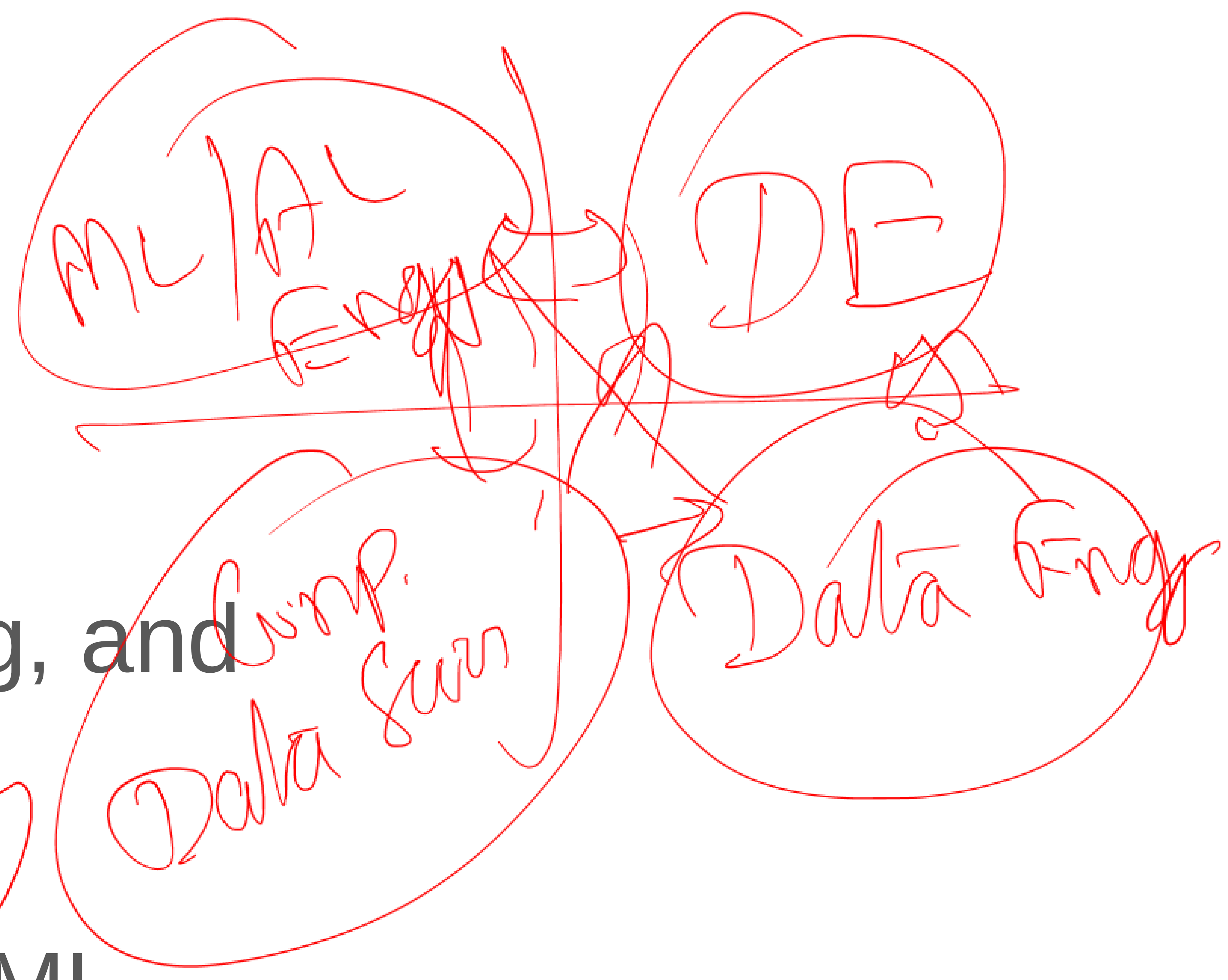
MLOps: How to start?

1
- Level 0: Manual Process
- Level 1: ML Pipeline automation
- Level 2: CI, CD and CT Pipeline automation

2

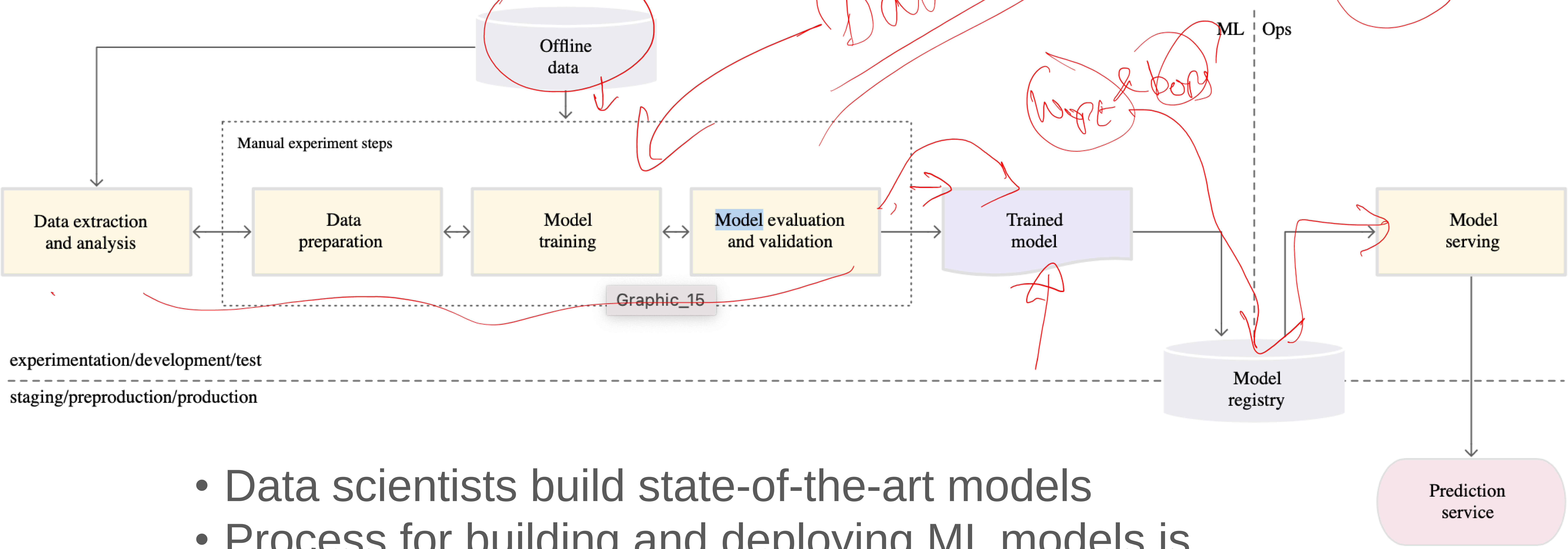


MLOps Level 0: Manual Process



- Manual, script-driven and iterative process
  - Every step is manual:
    - Data analysis, data preparation, model training, and validation
- Disconnect between ML and operations
  - Separates Data Engineers, Data scientists and ML Engineers
- Infrequent release or update or retraining
- No CI
- No CD
- Prediction service, not the full AI System, is referred to as deployment
- Lack of active performance monitoring

MLOps Level 0: Manual Process



- Data scientists build state-of-the-art models
- Process for building and deploying ML models is manual

Source : Google

MLOps Level 0: Manual Process

When to use?

- Common in many businesses
- Sufficient when the models change seldom

Recommendations

- Actively monitoring the quality of the model
- Frequently retraining the model, if needed
- Continuously experiment with new implementations
- Automate the process
  - Start implementing CI, CD, and CM/CT

Data → DVC

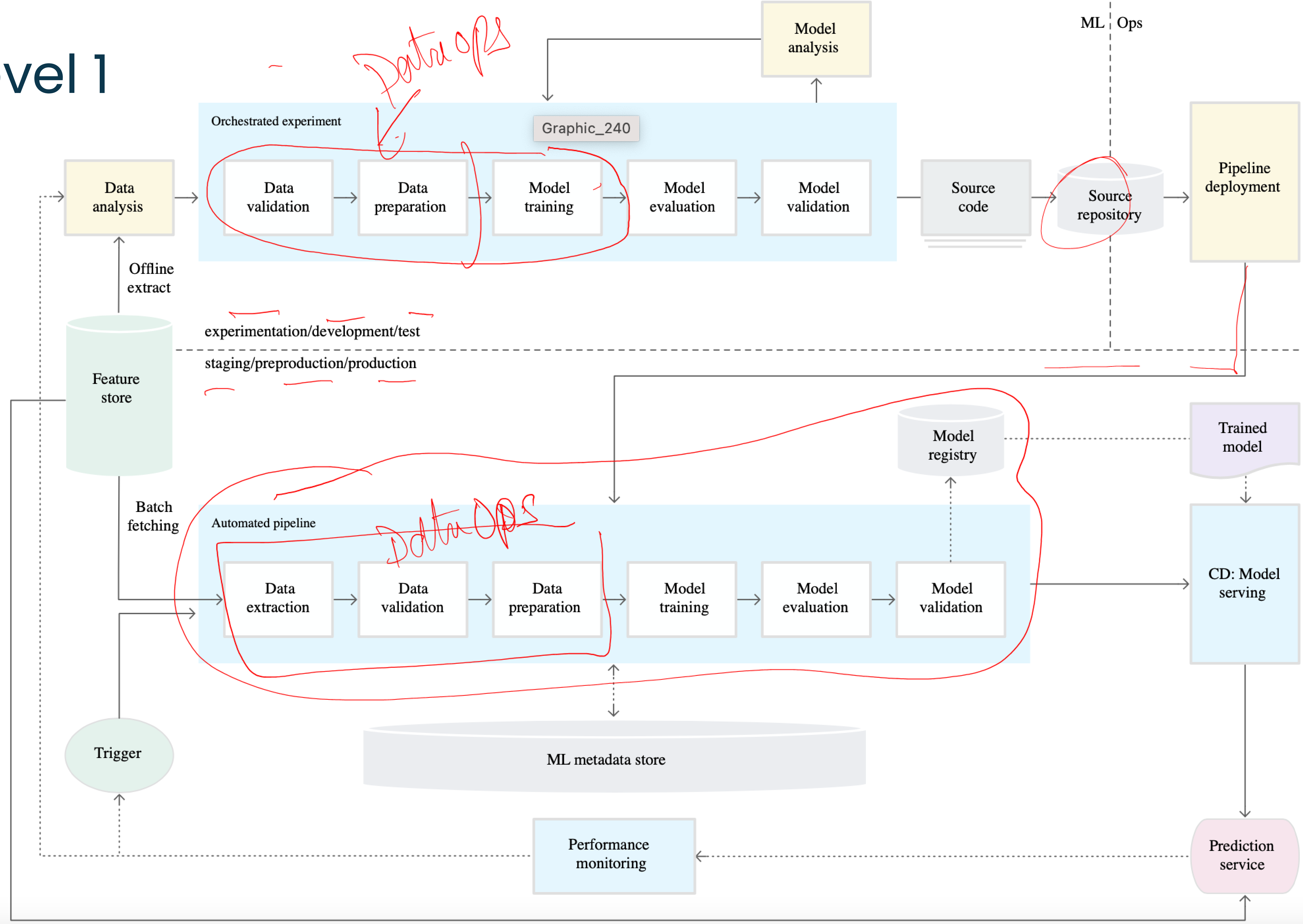
weights | bias

ClearML → Git, DVC, own, ML

1. IDE [VS Code]
2. GitHub | Gitlab
3. learn to write basic yaml file
4. Start a new repo & Push your format file & try to run unit test. CI
5. Open a new A/c in AWS / DVC / Cleanml
6.

MLOps Level 1

Level 1: ML pipeline automation



MLOps Level 1: ML pipeline automation

- Perform continuous training of the model by automating the ML pipeline
- Retraining
  - Introduce automated data and model validation
  - Implement pipeline triggers and metadata management
- Rapid experiments
  - ML experiment is orchestrated
  - Better readiness to move to the whole pipeline to production
- CM/CT automation
  - Model in production is automatically retrained
  - Development and experiment environment is used in preproduction and production environment

MLOps Level 1: ML pipeline automation

- CD of models
  - New models that are trained on new data are available
- Pipeline deployment
  - Unlike in Level 0, the whole training pipeline is deployed

MLOps Level 1: Data and Model Validation

- Data Validation
  - Needed to decide whether to retrain the model or stop the execution of the pipeline
  - Data schema skews
    - Due to anomalies in the data
    - Data for downstream pipeline (data processing and model training) does not comply with expected schema
    - Stop the execution of pipeline, and let data scientist team to investigate the data
  - Data Value Skew: Significant changes in the statistical properties of data

MLOps Level 1

Data and Model Validation

- Model Validation
  - Evaluate the retrained model before promoting to production
  - Produce evaluation metric using trained model on a test dataset
  - Compare the predictive metric of retrained model with current model (production model, baseline model and business-requirement model)
  - Check the metric with various segment of the data
  - Test deployment-readiness, incl. infrastructure compatibility

MLOps Level 1

Feature Store

- A centralized repository for standardized definitions, storage and access of features for training and serving

Metadata management

- Store information about each execution
  - The pipeline and component versions used
  - Start and end time, date
  - Execution time of each step in the pipeline
  - Executor of pipeline
  - Parameter arguments that were passed to the pipeline, etc

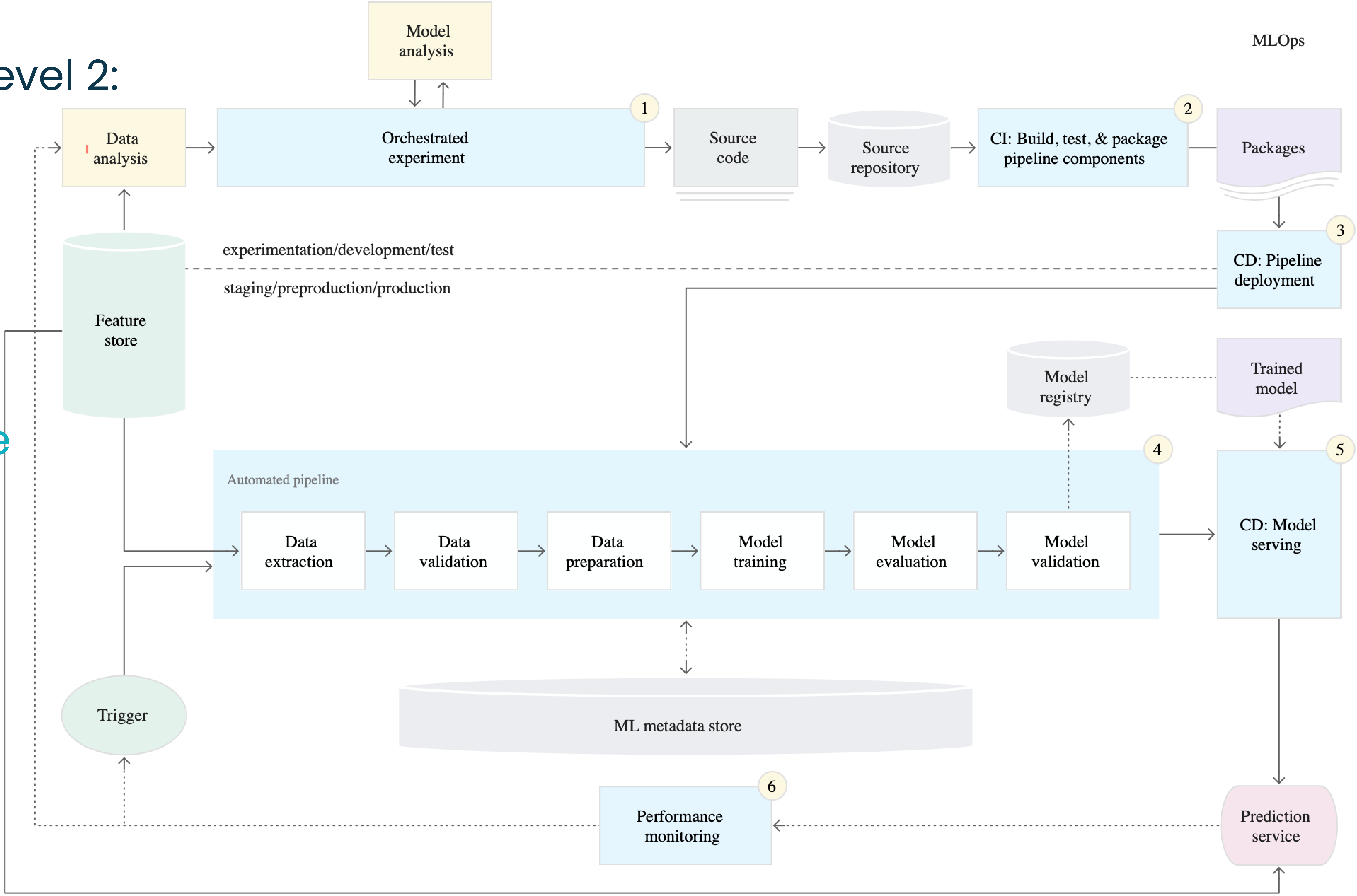
MLOps Level 1:

ML pipeline triggers

- On demand: Ad-hoc manual execution of the pipeline
- On a schedule: Daily, weekly or monthly basis whenever new, labelled data available
- On availability of new data
- On model performance degradation
- On significant changes in the data distributions

MLOps Level 2:

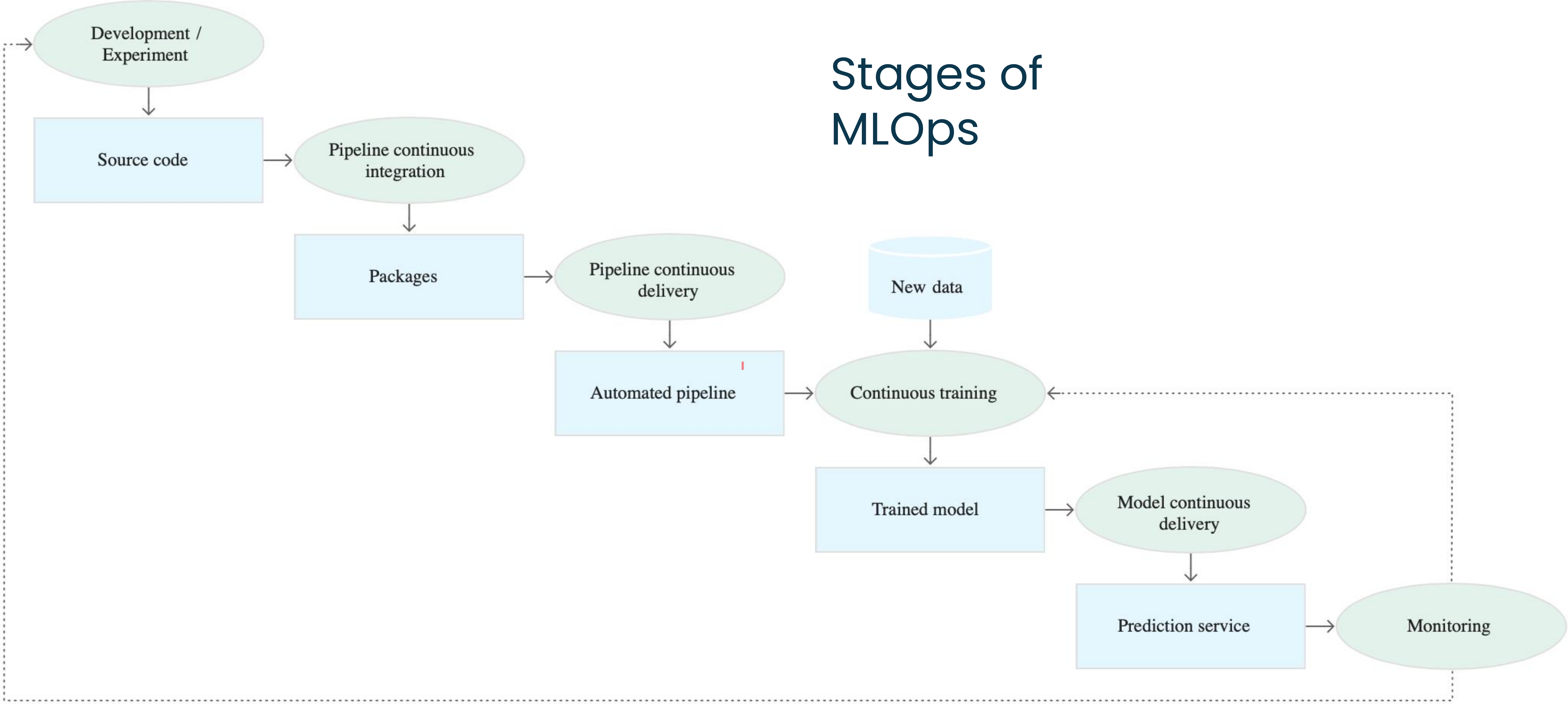
Level 2: CI/CD pipeline automation



MLOps Level 2: CI/CD pipeline automation

- Allows to rapidly explore new ideas around feature engineering, model architecture and hyperparameters
- Automatically build, test and deploy new pipeline components to the target environments
- Steps include
  - Source control
  - Test and build services
  - Deployment services
  - Model registry
  - Feature store
  - ML metadata store
  - ML pipeline orchestrator

Stages of MLOps



MLOps Level 2: CI/CD pipeline automation

- Development and experimentation:
  - iteratively try new ML models where the experiment steps are orchestrated
  - source code is then pushed to a source repository
- Pipeline continuous integration:
  - run various tests
  - pipeline components (packages, executables, and artifacts) are deployed in a later stage
- Pipeline continuous delivery:
  - deploy the artifacts produced by the CI stage to the target environment

MLOps Level 2: CI/CD pipeline automation

- Automated triggering:
  - Pipeline is automatically executed in production based on a schedule or in response to a trigger
  - Trained model is pushed to the model registry
- Model continuous delivery:
  - Serve the trained model as a prediction service for the predictions
  - The output of this stage is a deployed model prediction service
- Monitoring:
  - Collect statistics on the model performance based on live data.
  - Output of this stage is a trigger to execute the pipeline or to execute a new experiment cycle

Summary

- Implementation and Automation
  - Continuous Integration (CI)
  - Continuous Delivery (CD)
  - Continuous Monitoring/Training (CM/CT)
  - MLOps Levels
    - Level 0: Manual Process
    - Level 1: ML Pipeline automation
    - Level 2: CI, CD and CT Pipeline automation

Starting a new MLOps Project

- https://drivendata.github.io/cookiecutter-data-science/

Building a MLOps Project: Phase 1: Planning & Design

Define Objectives & Scope

- Clearly articulate project goals, target metrics (e.g., accuracy, latency, throughput), and desired business outcomes
- Identify stakeholders and their roles
- Create a project timeline with milestones

Data Strategy

- **Data Sources:** Determine data sources (internal, external, real-time, batch)
- **Data Quality:** Define data quality standards and establish cleansing/validation processes
- **Data Labeling:** Plan for data labeling if needed (in-house, outsourced, crowd-sourced)
- **Data Governance:** Establish policies for data access, security, and

Model Selection & Architecture Design

- **Problem Type:** Classify the ML problem (regression, classification, clustering, etc.)
- **Algorithm Research:** Explore potential algorithms and architectures based on the problem type and data characteristics
- **Baseline Model:** Create a simple model to establish a performance baseline
- **Model Architecture:** Design a more complex model architecture (if necessary)

Building a MLOps Project: Phase 2: Development & Experimentation

DataOps Implementation

- **Data Pipelines:** Create automated pipelines for data ingestion, cleaning, transformation, and feature engineering
- **Data Versioning:** Use tools like DVC or Pachyderm to track changes to data and pipelines
- **Feature Store: (Optional)** Implement a feature store like Feast or Tecton to centralize and manage features
- **Data Exploration and Visualization:** Explore the data thoroughly to

ModelOps Implementation

- **Experiment Tracking:** Log experiments with MLflow, TensorBoard, or Weights & Biases to track model hyperparameters, metrics, and artifacts
- **Model Versioning:** Use tools like MLflow to version models and track performance over time
- **Hyperparameter Tuning:** Perform systematic tuning of model hype[illegible]o optim[illegible]nce

Version Control and Collaboration

- **Git: Use Git for code versioning and collaboration**
- **Code Reviews: Implement code reviews for quality assurance**
- **Branching Strategy:** Adopt a branching strategy (e.g., Gitflow) to manage code changes effectively
- **Data and Model Testing:** In MLOps, it's crucial to test not only your code but also your data and model performance

Building a MLOps Project: Phase 3: Deployment and Monitoring

Containerization

- **Docker:** Containerize the model and its dependencies using Docker
- **Docker Registries:** Store container images in a registry like Docker Hub or a private registry
- Kubernetes (Module 7):
  - Cluster Setup: Set up a Kubernetes cluster (on-premises or in the cloud).
  - Deployment Manifests: Create Kubernetes manifests (YAML files) to define Pods, Services, Deployments, and other resources for your ML application.
  - Scalability: Utilize Kubernetes' autoscaling capabilities to adjust resources based on demand.
  - Rollouts and Rollbacks: Use Kubernetes to manage gradual rollouts and rollbacks of model versions.
- Tools: Consider Kubeflow, which provides a platform for building

CI/CD Pipeline

- **Automated Testing:** Write unit, integration, and regression tests
- **CI/CD Infrastructure**: Set up infrastructure for automated testing and deployment
- **CI/CD Tools:** Use tools like GitHub Actions, GitLab CI, Jenkins, or CircleCI

Building a MLOps Project: Phase 3: Deployment and Monitoring

Deployment

- **Cloud Deployment:** Deploy to cloud platforms like AWS SageMaker, Azure ML, or Google Cloud AI Platform
- **Model Serving:** Use tools like KFServing, Seldon Core, or BentoML for model serving
- **Web Applications:** Create web applications (e.g., with Streamlit, Gradio, Flask) for model interaction
- **Edge Deployment:** (Optional) Deploy models to edge devices if applicable

Continuous Monitoring and Maintenance

- **Logging:** Implement logging of model inputs, outputs, and performance metrics
- **Monitoring Tools:** Use tools like Prometheus, Grafana, or cloud-specific monitoring solutions
- **Alerting:** Set up alerts for performance degradation or anomalies
- **Model Retraining:** Establish a process for retraining models with fresh data

Building a MLOps Project: Phase 4: Optimization and Governance

Model Optimization

- **Performance Optimization:** Profile the model to identify bottlenecks and optimize for speed and resource usage
- **Model Compression:** Apply techniques like quantization or pruning to reduce model size
- **Batch Prediction:** Use batch prediction for more efficient

Security and Governance

- **Model Security:** Implement security measures to protect the model from unauthorized access and adversarial attacks
- **Model Bias and Fairness:** Regularly audit the model for bias and take corrective measures
- **Model Explainability:** Use tools like SHAP or LIME to explain model

Documentation

- Create thorough documentation for the project, including data descriptions, model architectures, deployment instructions, and monitoring procedures

Built with Sphinx

Feedback & Iteration

- **Collect Feedback:** Gather feedback from users and stakeholders
- **Analyze Performance:** Continuously analyze model performance metrics and user feedback
- **Iterate:** Refine the model, data pipelines, and deployment process based on feedback **and analysis**

“Docker

(Hands-on session in the afternoon)

Docker

- What is Docker?
  - A platform for developing, shipping, and running applications.
  - Utilizes containerization to make applications portable and consistent across different environments.
  - Containers encapsulate an application with all of its dependencies.

Docker

- Relevance in Modern Software Development
  - Facilitates continuous integration and continuous deployment (CI/CD) by ensuring that software runs the same in all environments.
  - Reduces "it works on my machine" problems by providing a consistent environment from development to production.
  - Enables microservices architecture by allowing each service to be containerized and scaled independently.
  - Streamlines development by allowing developers to create predictable and efficient work environments.
  - Enhances collaboration between development and operations teams for faster and more reliable software delivery.

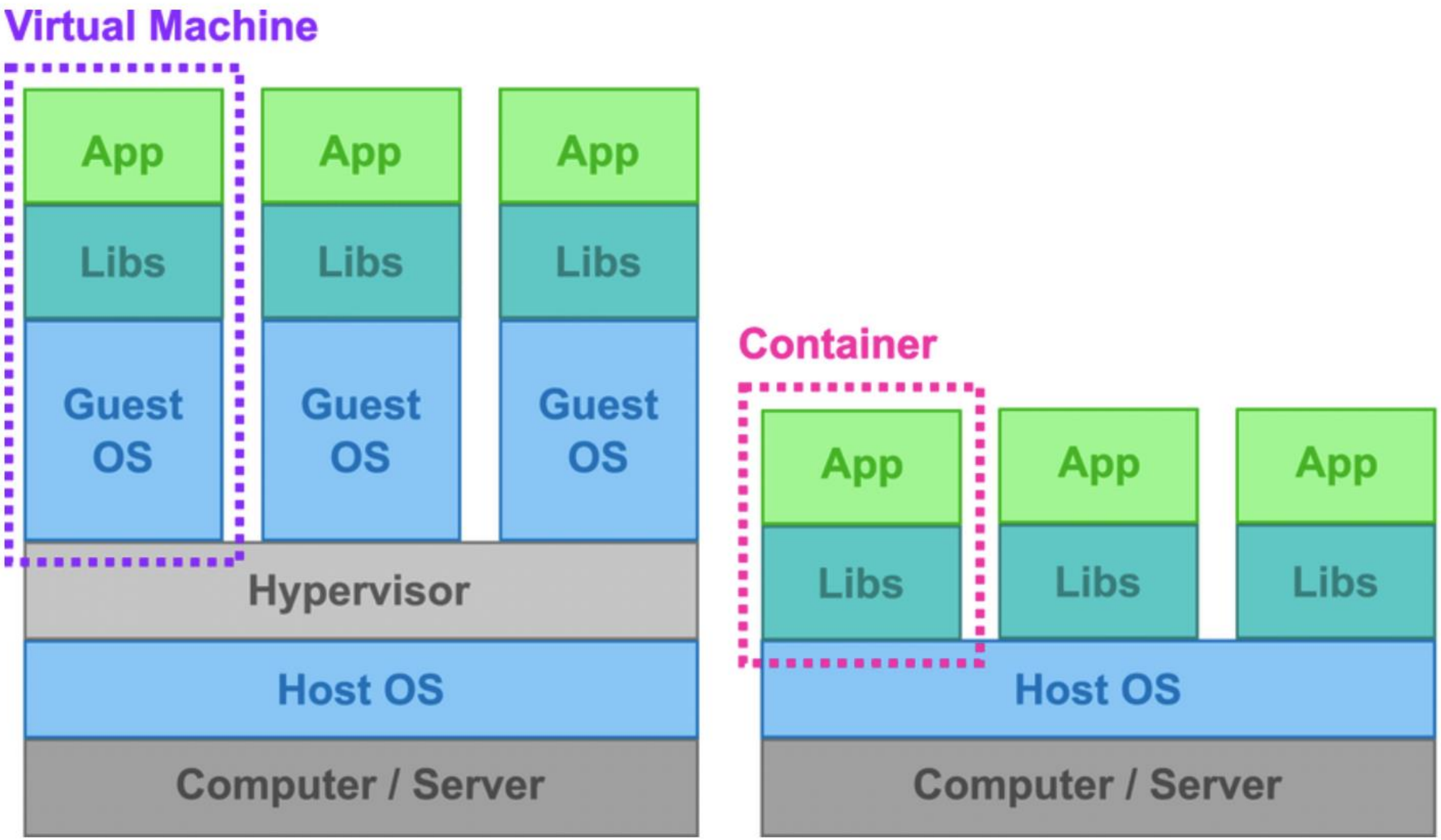
Docker: Containers vs. Virtual Machines

- Containers
  - Lightweight: Containers are much smaller and more resource-efficient than VMs because they share the underlying host operating system (OS) kernel.
  - Portable: Containers package applications with all their dependencies, ensuring consistent behavior across different environments.
  - Fast Startup: Since containers don't need to boot up a full OS, they start up almost instantly.
  - Efficient Resource Usage: Containers share the host OS resources, reducing overhead and allowing more applications to run on the same hardware.
  - Suitable for Microservices: Containers are well-suited for breaking down applications into smaller, independent components (microservices), which can be developed, deployed, and scaled independently.

Docker: Containers vs. Virtual Machines

- Virtual Machines
  - Full OS: Each VM runs its own complete guest operating system, providing strong isolation between VMs.
  - More Overhead: VMs require more resources than containers due to the need to run a full OS for each VM.
  - Slower Startup: Booting up a VM takes longer than starting a container.
  - Stronger Isolation: VMs provide better security and fault isolation than containers due to the separation provided by the hypervisor.
  - Suitable for Legacy Applications: VMs are often used to run legacy applications that may not be easily containerized.

Docker: Containers vs. Virtual Machines



Docker: Core Concepts

- Images and Containers
  - Docker images are lightweight, stand-alone, executable software packages that include everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and config files.
  - Containers are a runtime instance of Docker images – an image becomes a container when it runs on Docker Engine.
  - Containers are isolated from each other and the host system, but can communicate through well-defined channels.

Docker: Core Concepts

- Docker Hub
  - Docker Hub is a cloud-based registry service that allows you to link to code repositories, build your images, test them, store manually pushed images, and link to Docker Cloud.
  - Provides a comprehensive repository for Docker container images with both public and private storage options.
  - It is the default registry where Docker looks for images.
  - Offers automated build capabilities for creating Docker images from online source code repositories.

Docker

- Docker's Benefits
  - Promotes consistency across development, staging, and production environments.
  - Ensures that applications run the same, regardless of where they are deployed.
  - Reduces overhead and increases efficiency compared to traditional virtual machines.

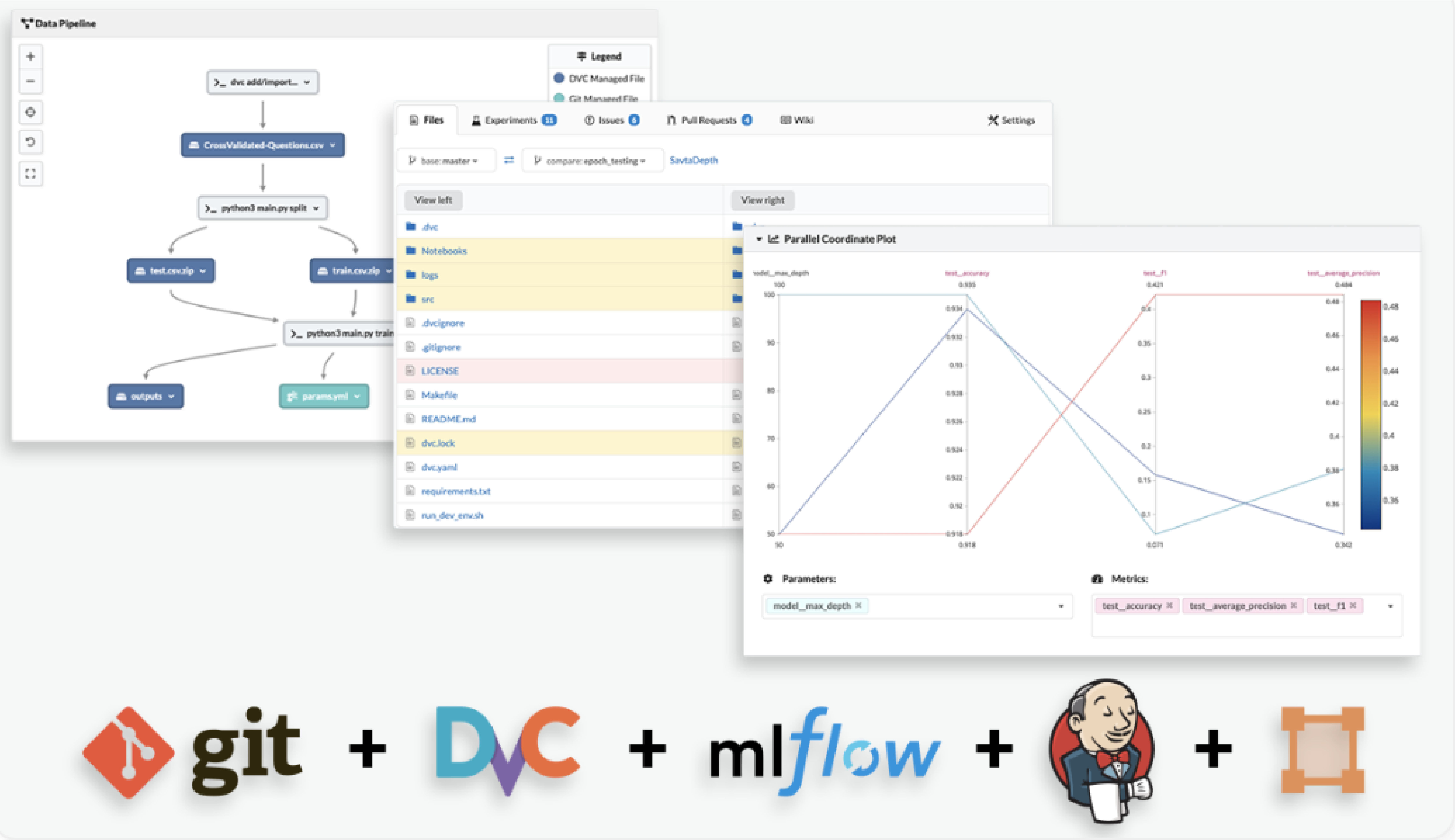
Docker

- Docker in the Development Toolchain
  - Integrates with GitHub for version control, enabling seamless code updates and tracking.
  - Works alongside GitHub Actions for automated testing and deployment pipelines.
  - Complements AWS Compute services to provide scalable and secure hosting for containerized applications.
  - Facilitates data version control with DVC, making it easier to track and manage datasets and machine learning models.
  - Enhances ML operations by working with MLflow to manage the lifecycle of machine learning models, including experimentation, reproducibility, and deployment.

Docker

- Synergy Between Technologies
  - Containerization with Docker offers a standardized unit for software development, enabling a smooth workflow across tools like GitHub, AWS, DVC, and MLflow.
  - Simplifies the complexity of managing dependencies and environments in machine learning projects.
  - Empowers teams to build, test, and release software faster and more reliably, contributing to the DevOps culture of collaboration and efficiency.

Docker



Data Pipeline
Legend
DVC Managed File
Files
Experiments
Issues
Pull Requests
Wiki
Settings
View left
View right
Parallel Coordinate Plot
Parameters:
Metrics:
git
+
DVC
+
mlflow
+
+