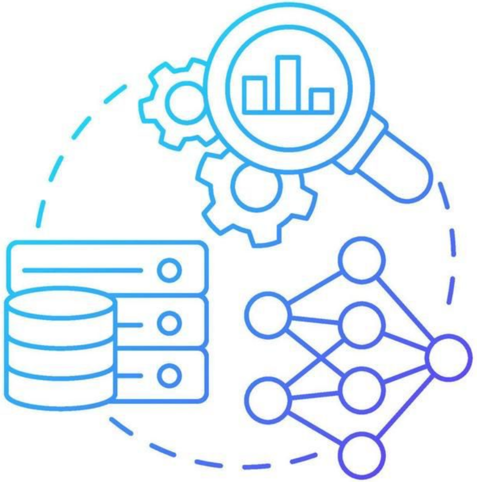
TinyOps: MLOps for TinyML

Dr Sudeepta Mishra

What is important in ML Applications?

What is important in ML Applications?

The Model



What is important in ML Applications?

A model alone is a part of the bigger picture.

Orchestrating the entire flow

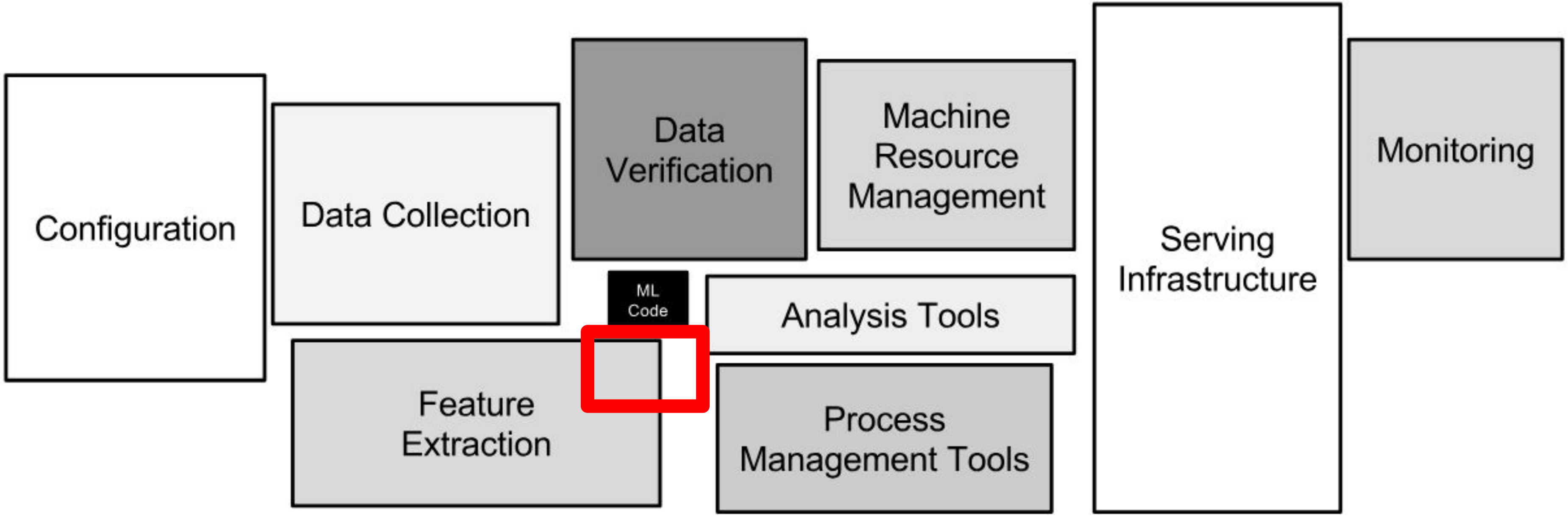
Neural Network Computation + Non Neural Network Computation

Managing Model

Managing Data

Monitoring etc.

Infrastructure Surrounding ML Systems



ImageSource: **Hidden technical debt in Machine learning systems.** In Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15).

Designing an ML system

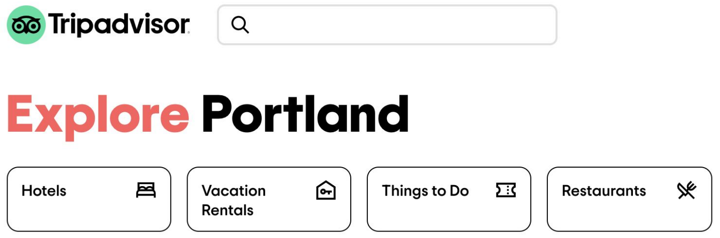
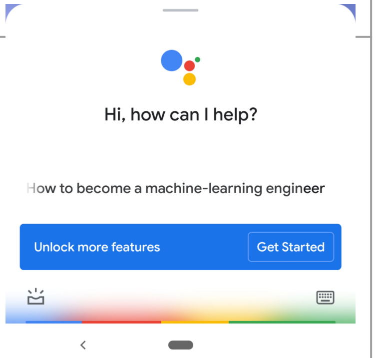
- Reliable
- Scalable
- Maintainable
- Adaptable

Different types of ML systems

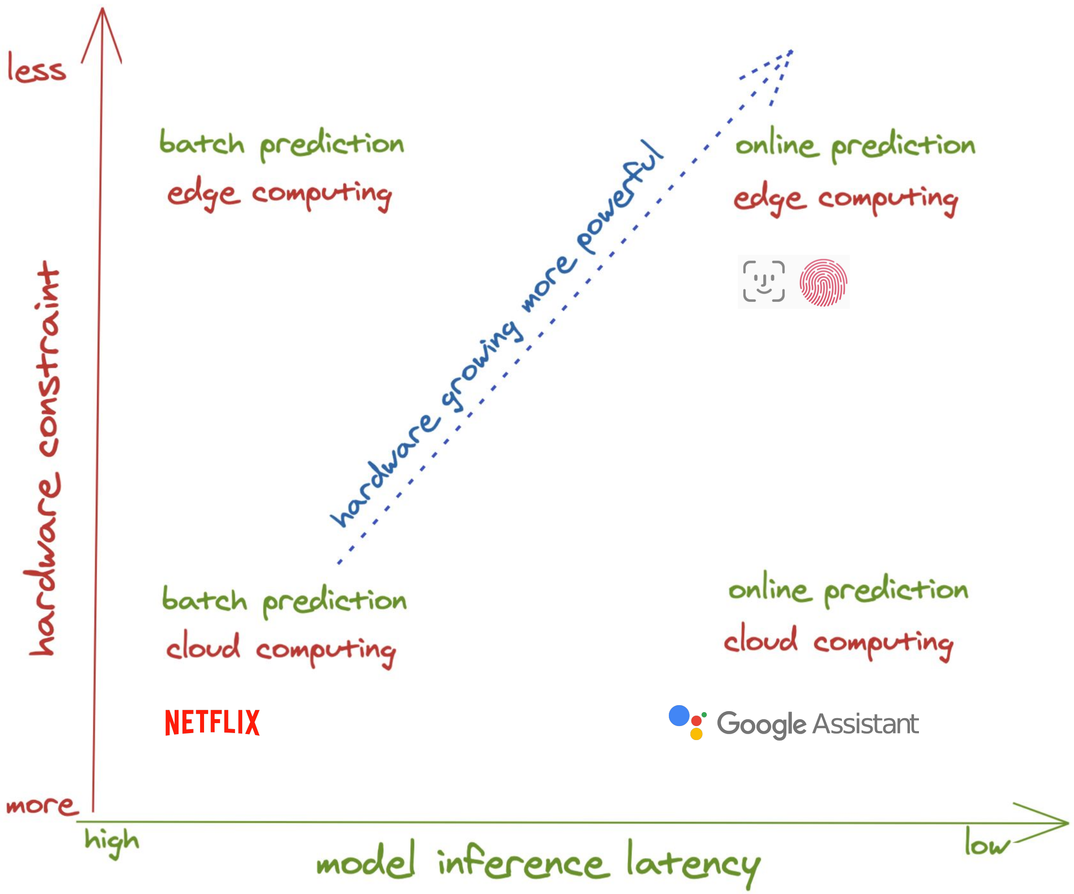
- How their ML models serve their predictions (batch prediction vs. online prediction)
- Where the majority of computation is done (edge computing vs. cloud computing)
- How often their ML models get updated (online learning vs. offline learning)

Different types of ML systems

- How their ML models serve their predictions (batch prediction vs. online prediction)
- Where the majority of computation is done (edge computing vs. cloud computing)
- How often their ML models get updated (online learning vs. offline learning)

	Batch prediction	**Online prediction**
Frequency	Periodical (e.g. every 4 hours)	As soon as requests come
Useful for	Processing accumulated data when you don't need immediate results (e.g. recommendation systems)	When predictions are needed as soon as data sample is generated (e.g. fraud detection)
Optimized	High throughput	Low latency
Input space	Finite: need to know how many predictions to generate	Can be infinite
Examples	• TripAdvisor hotel ranking • Netflix recommendations 	• Google Assistant speech recognition • Twitter feed • Wakeword 

Future of ML: online and on-device



Offline learning vs. $\underbrace{\text{online learning}}_{\text{Harder \& less common}}$

	Offline learning	**Online learning**
Iteration cycle	Periodical (months)	Continual (minutes) $\neq$ continuous
Batch size	batch (thousands -> millions of samples) GPT-3 125M params: batch size 0.5M GPT-3 175B params: batch size 3.2M	microbatch (hundreds of samples)
Data usage	Each sample seen multiple times (epochs)	Each sample seen at most once
Evaluation	Mostly offline evaluation	Offline evaluation as sanity check Mostly relying on online evaluation (A/B testing)
Examples	Most applications	TikTok recommendation system, Twitter hashtag trending

ML in production

ML in production: expectation

ML in production: expectation

Collect data → Train model → Deploy model

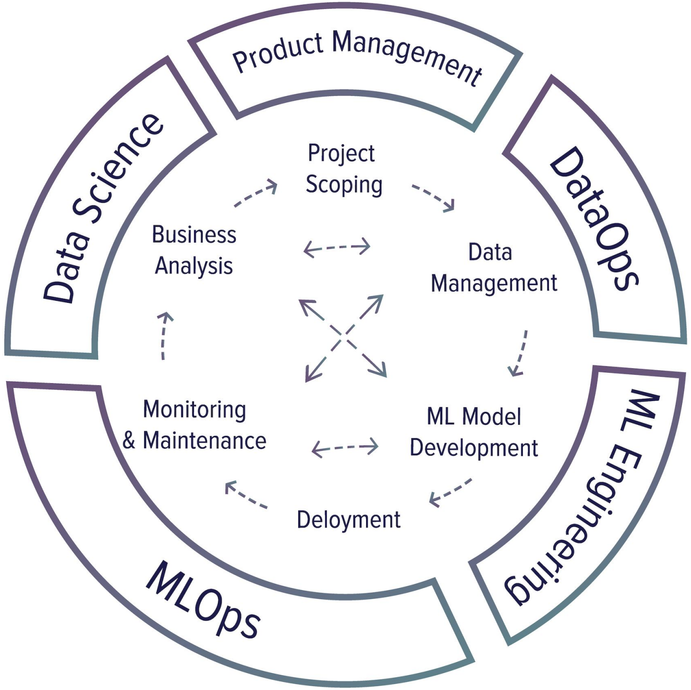
Waterfall model

ML in production: reality

1. Choose a metric to optimize
2. Collect data
3. Train model
4. Realize many labels are wrong -> relabel data
5. Train model
6. Model performs poorly on one class -> collect more data for that class
7. Train model
8. Model performs poorly on most recent data -> collect more recent data
9. Train model
10. Deploy model
11. Dream about $$$
12. Wake up at 2am to complaints that model biases against one group -> revert to older version
13. Get more data, train more, do more testing
14. Deploy model
15. Pray
16. Model performs well but revenue decreasing
17. Cry
18. Choose a different metric
19. Start over

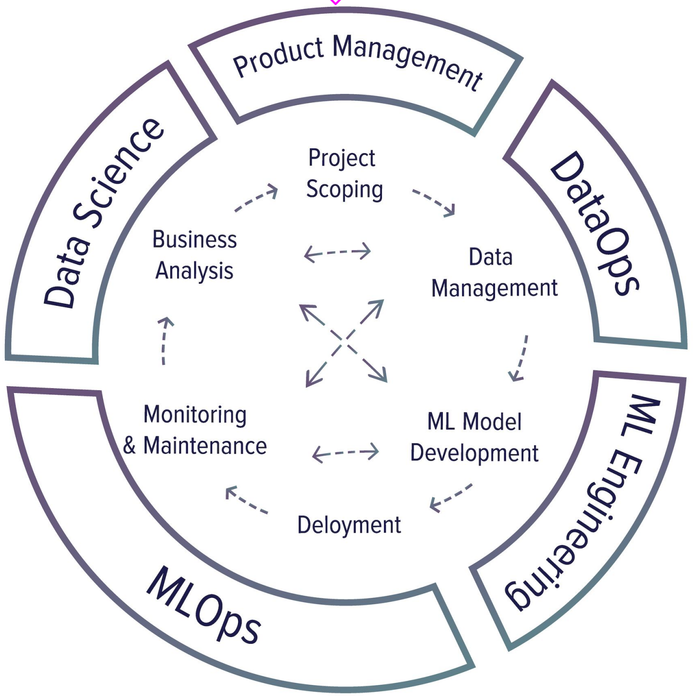
ML Project

Iterative Process



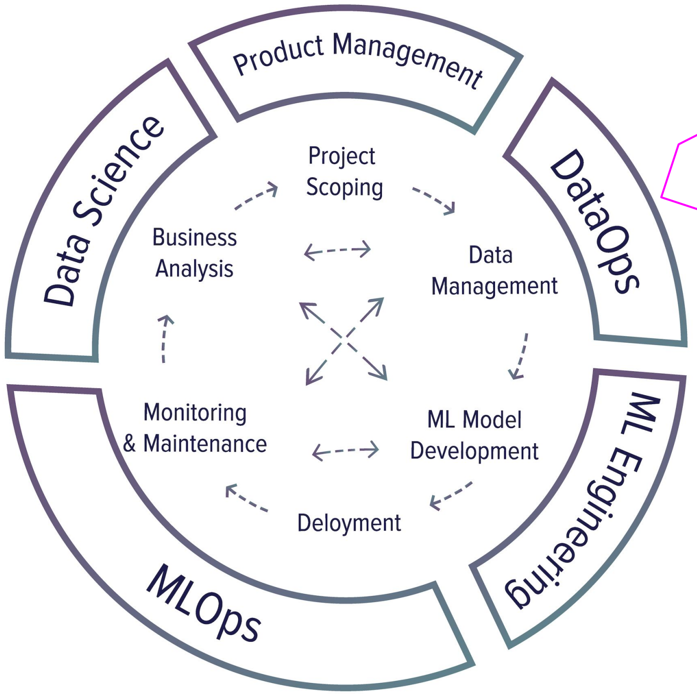
Project scoping

- Goals & objectives
- Constraints
- Evaluation
- Resources estimated and allocated



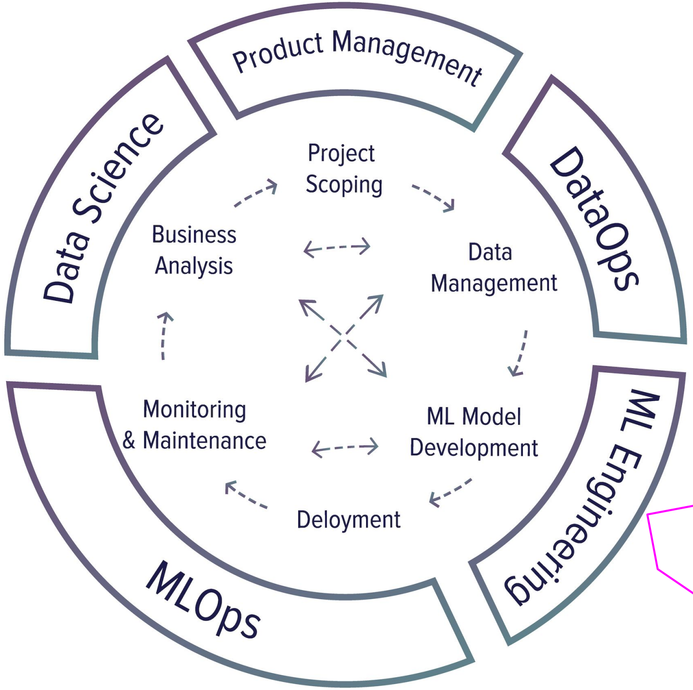
Data management

- Data sources
- Data format
- Processing
- Storage
- Data consumer
- Data controller



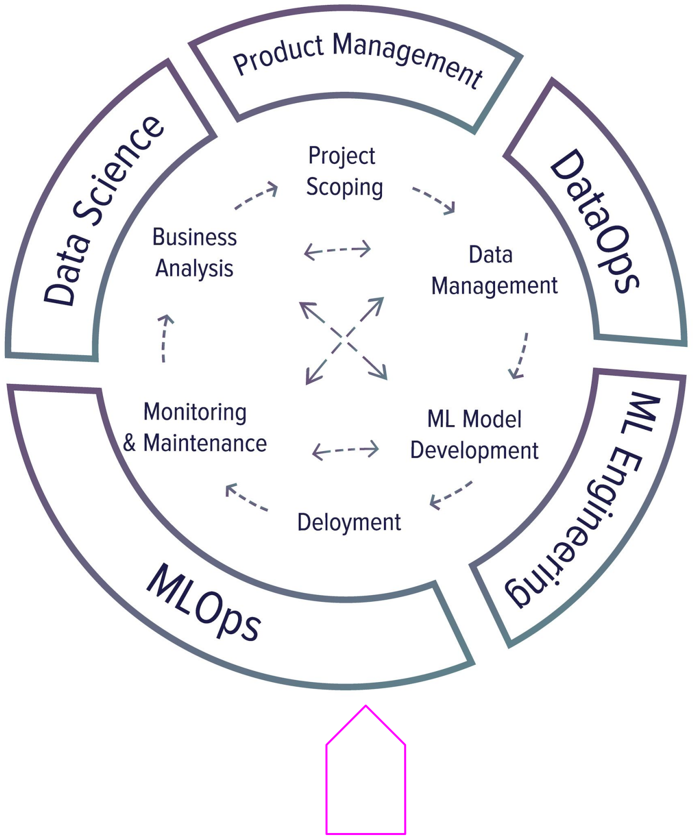
Model development

- Dataset creation
- Feature engineering
- Model training
- Offline model evaluation
- Requires the most ML knowledge



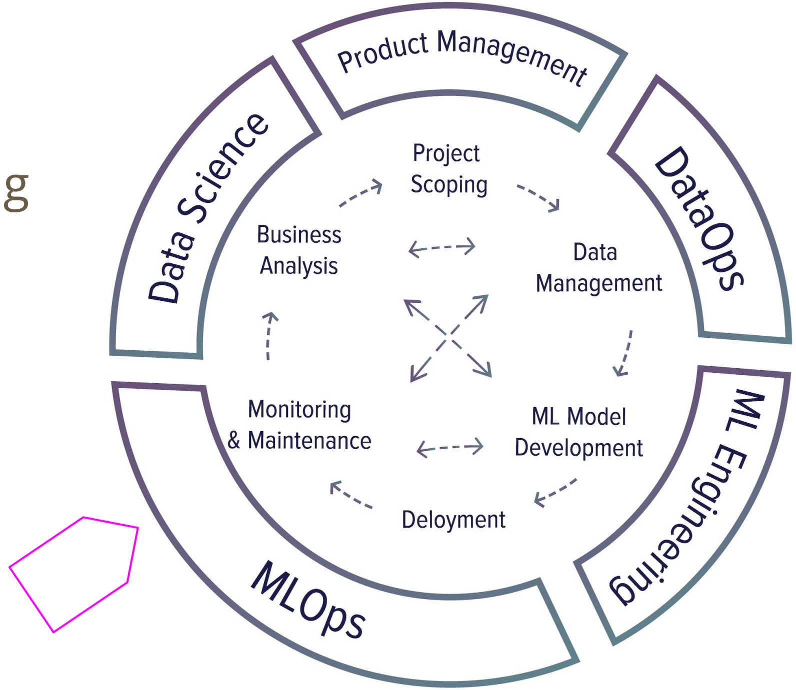
Deployment

- Deploying and serving
- Release strategies
- Online model evaluation
- Accessible to users
- Earn $$$



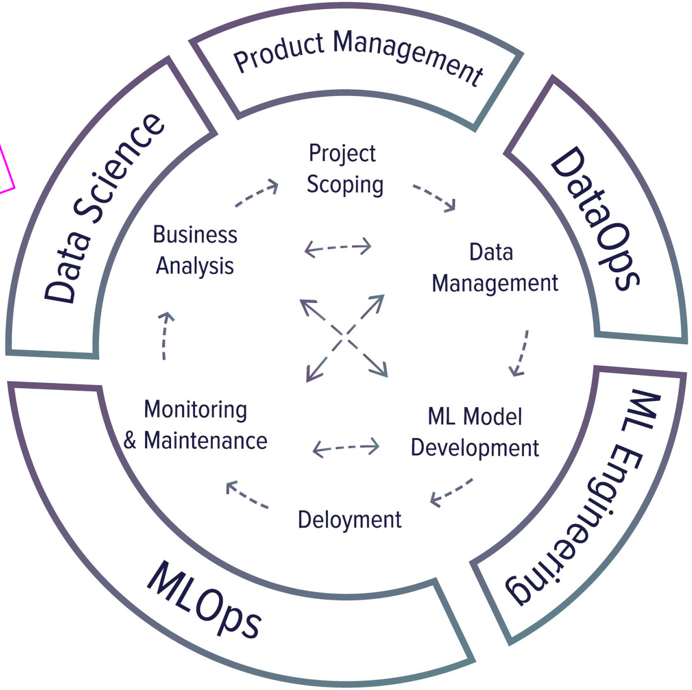
Monitoring & maintenance

- Model performance & data monitoring
- Model retraining
- Model updates



Business analysis

- User experience
- Evaluate model performance against business performance



Your Great Idea: Predictive Maintenance

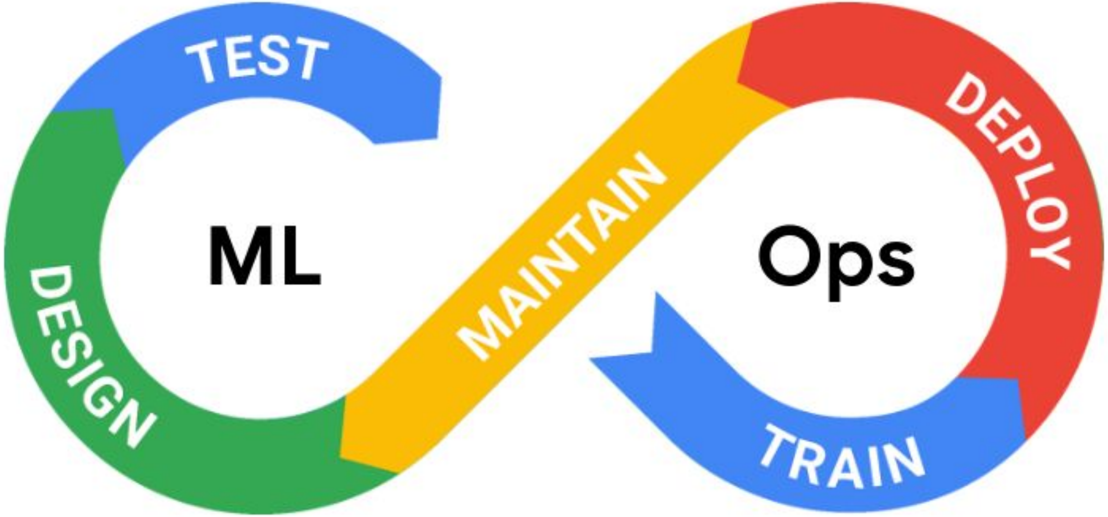
Develop

Data

Train

Manage

Privacy


TEST
DESIGN
ML
MAINTAIN
Ops
DEPLOY
TRAIN

Machine Learning Operations (MLOps)

Practices and tools that streamline, automate, and unify the process of taking machine learning models from development to production, and maintaining them over time.

Purpose:

- Bridges the gap between ML development and operational deployment
- Involves collaboration between data scientists, ML engineers, DevOps, and IT
- Ensures models are reliable, scalable, and maintainable in real-world environments

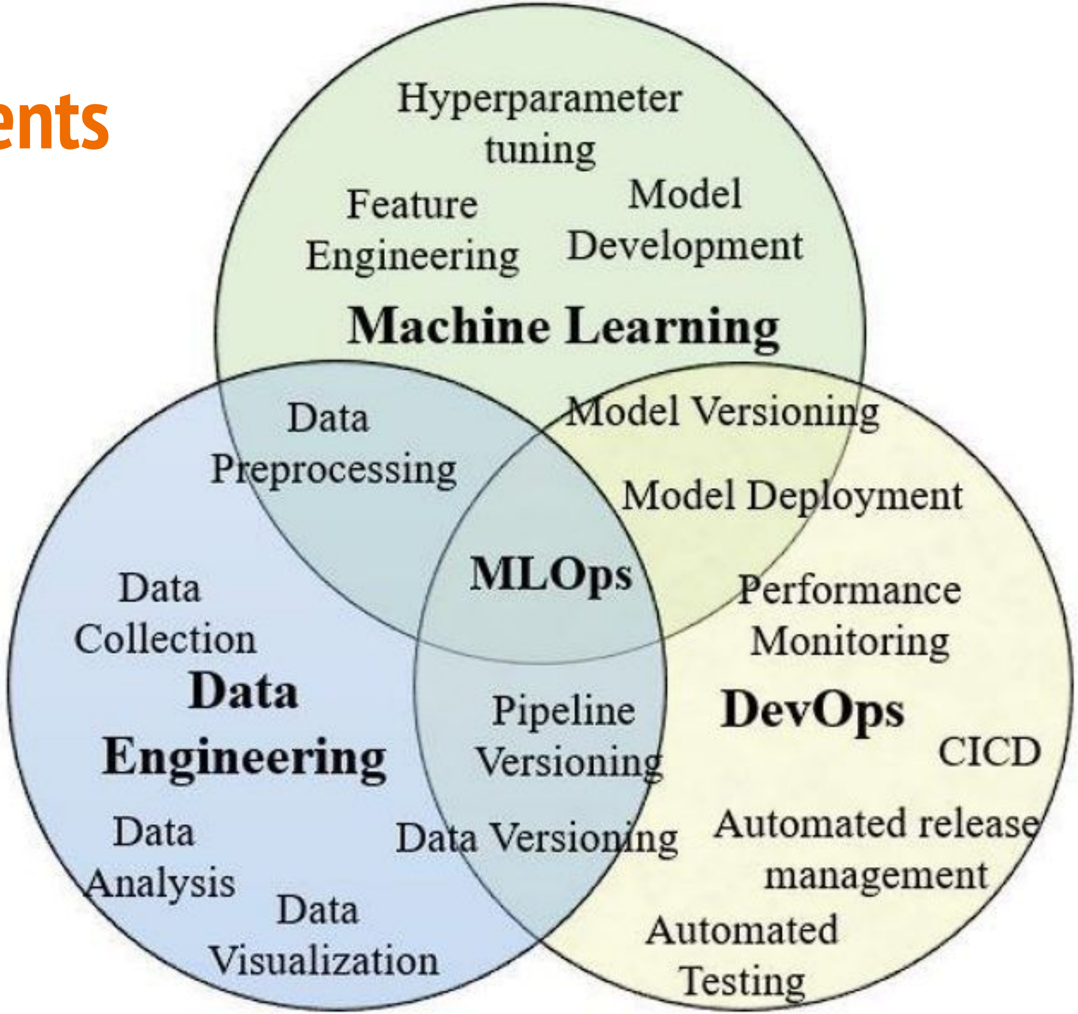
Key Benefits

- Faster and more reliable model deployment
- Automated workflows (CI/CD for ML)
- Improved monitoring, validation, and governance
- Reduces manual errors and technical debt

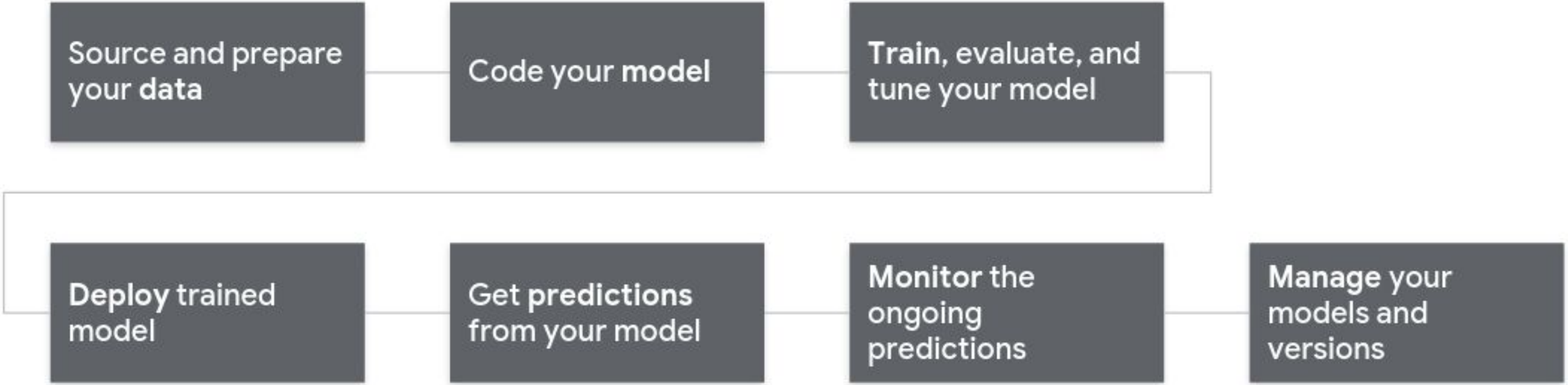
MLOps: Key Components

MLOps means:

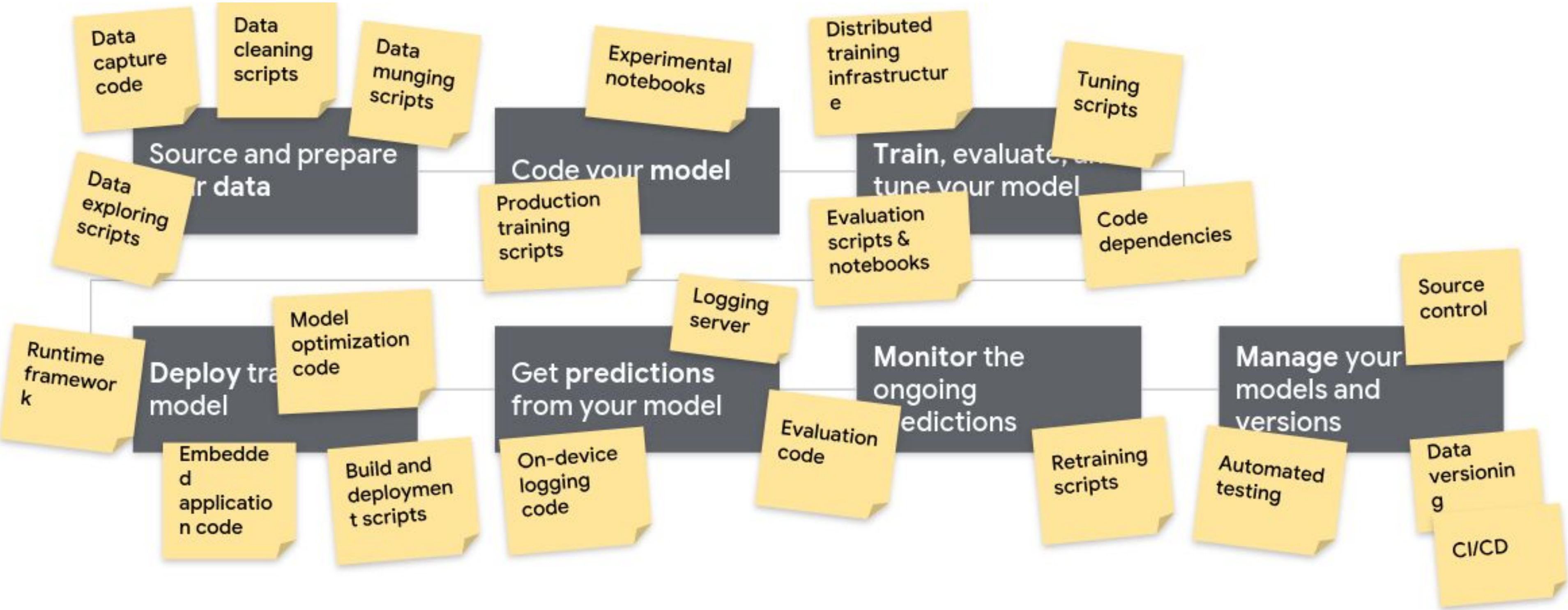
- Running end-to-end
- Managing Complexity
- Evaluating Results
- Improving Models
- Tracking deployment



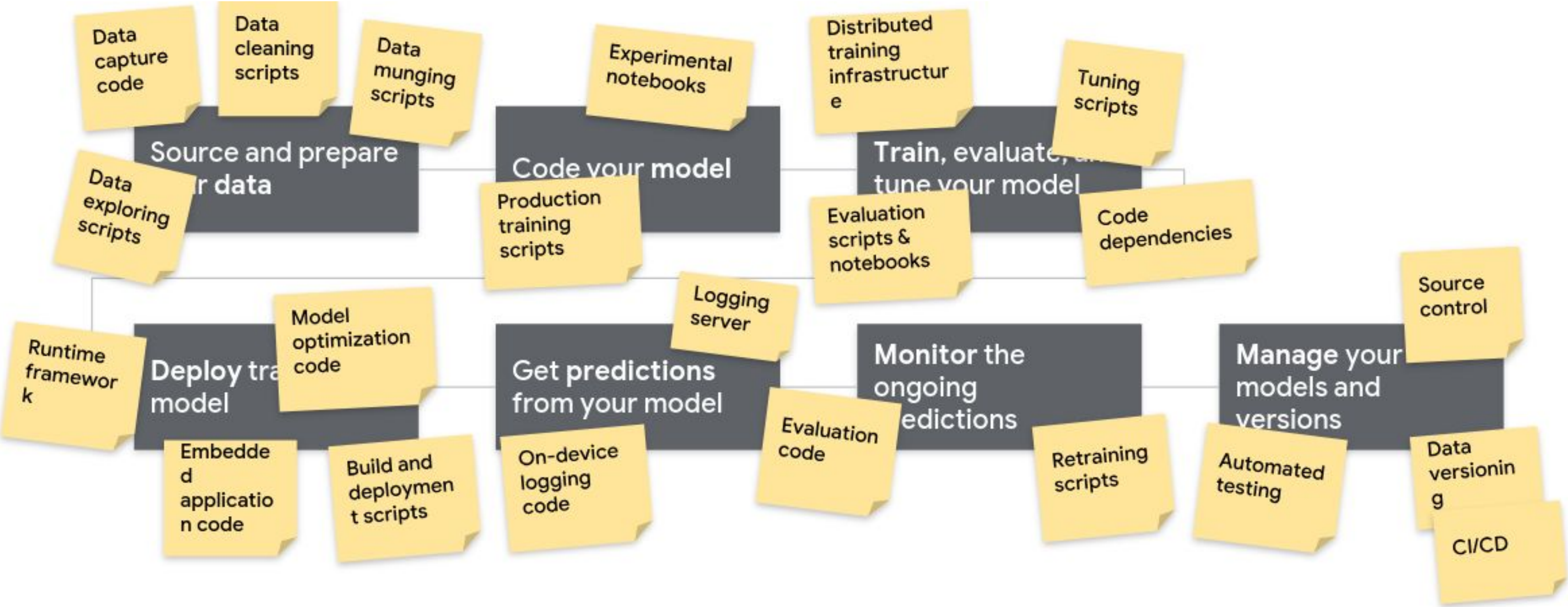
ML Workflow


Source and prepare your data
Code your model
Train, evaluate, and tune your model
Deploy trained model
Get predictions from your model
Monitor the ongoing predictions
Manage your models and versions

ML Workflow

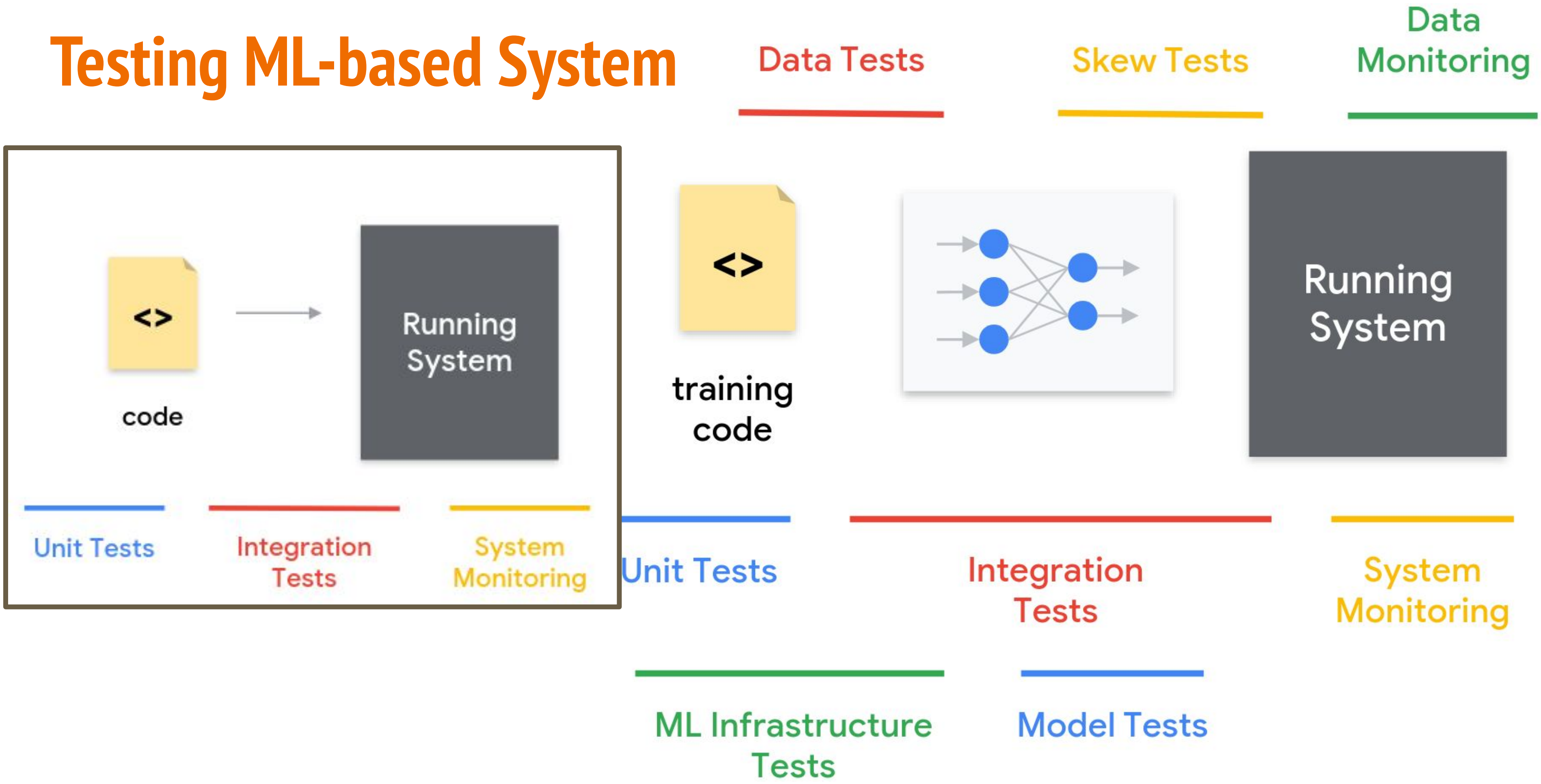

Data capture code
Data cleaning scripts
Data munging scripts
Experimental notebooks
Distributed training infrastructur e
Tuning scripts
Source and prepare r data
Code your model
Train, evaluate, tune your model
Data exploring scripts
Production training scripts
Evaluation scripts & notebooks
Code dependencies
Logging server
Source control
Model optimization code
Runtime framewor k
Deploy tra model
Get predictions from your model
Monitor the ongoing edictions
Manage your models and versions
Embedde d applicatio n code
Build and deploymen t scripts
On-device logging code
Evaluation code
Retraining scripts
Automated testing
Data versionin g
CI/CD

ML Workflow



MLOps = ML Workflow + Automation

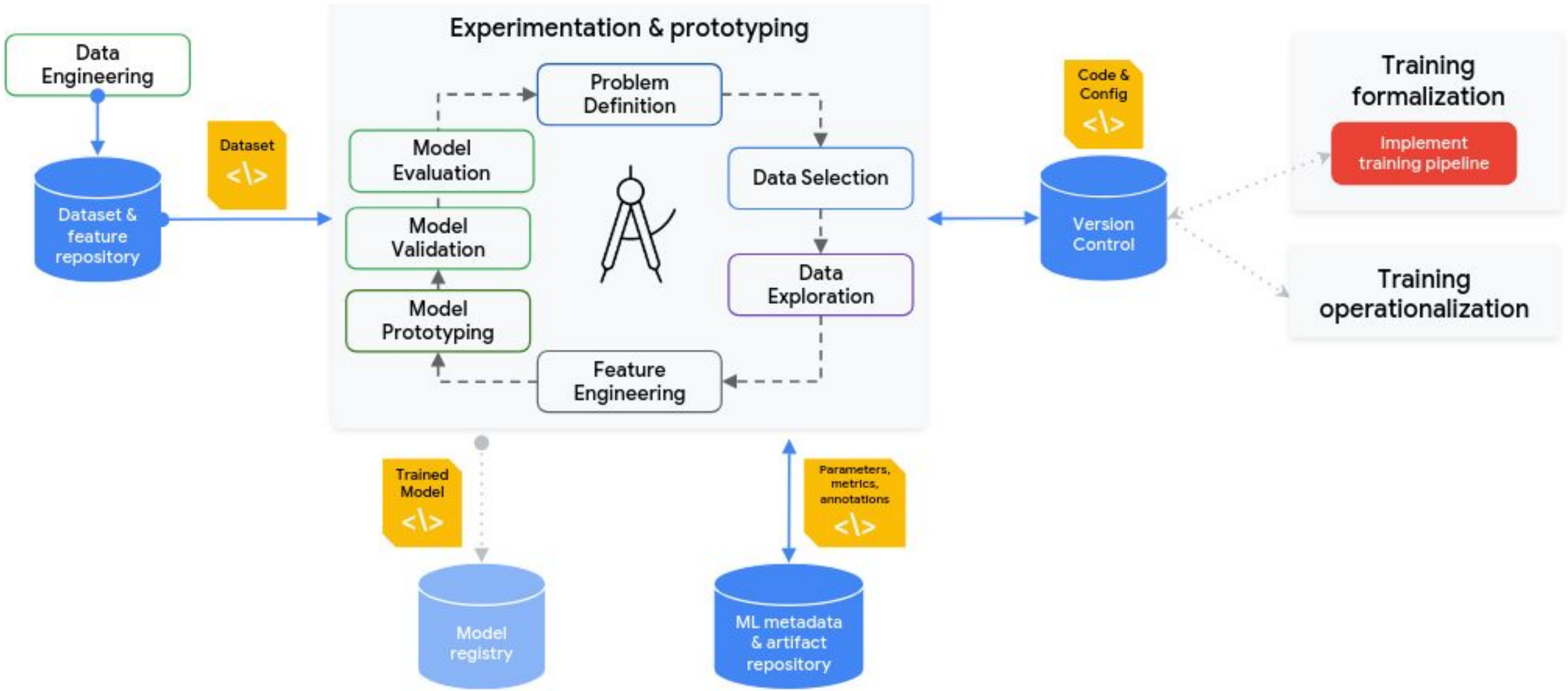
Testing ML-based System



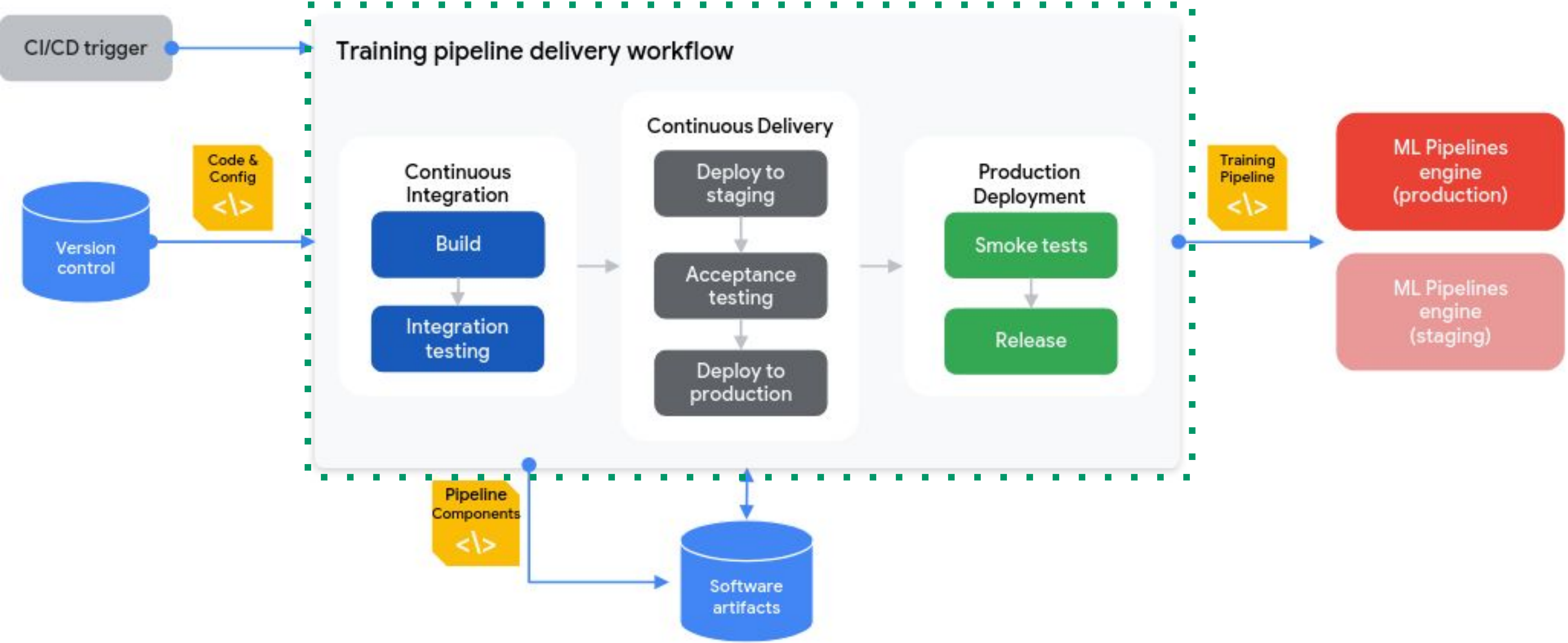
Steps


Data & model management
ML development
Training operationalization
Continuous training
Model deployment
Prediction serving
Continuous monitoring

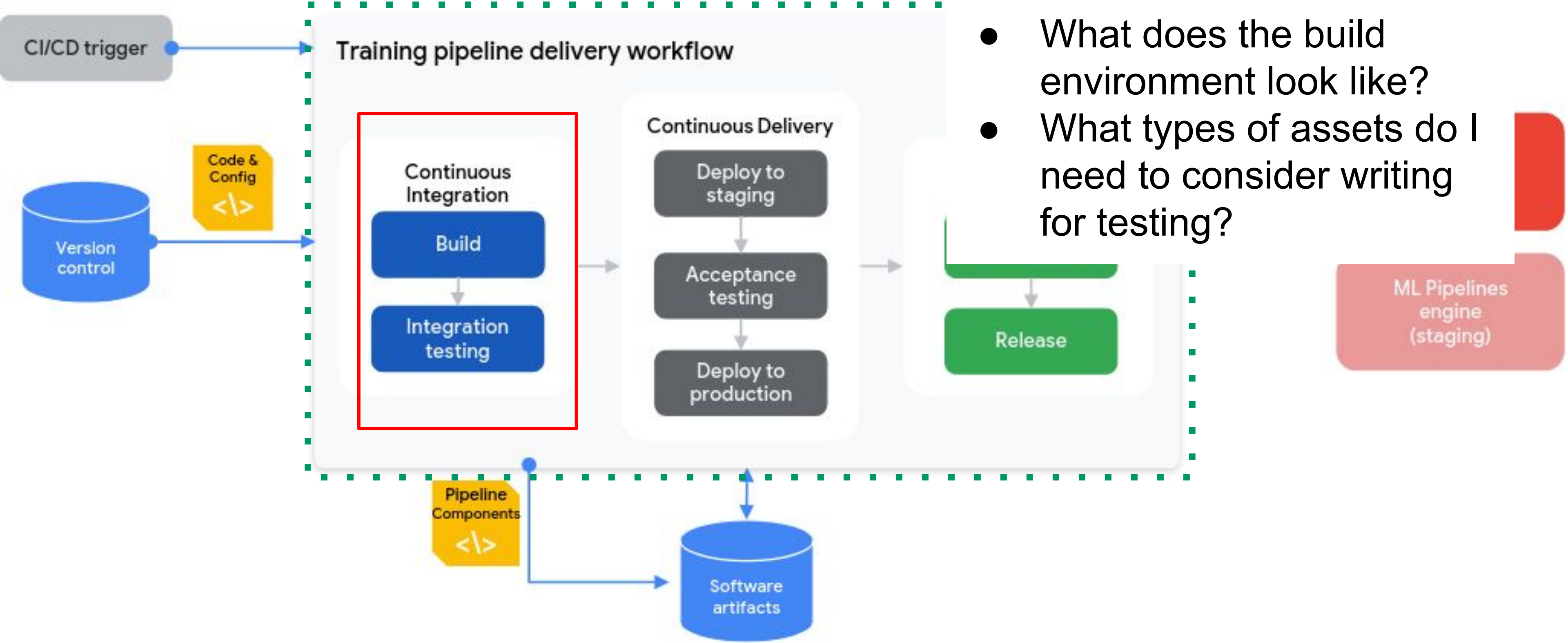
MLOps: ML Development



MLOps: Training Operationalization

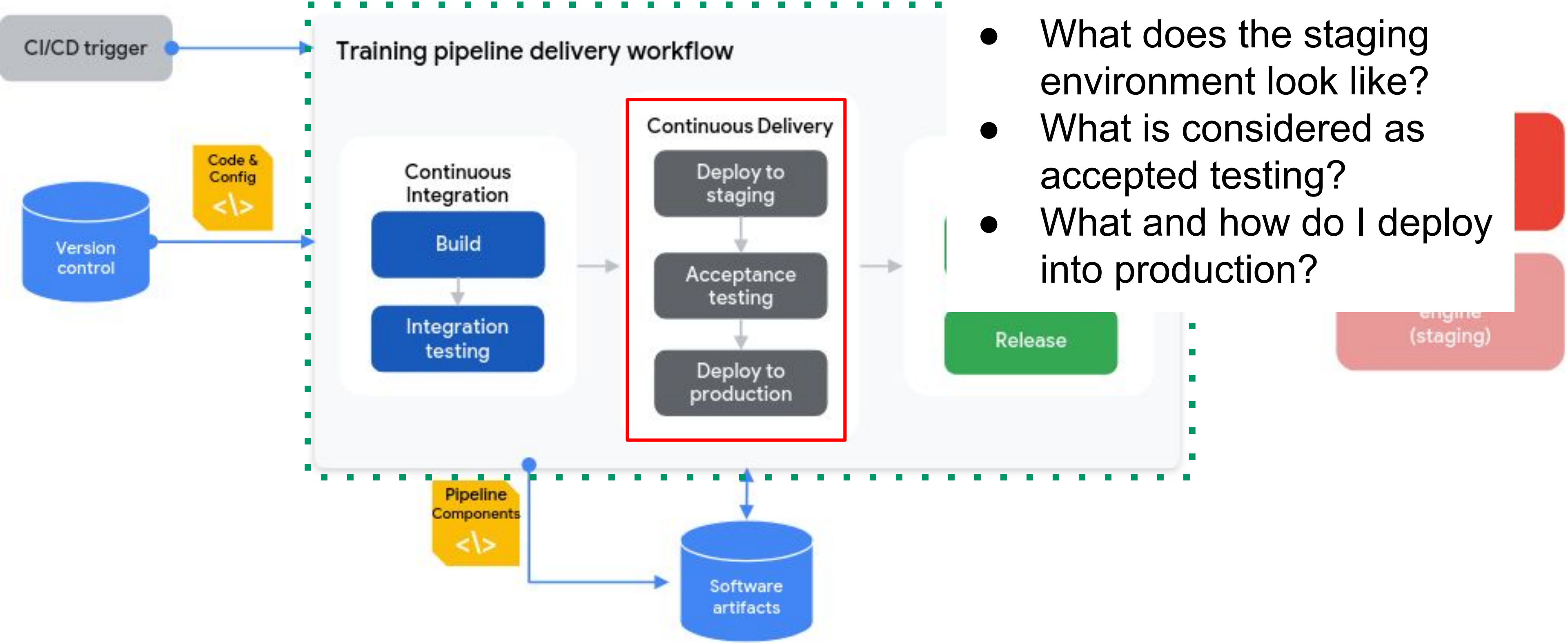


MLOps: Training Operationalization



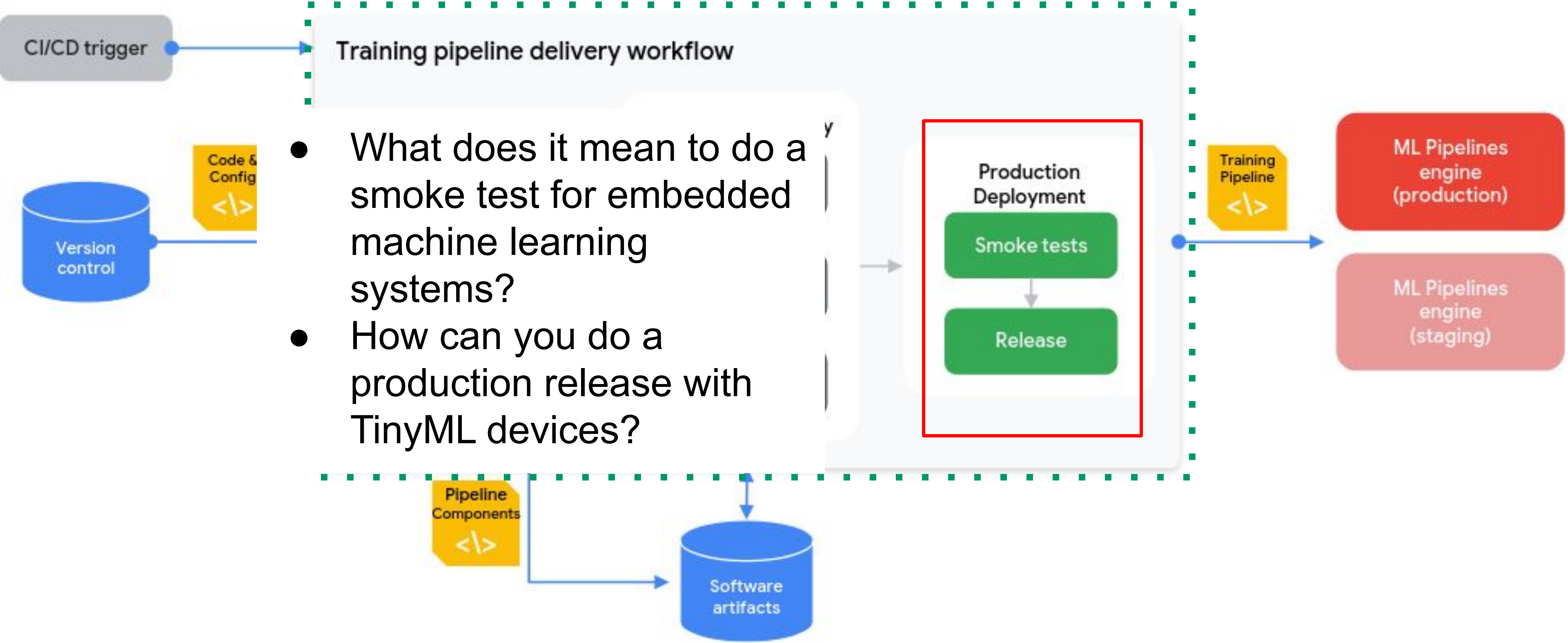
- What does the build environment look like?
- What types of assets do I need to consider writing for testing?

MLOps: Training Operationalization



- What does the staging environment look like?
- What is considered as accepted testing?
- What and how do I deploy into production?

MLOps: Training Operationalization



- What does it mean to do a smoke test for embedded machine learning systems?
- How can you do a production release with TinyML devices?

Deployment Challenges

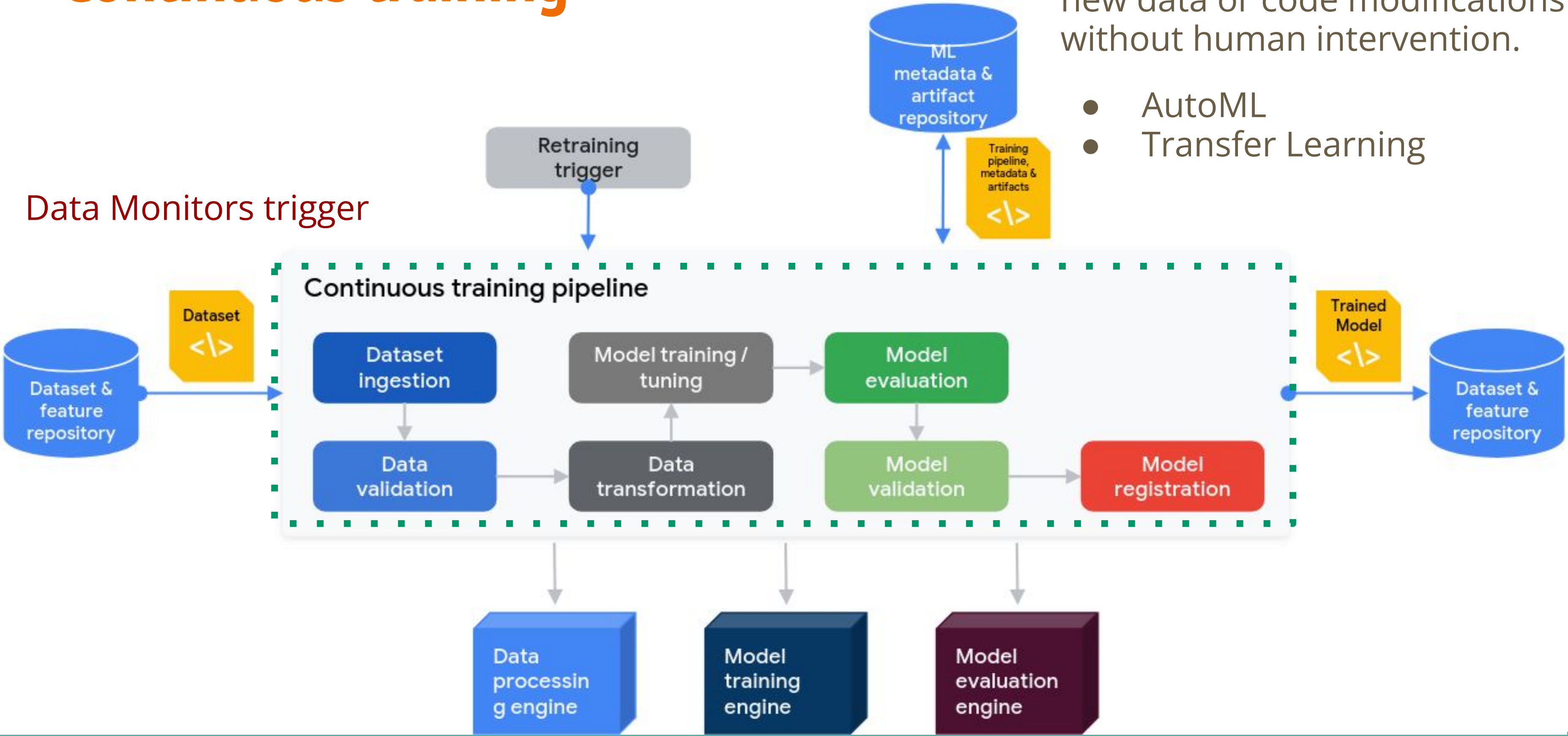
	Board	MCU / ASIC	Clock	Memory	Sensors	Radio
	Himax WE-I Plus EVB	HX6537-A 32-bit EM9D DSP	400 MHz	2MB flash 2MB RAM	Accelerometer, Mic, Camera	None
	Arduino Nano 33 BLE Sense	32-bit nRF52840	64 MHz	1MB flash 256kB RAM	Mic, IMU, Temp, Humidity, Gesture, Pressure, Proximity, Brightness, Color	BLE
	SparkFun Edge 2	32-bit ArtemisV1	48 MHz	1MB flash 384kB RAM	Accelerometer, Mic, Camera	BLE
	Espressif EYE	32-bit ESP32-D0WD	240 MHz	4MB flash 520kB RAM	Mic, Camera	WiFi, BLE

Continuous training

Running the training pipeline on a regular basis, maybe with fresh training settings, in response to new data or code modifications without human intervention.

- AutoML
- Transfer Learning

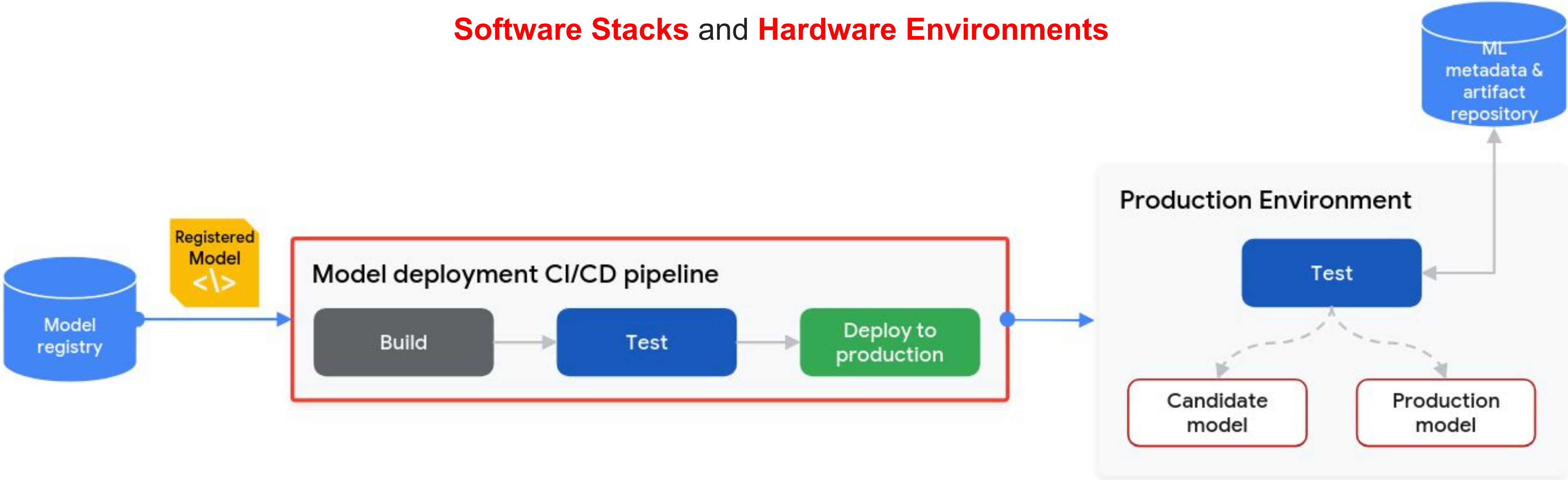
Data Monitors trigger

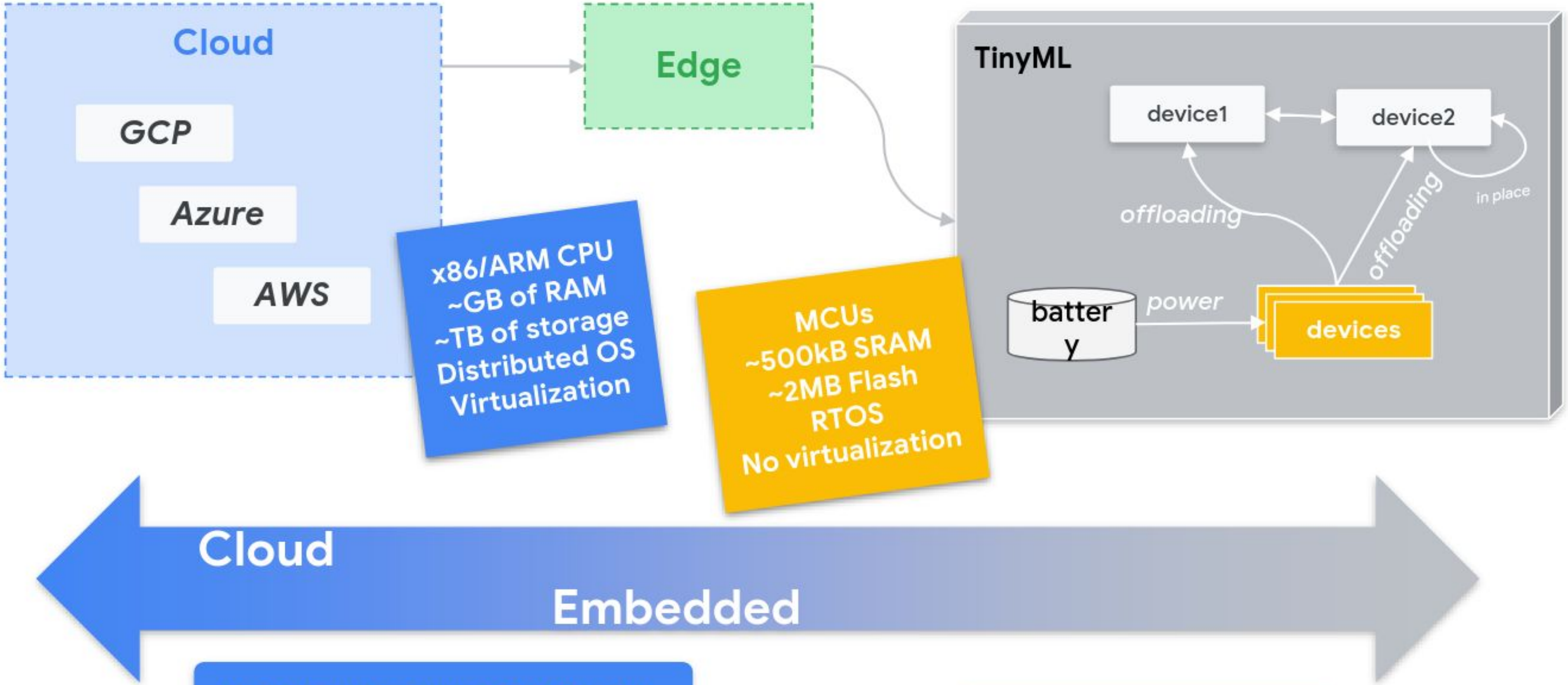


Model Deployment

Packaging, testing, and deploying a model for online experimentation or end users.

Software Stacks and **Hardware Environments**



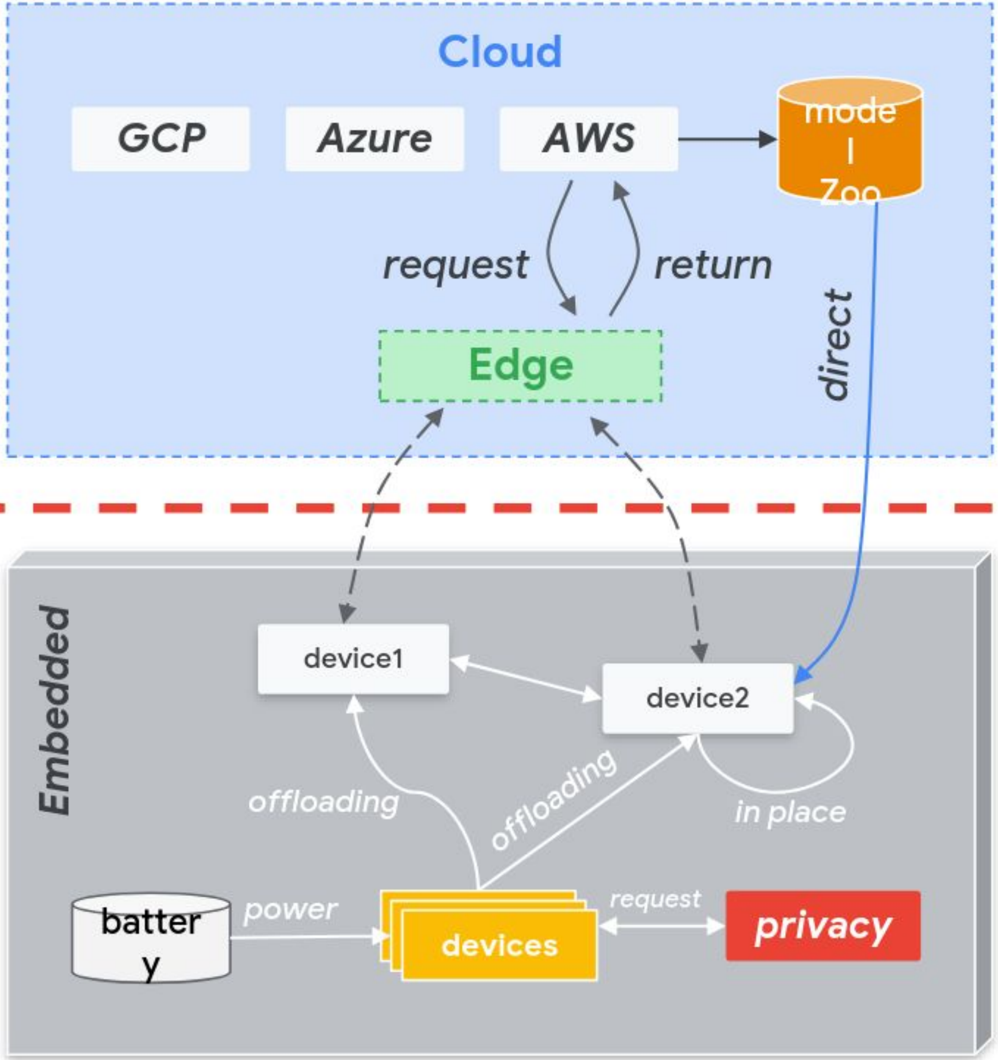


ML-dedicated hardware: CPU, GPU, TPU
ML-dedicated software: many tools
ML Tasks ⟶ Data collection and preprocessing, data transformation, model training, model deployment, inference

No ML-dedicated Hardware
No ML-dedicated software
ML Tasks ⟶ Inference

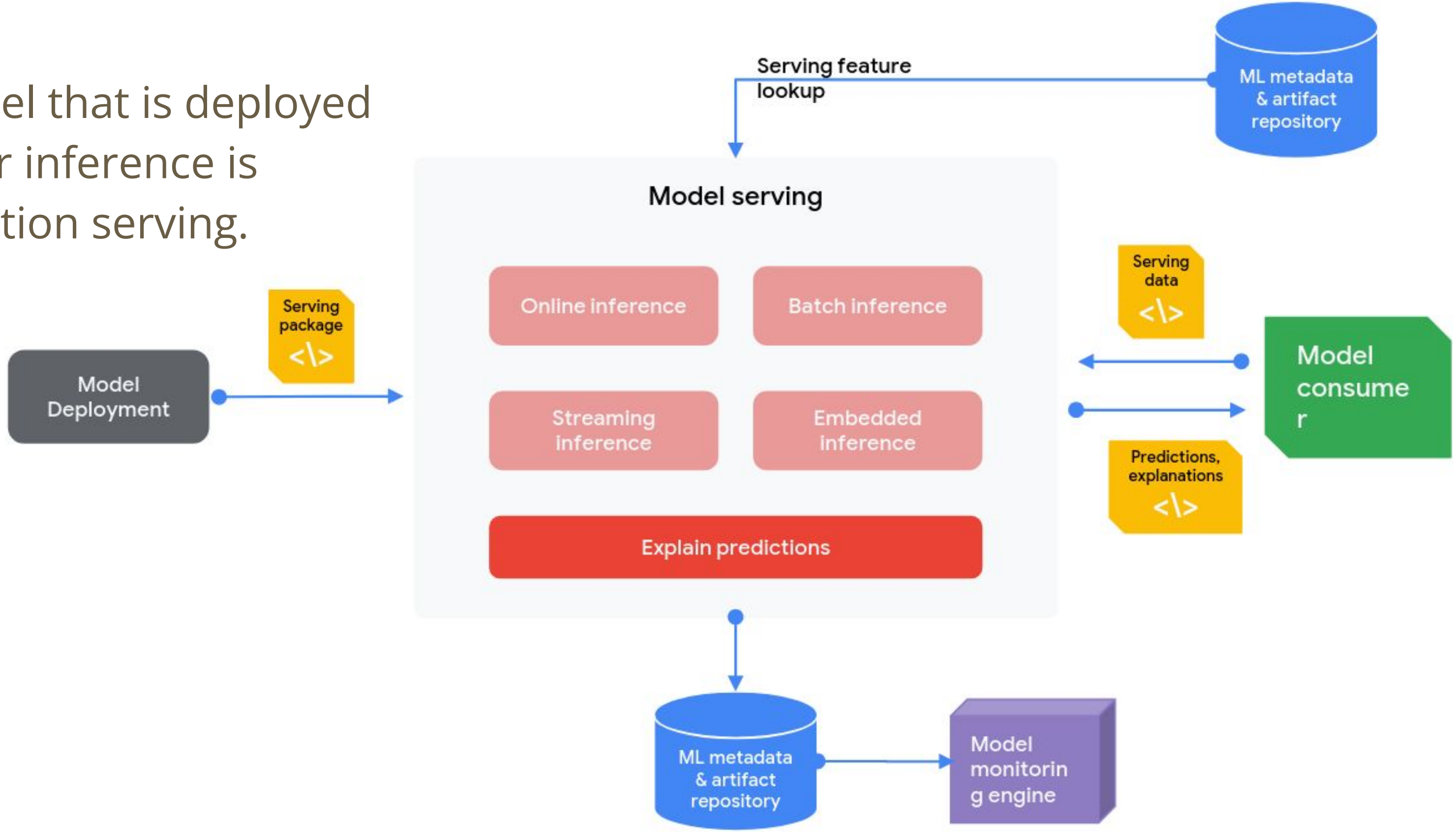
Decouple the cloud development environment from the embedded model deployment environment

Simplify deployment of ML models to tiny devices and develop an abstraction



Prediction Serving

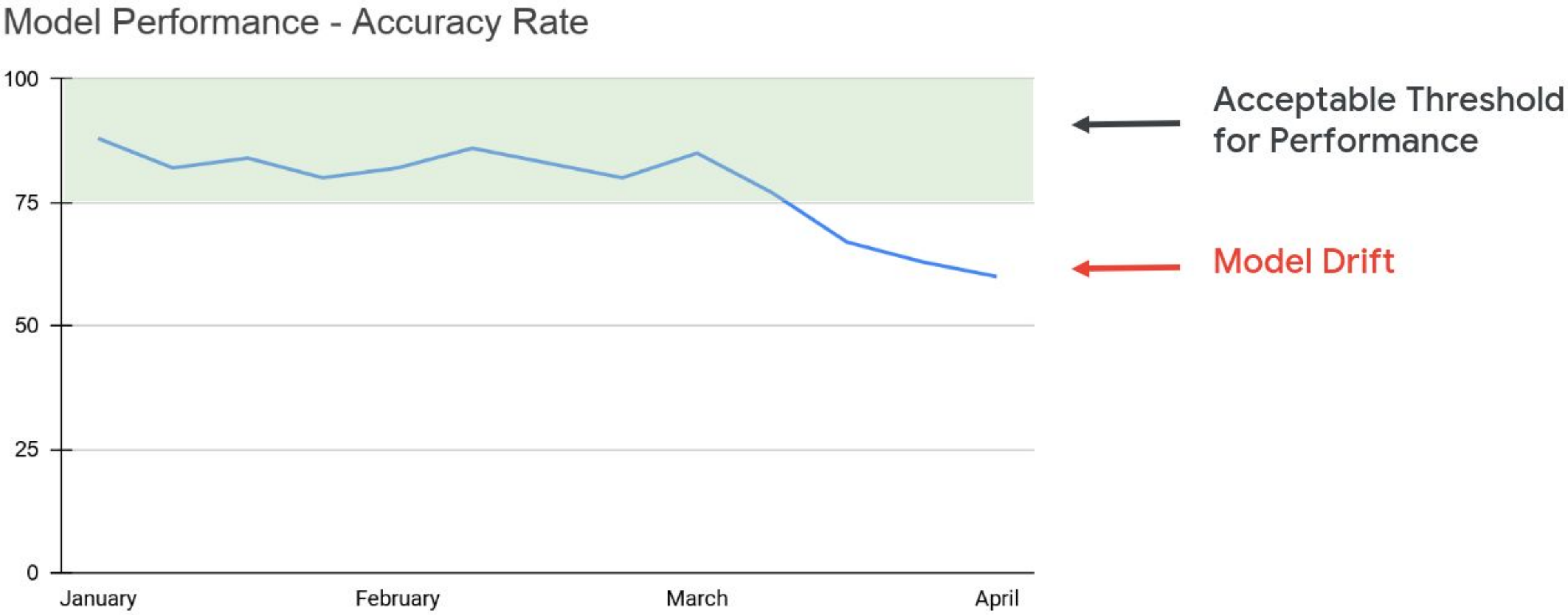
Serving the model that is deployed in production for inference is known as prediction serving.



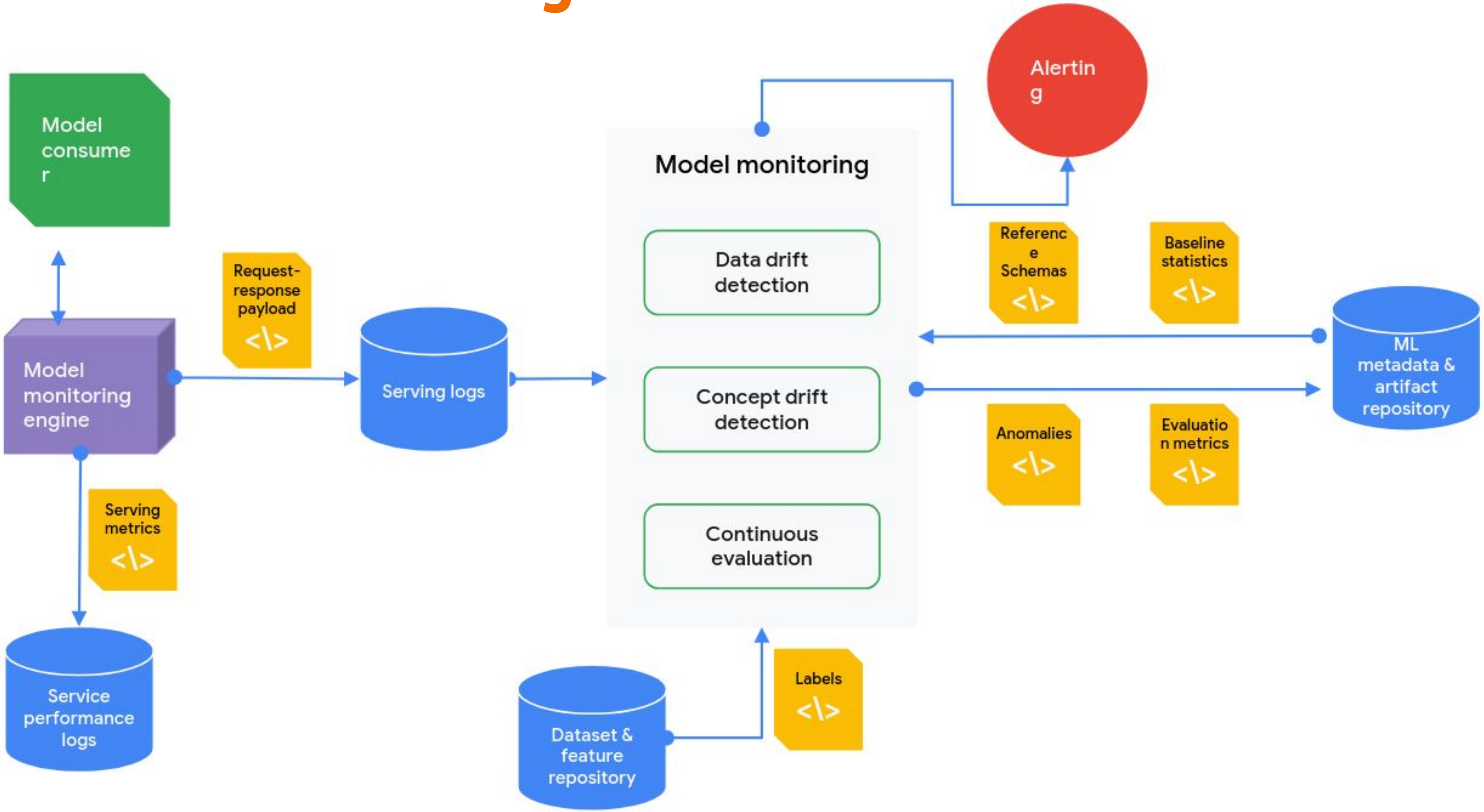
	Scenario	Metric
	**Batch inference** (e.g. photo sorting app)	**Throughput**
	**Online inference** (e.g. translation app)	**QPS** subject to latency bound
	**Streaming inference** (e.g. multiple camera driving assistance)	**Number streams** subject to latency bound
	**Embedded inference** (e.g. cell phone augmented vision)	**Latency**

Continuous Monitoring

Continuous monitoring refers to keeping track of a deployed model's effectiveness and efficiency.



Continuous Monitoring



Drift Types

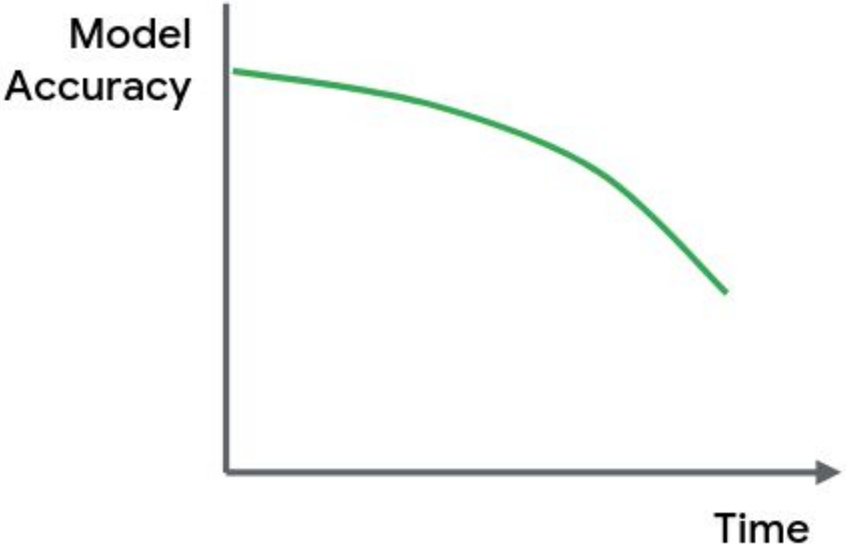
Concept Drift

- the affected old data needs to be relabeled
- Concept drift in machine learning is when the relationship between the input and target changes over time.

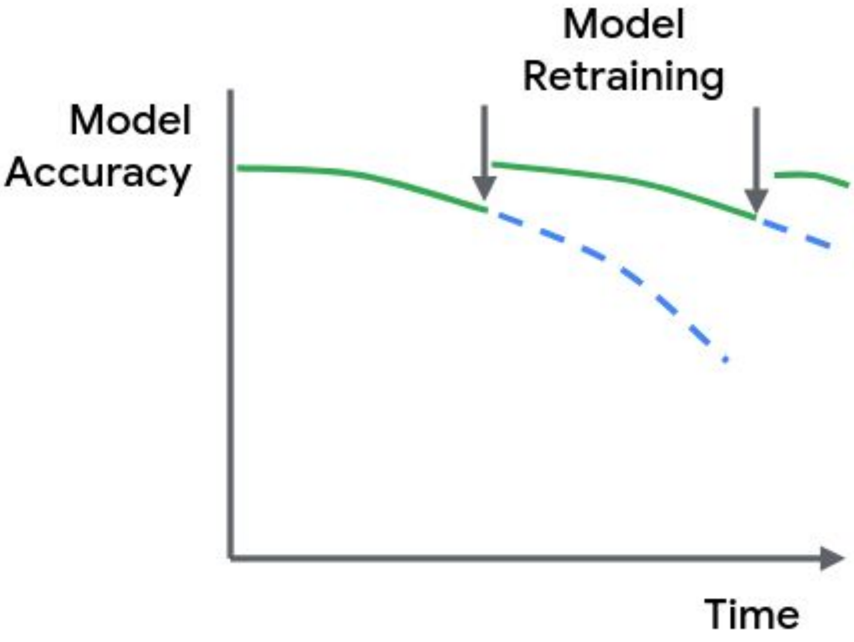
Data Drift

- enough new data needs to be labeled
- Data drift is a change in the distribution of data over time.

Goal of Continuous Training



Model Decay over time



Regularly updated model

Continuous Monitoring for TinyML

- Monitoring may not always be a feasible option
  - Low power communication protocol
  - Device isn't wifi-enabled
- Monitoring opens up security and privacy risks

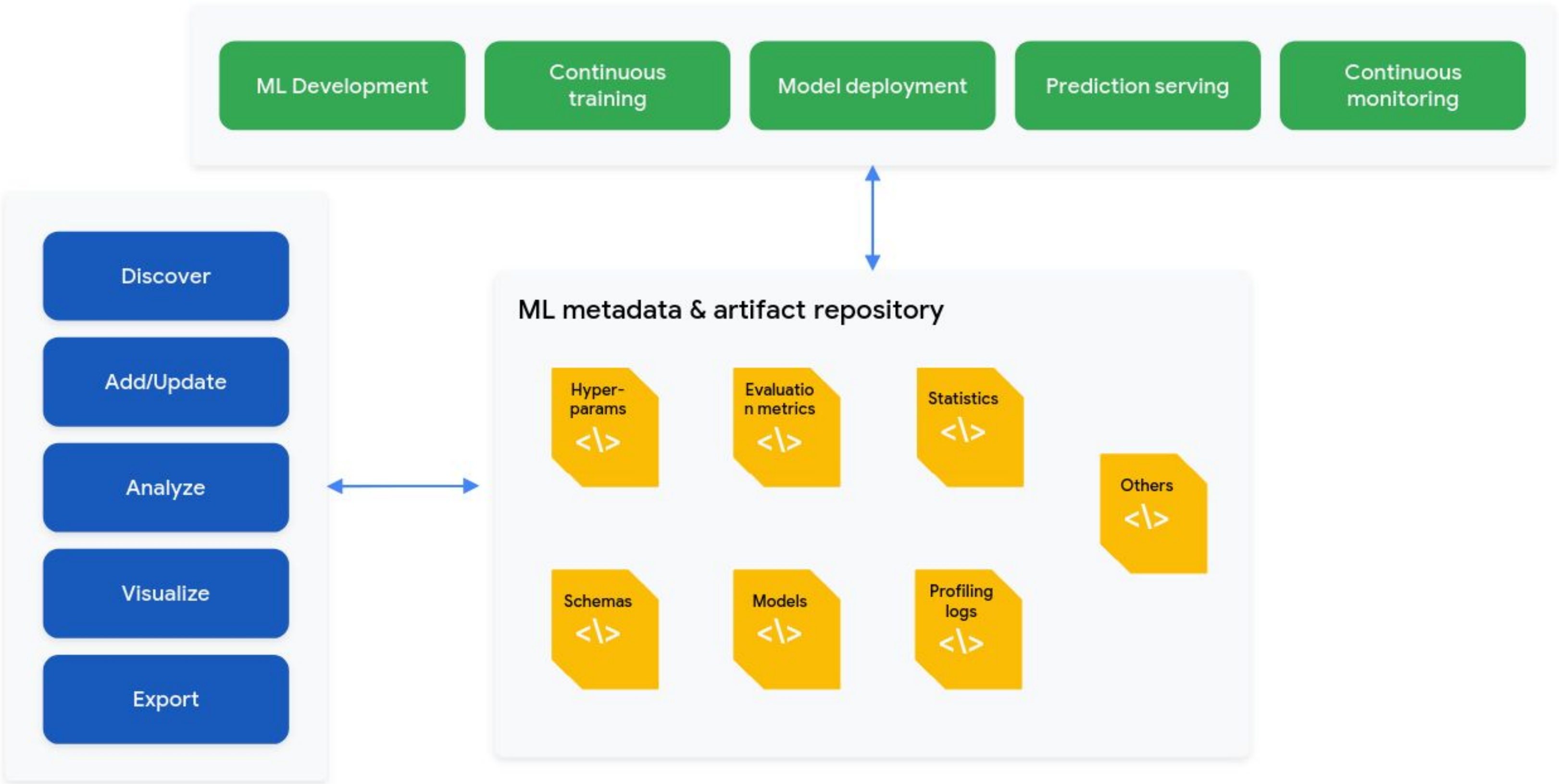
Continuous Monitoring for TinyML

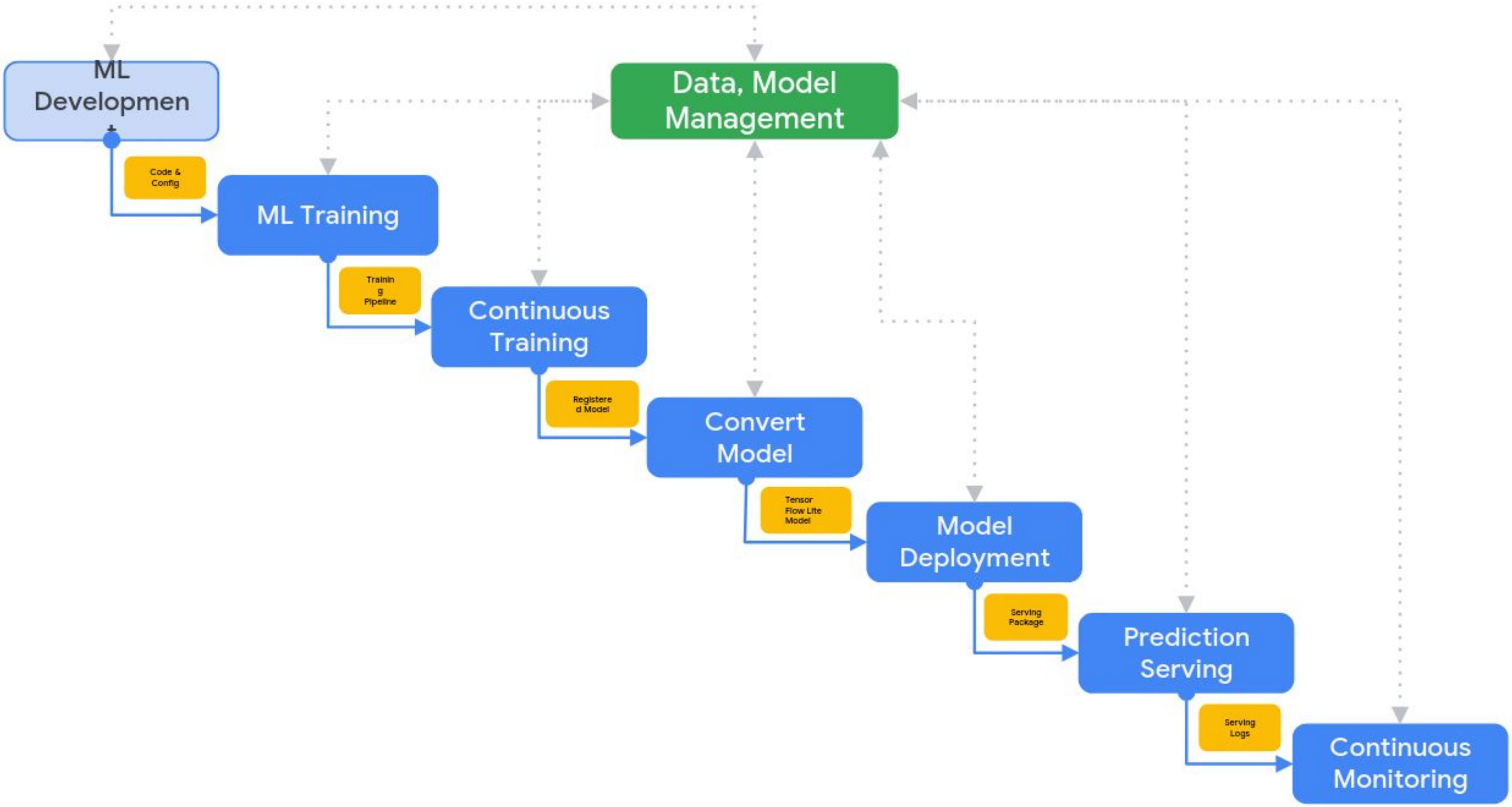
- Monitoring may not always be a feasible option
  - Low power communication protocol
  - Device isn’t wifi-enabled
- Monitoring opens up security and privacy risks
- How can we enable Continuous Monitoring to enable Continuous Training without moving the data off the endpoint tiny ML device?

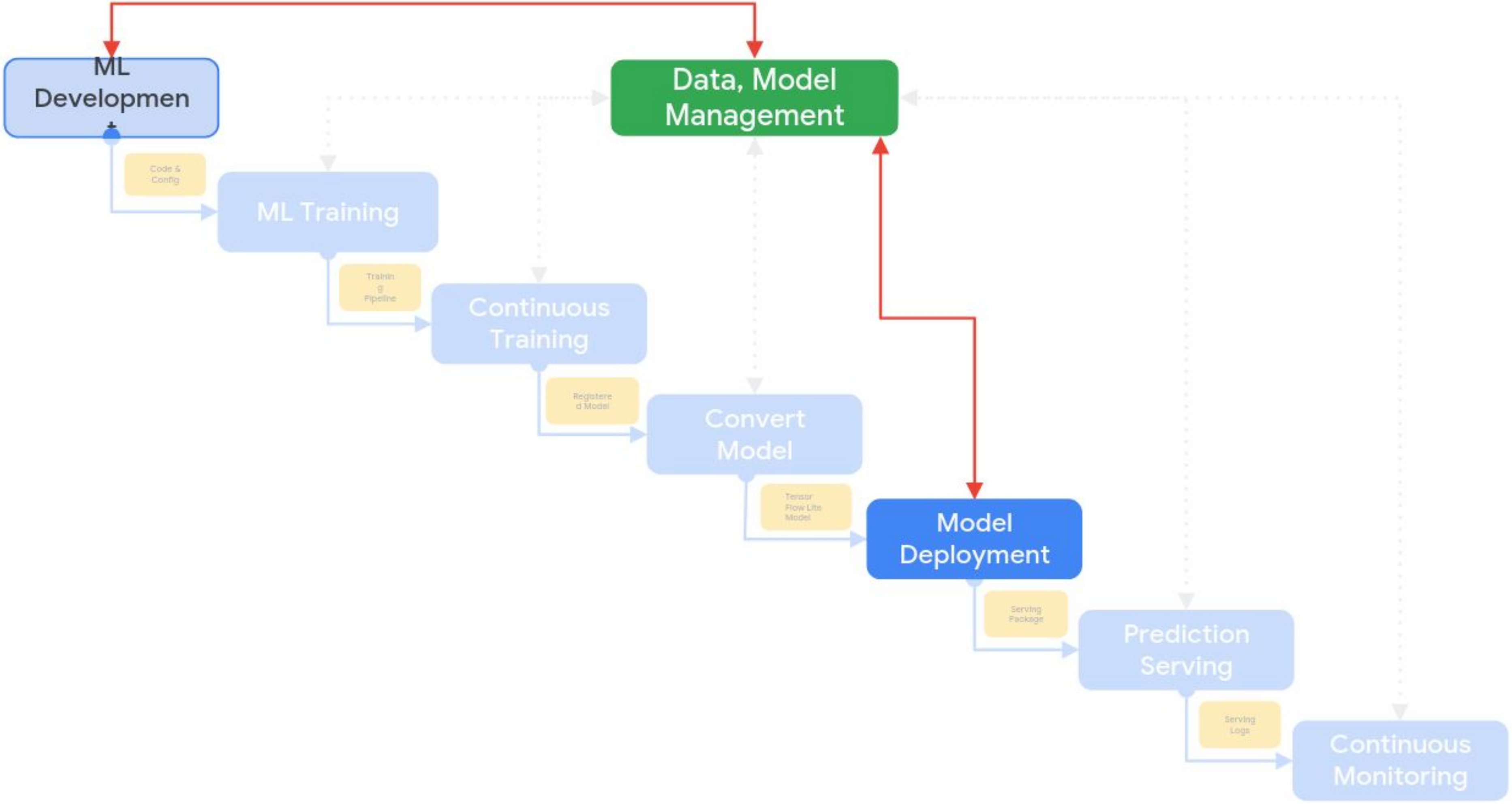
Data & Model Management

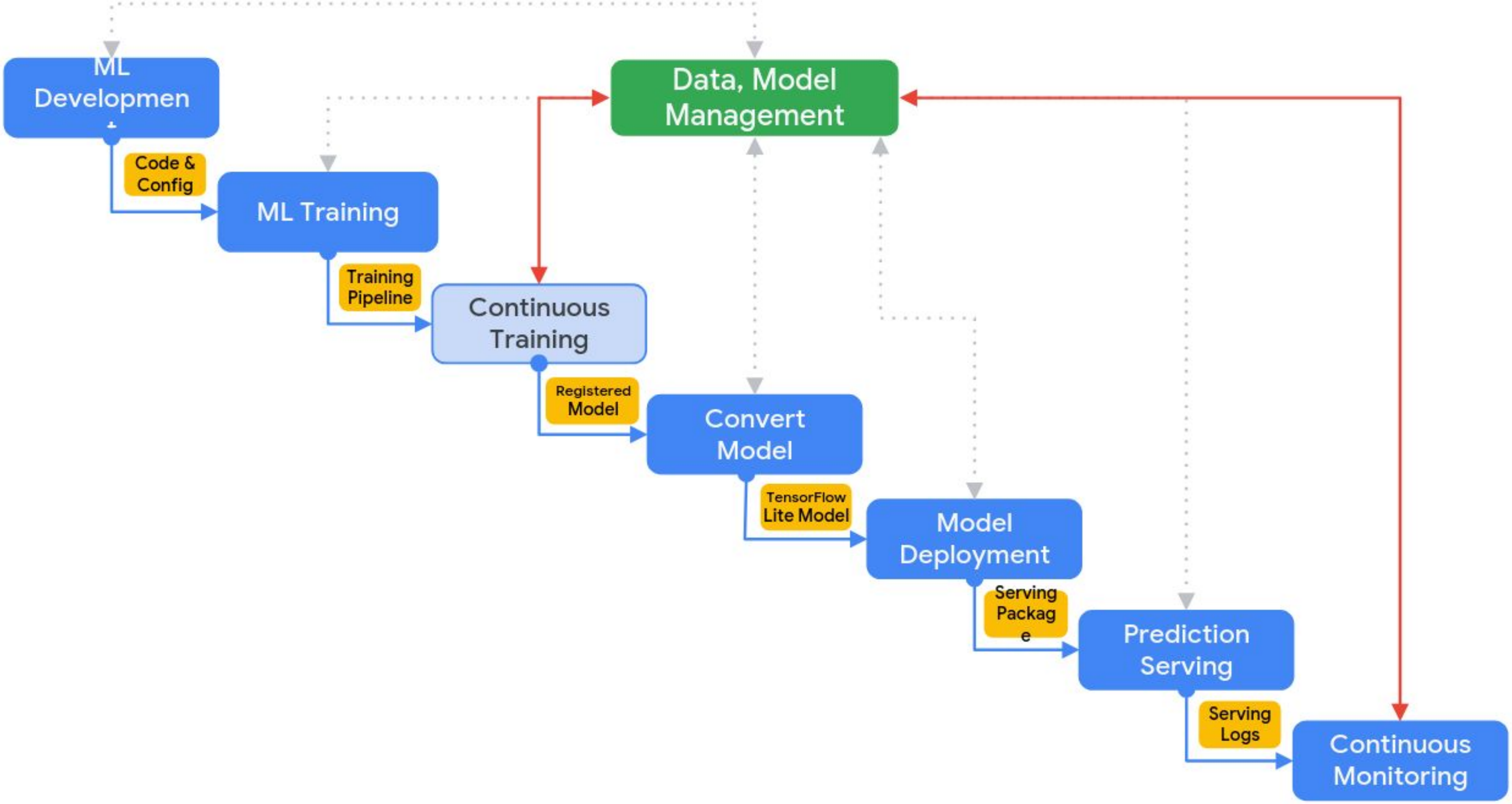
Data and model management is a central, cross-cutting function for governing ML artifacts to support ability, traceability, and compliance. Data and model management can also promote shareability, reusability, and discoverability of ML assets.

Data & Model Management

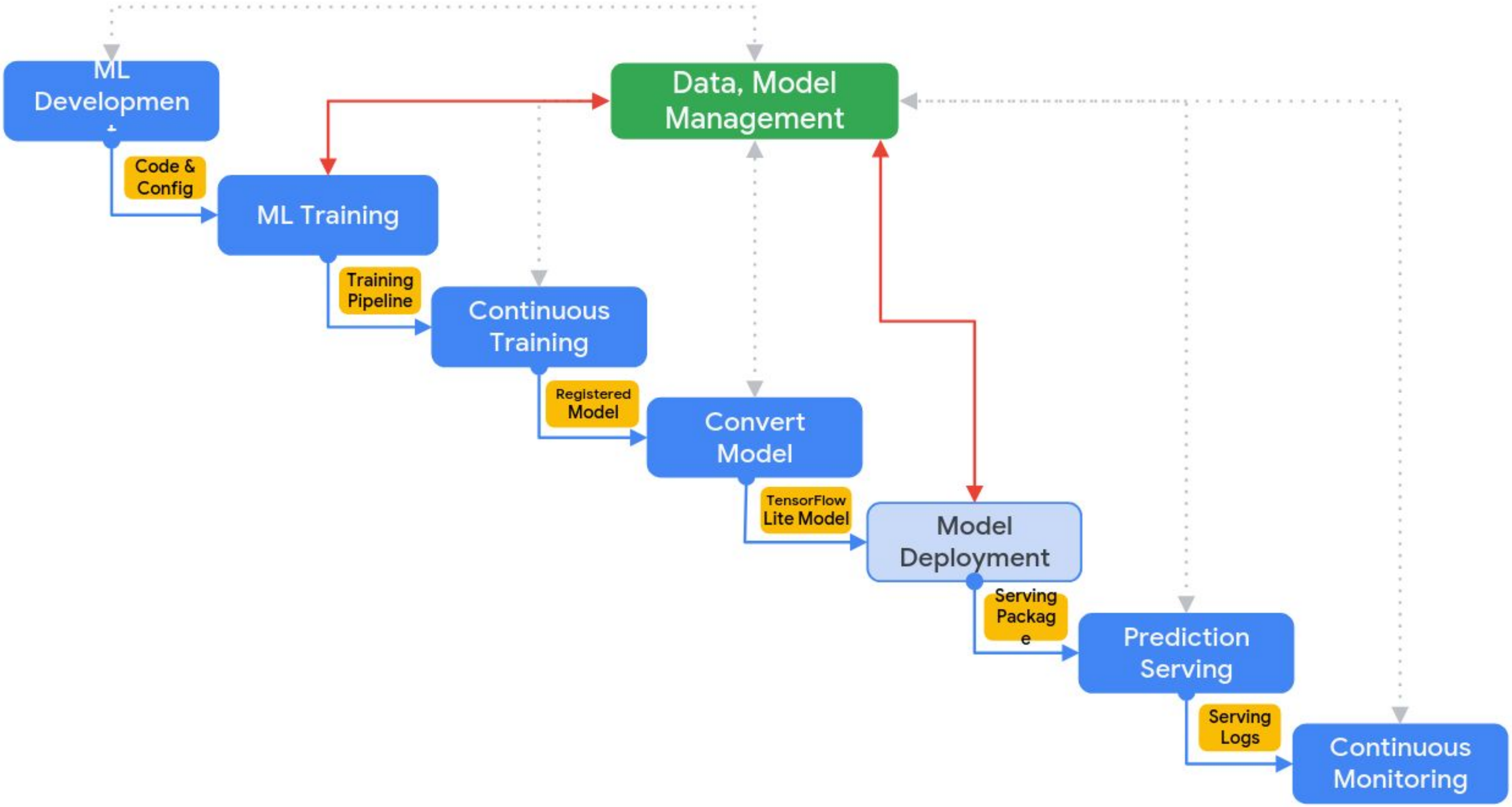


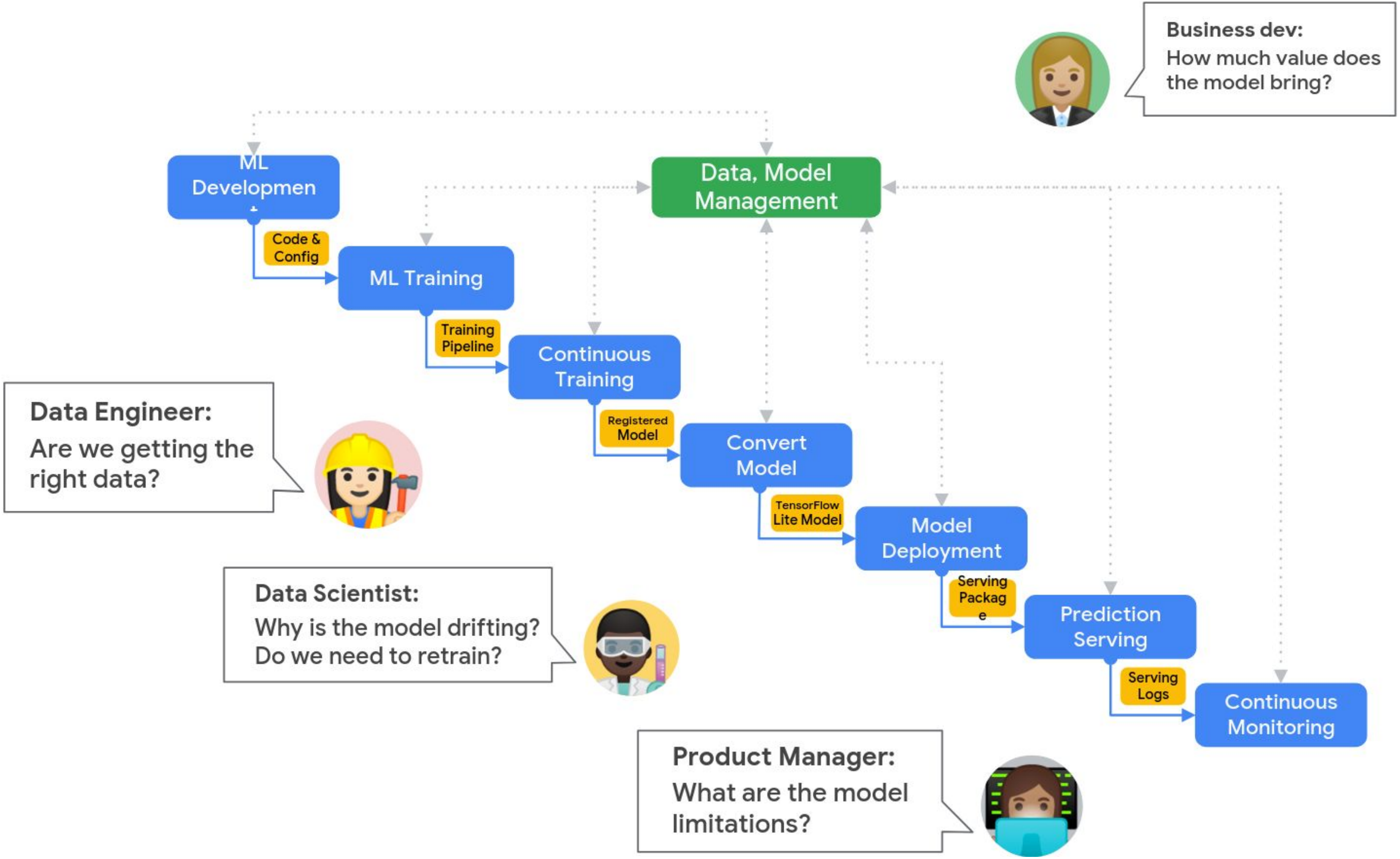

ML Developmen t
Code & Config
ML Training
Trainin g Pipeline
Continuous Training
Registere d Model
Convert Model
Tensor Flow Lite Model
Model Deployment
Serving Package
Prediction Serving
Serving Logs
Continuous Monitoring
Data, Model Management


ML Developmen t
Data, Model Management
Code & Config
ML Training
Trainin g Pipeline
Continuous Training
Registere d Model
Convert Model
Tensor Flow Lite Model
Model Deployment
Serving Package
Prediction Serving
Serving Logs
Continuous Monitoring



ML Developmen t
Code & Config
ML Training
Training Pipeline
Continuous Training
Registered Model
Convert Model
TensorFlow Lite Model
Model Deployment
Serving Packag e
Prediction Serving
Serving Logs
Continuous Monitoring
Data, Model Management


ML Developmen t
Code & Config
ML Training
Training Pipeline
Continuous Training
Data, Model Management
Registered Model
Convert Model
TensorFlow Lite Model
Model Deployment
Serving Packag e
Prediction Serving
Serving Logs
Continuous Monitoring



Business dev:
How much value does the model bring?
ML Developmen t
Data, Model Management
Code & Config
ML Training
Training Pipeline
Continuous Training
Registered Model
Convert Model
TensorFlow Lite Model
Model Deployment
Serving Packag e
Prediction Serving
Serving Logs
Continuous Monitoring
Data Engineer:
Are we getting the right data?
Data Scientist:
Why is the model drifting? Do we need to retrain?
Product Manager:
What are the model limitations?

Question

TensorFlow and TensorFlow Lite support the same set of operations.

A. True, TensorFlow Lite is simply a more optimized version of TensorFlow.
B. False, TensorFlow Lite is an optimized subset of TensorFlow designed for mobile inference.

Answer

TensorFlow and TensorFlow Lite support the same set of operations.

A. True, TensorFlow Lite is simply a more optimized version of TensorFlow.
B. False, TensorFlow Lite is an optimized subset of TensorFlow designed for mobile inference.

Question

Transfer learning can be used to:

A. Shorten the training process by re-using many layer values from an existing model.
B. Repurpose a dataset by extracting many values for another application.
C. Train models in the cloud and then deploy them on device.

Answer

Transfer learning can be used to:

A. Shorten the training process by re-using many layer values from an existing model.
B. Repurpose a dataset by extracting many values for another application.
C. Train models in the cloud and then deploy them on device.

Thank You