# DS-UA 301
# Advanced Topics in Data Science
## *Advanced Techniques in ML and Deep Learning*

**LECTURE 13**

**Parijat Dube**

# ONNX

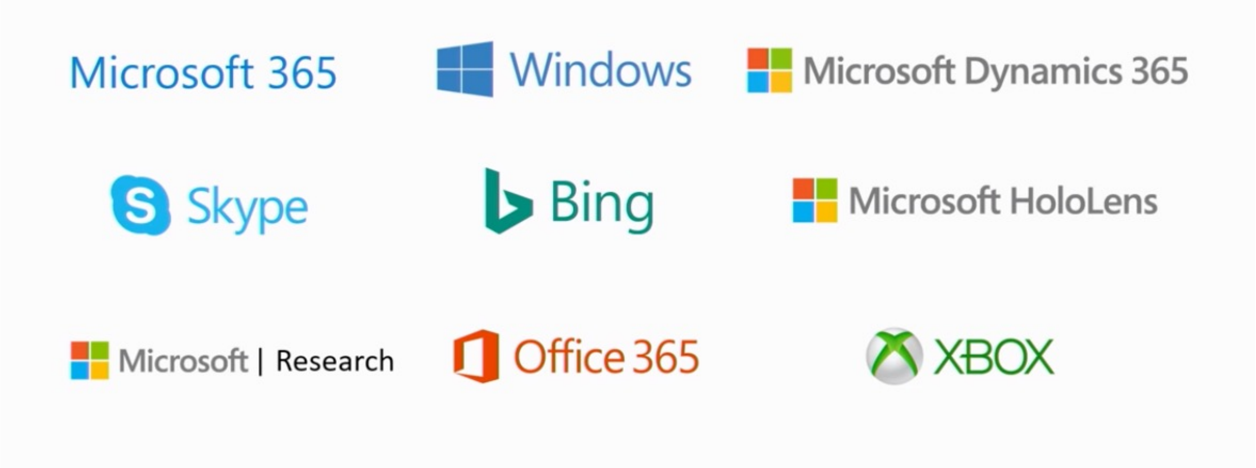
- Open Neural Network Exchange
- An open format to represent traditional machine learning and deep learning models
- ONNX Features
  - Framework interoperability
    - Models trained in one DL framework to be transferred to another for inference
  - Hardware optimizations
    - ONNX-compatible runtimes and libraries designed to maximize performance on specific DL hardware
- Supported by a community of over 20 leading companies
- Visit [http://onnx.ai](http://onnx.ai)

# ONNX Capabilities

- Common set of operators related to ML
- Common file format for representing ML models
- Supported Tools
  - Visit http://onnx.ai/supported-tools
- ONNX Tutorials
  - Visit https://github.com/onnx/tutorials
- Model Zoo
  - A collection of pre-trained, state-of-the-art models in the ONNX format contributed by community members
  - Visit https://github.com/onnx/models

# ONNX in Enterprise: Microsoft

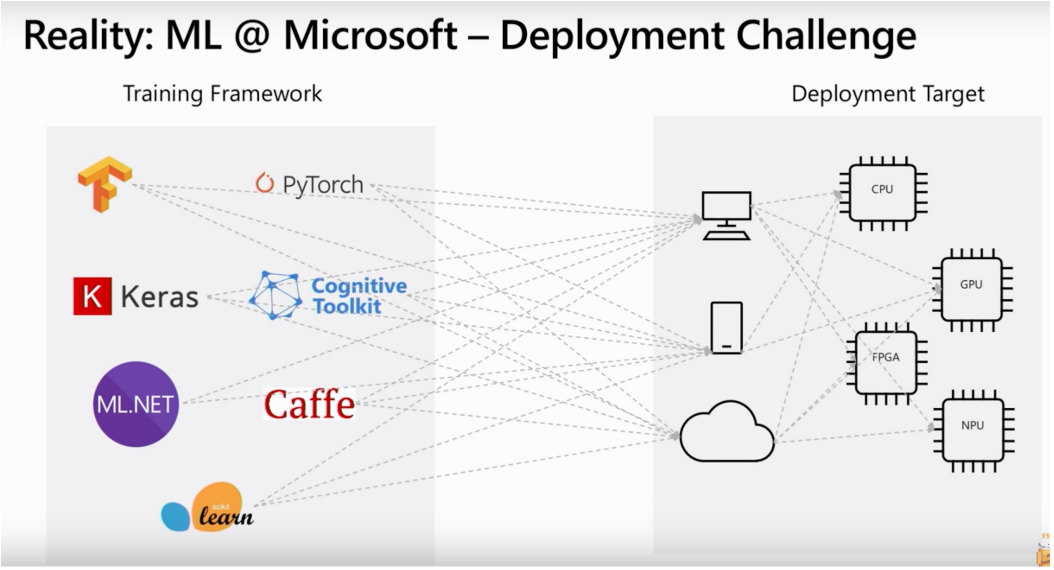
ML used in many products to improve performance and productivity




- AI developers work with different frameworks
- Deployment of trained models to production
  - Different deployment targets: cloud, IoT devices, edge devices
  - Different hardware: CPUs, GPUs, TPUs, FPGAs

Youtube Video: Open Neural Network Exchange (ONNX) in the enterprise: how Microsoft scales ML

# Multiple ML frameworks + multiple deployment targets

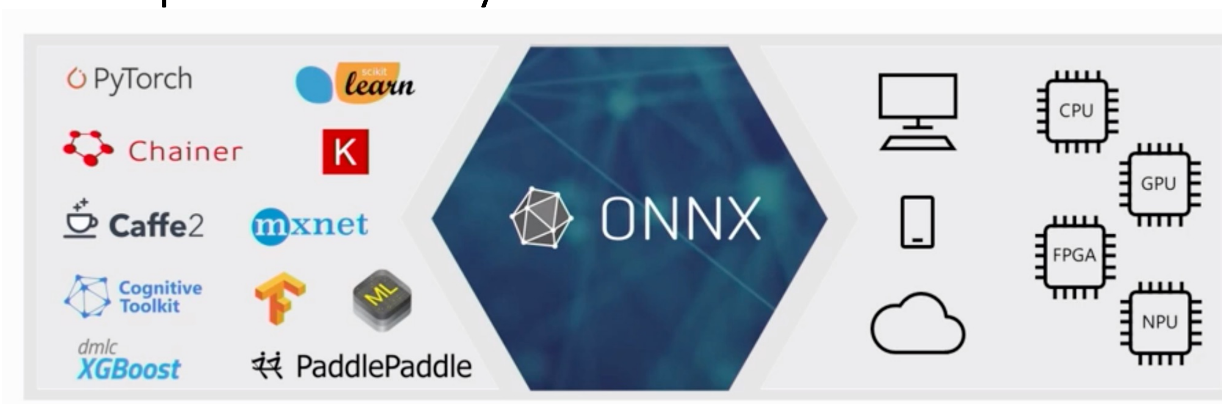



- Maintaining multiple frameworks in deployment
  - Not scalable
  - Hard to maintain
  - Degrades application performance
    - Frameworks compete for system resources
- Decouple training and deployment frameworks

Youtube Video: Open Neural Network Exchange (ONNX) in the enterprise: how Microsoft scales ML

# Bridge Model Training and Deployment

Open and Interoperable industry wide standard




Build and train model in framework A → Convert trained model in framework A to ONNX model → Convert ONNX model to framework B and deploy

Convert trained model in framework A to ONNX model → Deploy ONNX model using ONNX Runtime

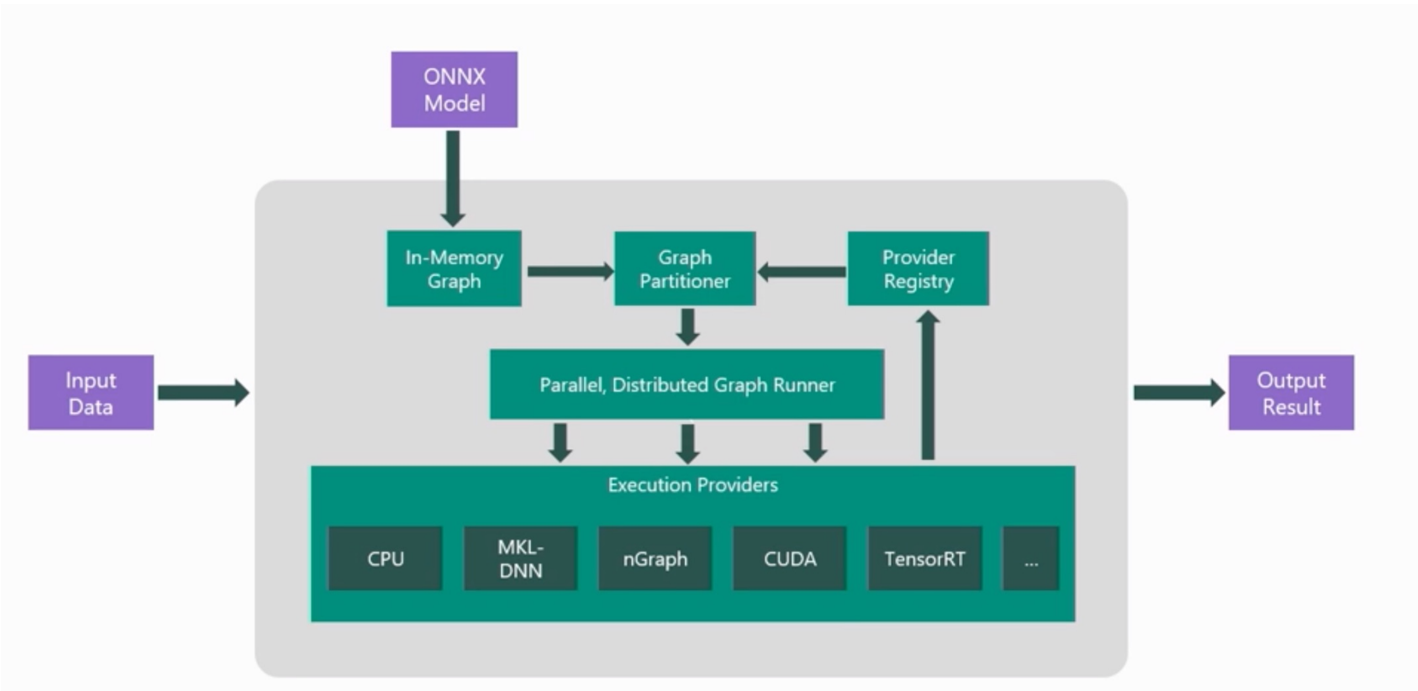
# Example Use Case

- Pytorch-ONNX-CoreML
  - https://attardi.org/pytorch-and-coreml

# ONNX Runtime


- ONNX Runtime is a cross-platform inference and training machine-learning accelerator.
- Available on Mac, Windows, Linux platforms
- Supports full ONNX-ML spec
- Open-sourced https://github.com/microsoft/onnxruntime

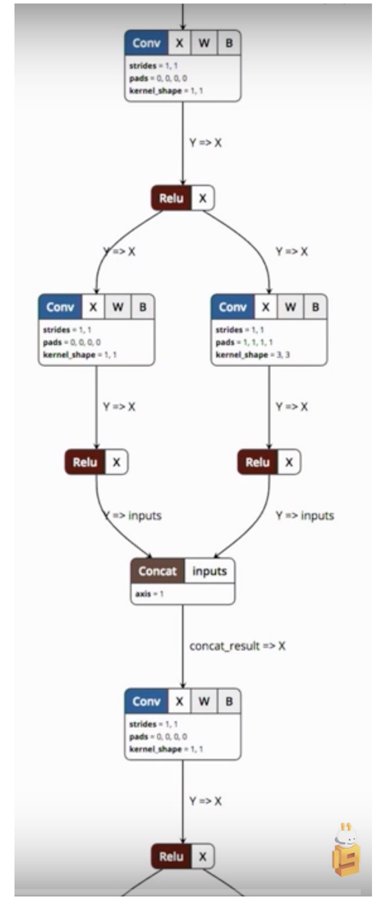
# ONNX Runtime Architecture




ONNX Runtime has a graph parser which takes ONNX model, parses the graph, applies runtime optimizations like fusion ops, executes portions of the graph on specific hardware, provides the inference output result

Youtube Video: Open Neural Network Exchange (ONNX) in the enterprise: how Microsoft scales ML

Computational dataflow graph




Frameworks provide implementations of ONNX operators

# Factors Contributing to Democratization of AI

- Open benchmarking communities
  - Kaggle, Open ML
- ONNX
  - Faster and broader reuse of models; Shorten time to move to production
  - Hardware optimizations can be leveraged by many developers easily
  - Makes deep learning models portable thus preventing framework lock in
  - Greater interoperability in AI tools community
- Cloud based AI solutions

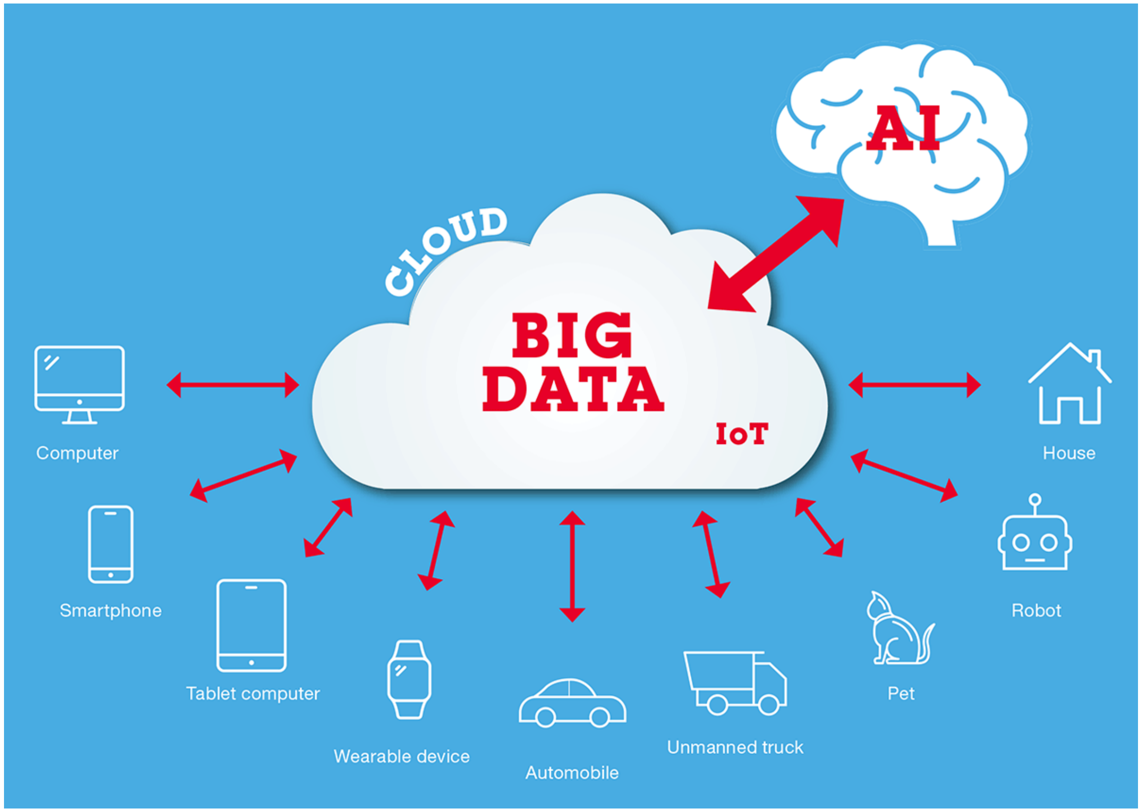
# Cloud Computing

- Access to computing resources and storage on demand
- Pay-as-you go model
- Heterogeneous resources: GPUs, CPUs, storage type
- Different offering models: IaaS, PaaS, SaaS, MLaaS
- Different deployment models: Public, private, hybrid cloud
- Provisioning, maintenance, monitoring, life-cycle-management

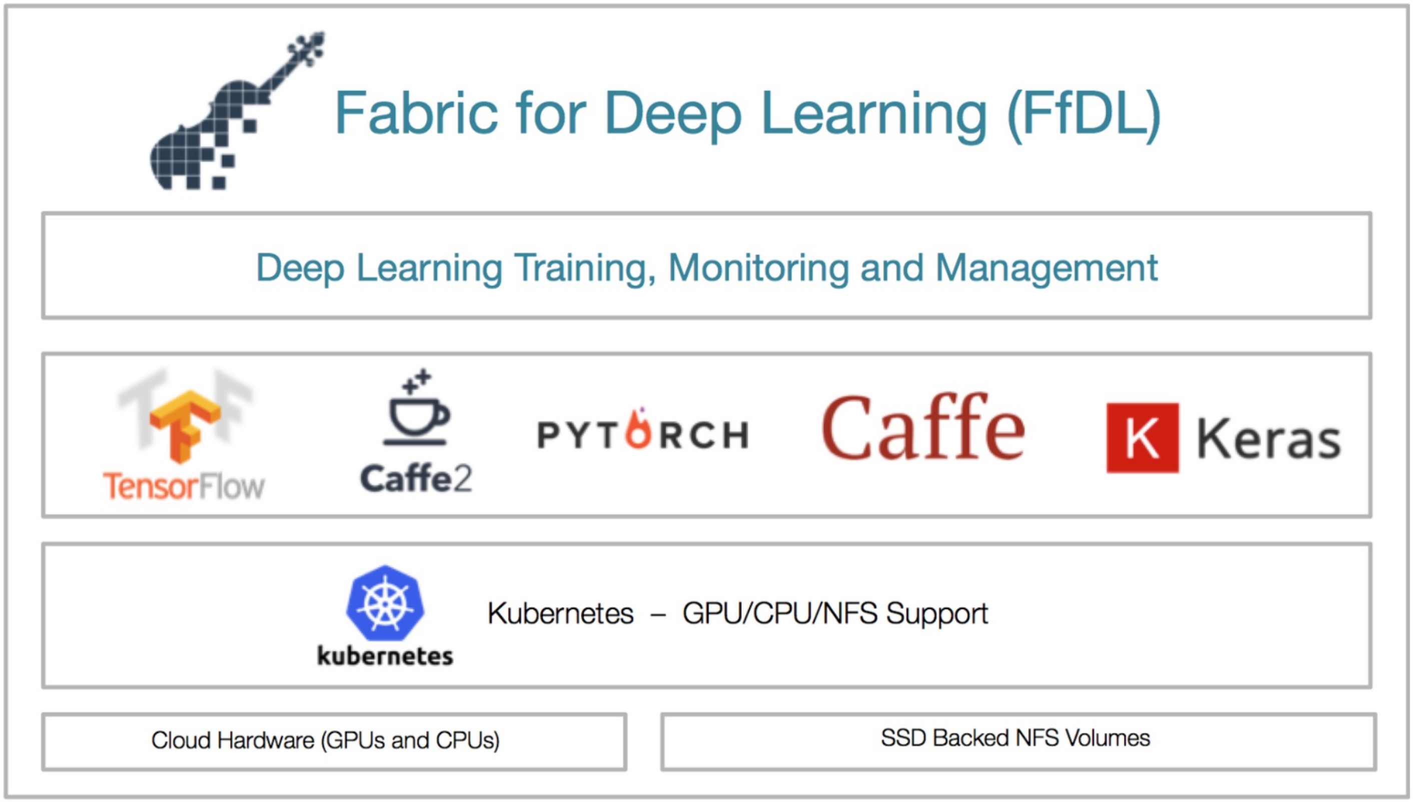
# Marriage of Cloud and AI

- AI
  - Harness power of Big Data and compute
- Cloud
  - Access to Big Data
  - Platform to quickly develop, deploy, and test AI solutions
  - Ease in AI reachability

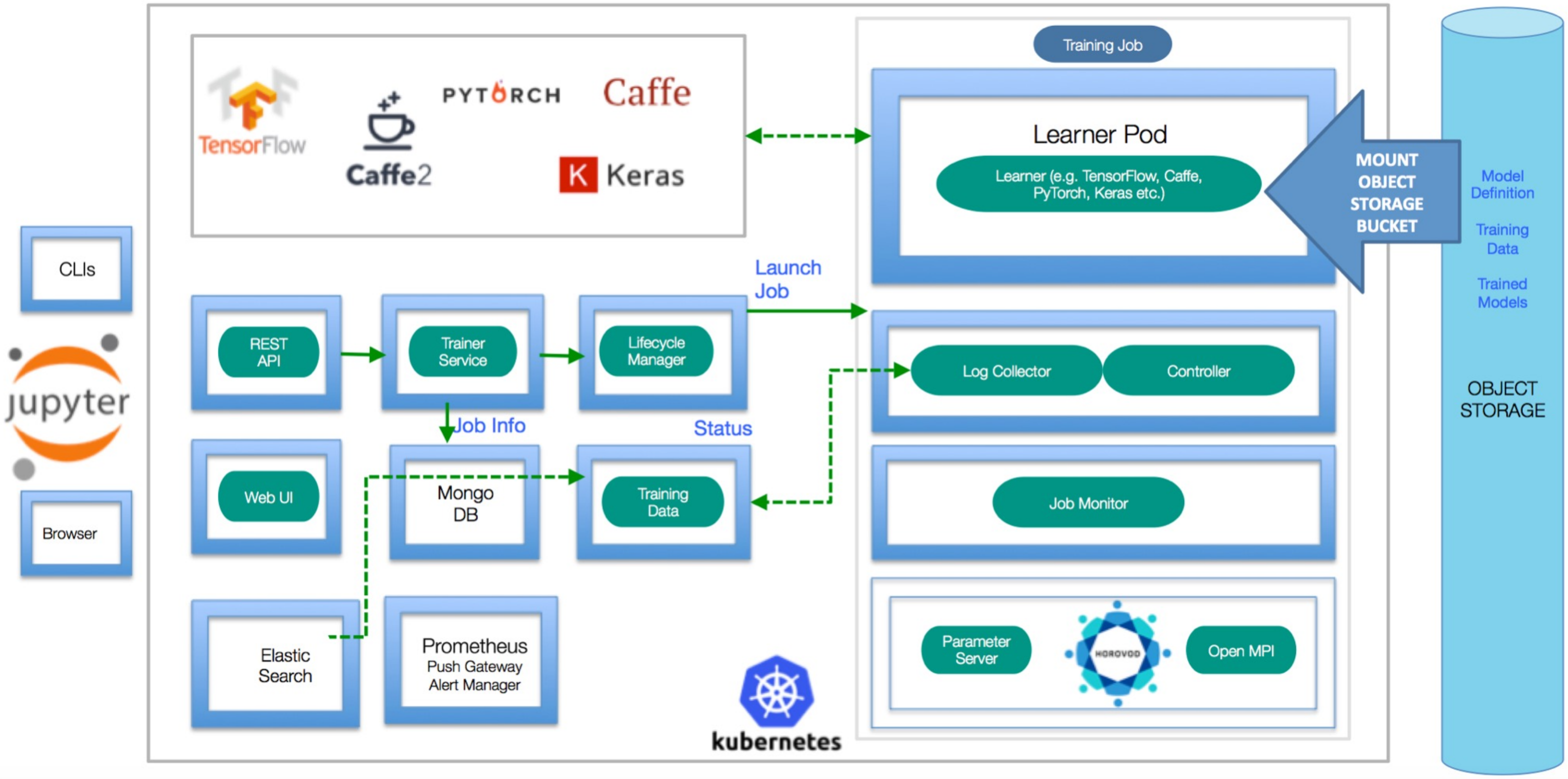
- Cloud + AI is the winning combination



# Deep Learning on Cloud Stack



# Typical DL on Cloud architecture



# Features of a Machine Learning Platform

- Frameworks: DL framework(s) supported and their version.
  - Machine learning platforms which have their own inbuilt images for different frameworks.
- Compute units: type(s) of compute units offered, i.e., GPU types.
- Model lifecycle management: tools supported to manage ML model lifecycle.
- Monitoring: availability of application logs and resource (GPU, CPU, memory) usage monitoring data to the user.
- Visualization during training: performance metrics like accuracy and throughput
- Elastic Scaling: support for elastic scaling compute resources of an ongoing job.
- Training job description: training job description file format.

# Cloud based Machine Learning Services

- IBM Watson Machine Learning

https://www.ibm.com/cloud/machine-learning

- Amazon Sagemaker

https://aws.amazon.com/sagemaker

- Microsoft Azure Machine Learning

https://azure.com/ml

- Google Cloud Machine Learning

https://cloud.google.com/ml-engine

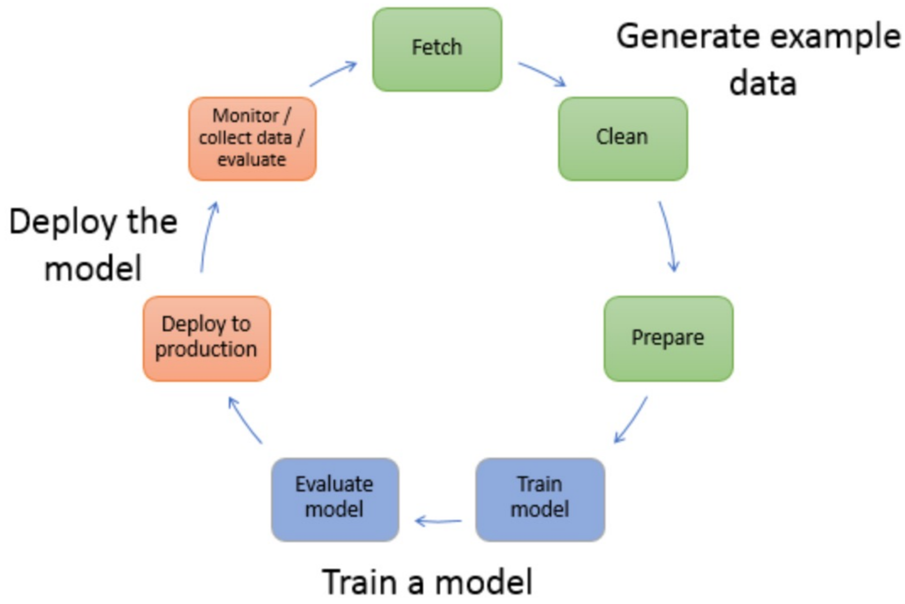
# Training job specification

- YAML file
- https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml_dlaas_e2e_example.html

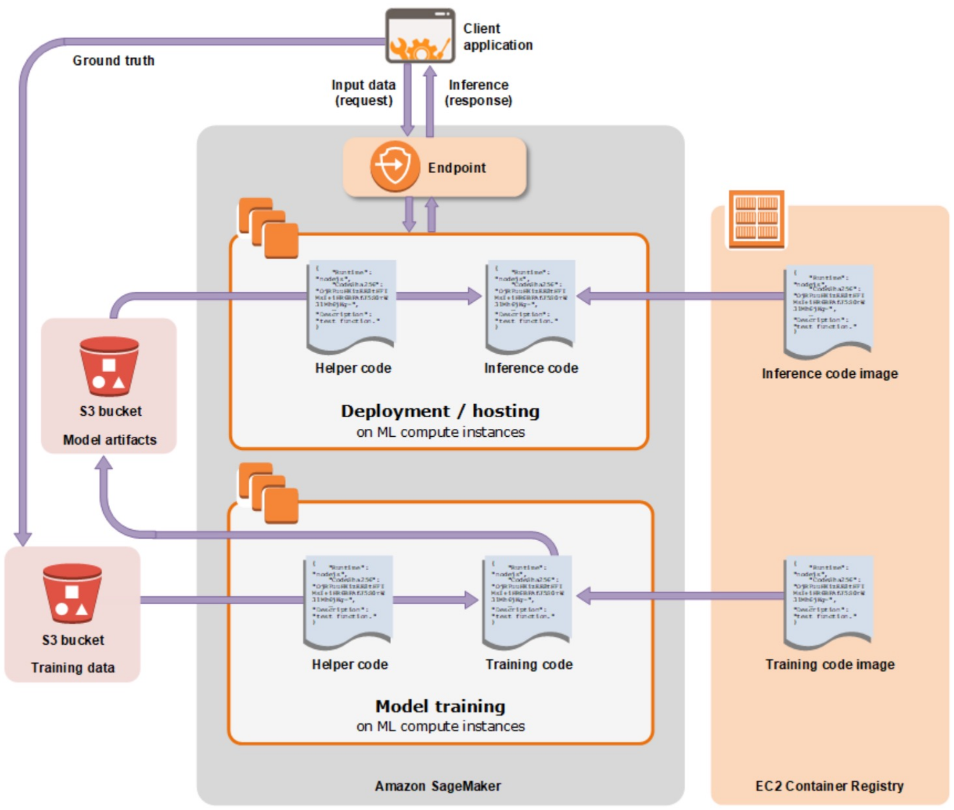
# Amazon Sagemaker

- Fully managed machine learning service by Amazon
- Supports:
  - Quick and easy building and training of ML models
  - Model deployment in production-ready hosted environment

**Typical ML Workflow**



# Train with Amazon Sagemaker




Training algorithm options

- Use an out-of-the-box algorithm provided by Amazon
- Use Apache Spark MLLib with Amazon Sagemaker
- Custom python code to train with DL frameworks
  - Tensorflow and Apache MXNet
- Use your own custom algorithm in any programming language and framework
- Use an algorithm from AWS Marketplace

https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

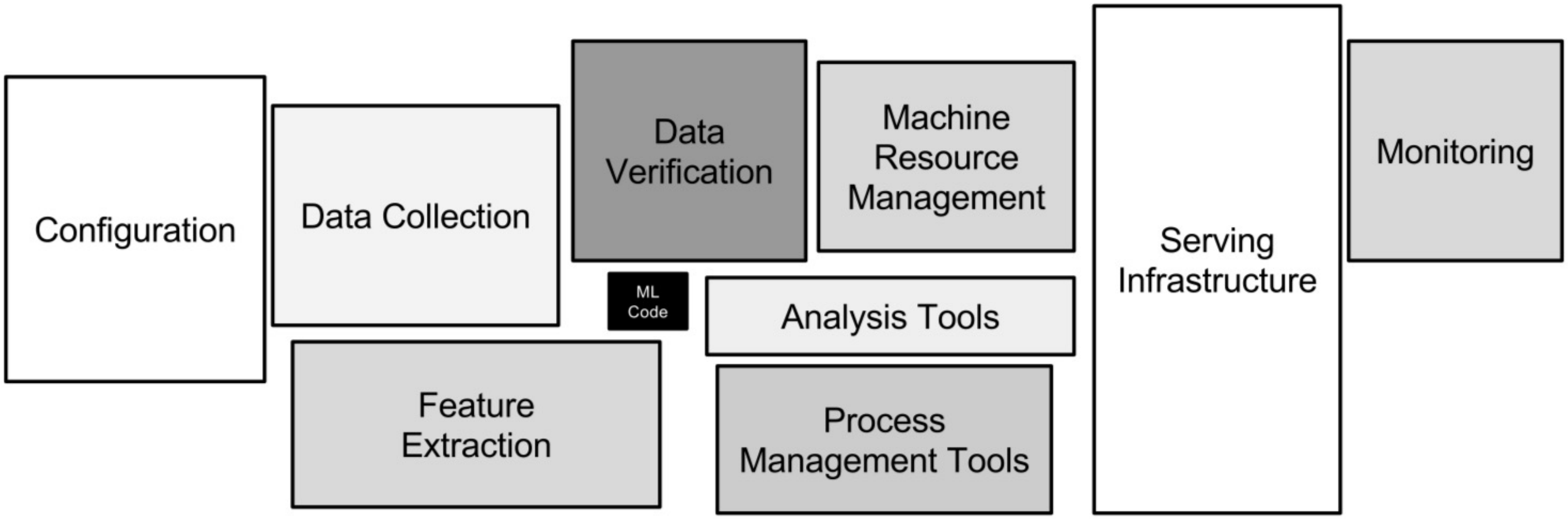
# CreateTrainingJob API for Amazon Sagemaker

- https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateTrainingJob.html

# Comparing Machine Learning Service Platforms

- https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/

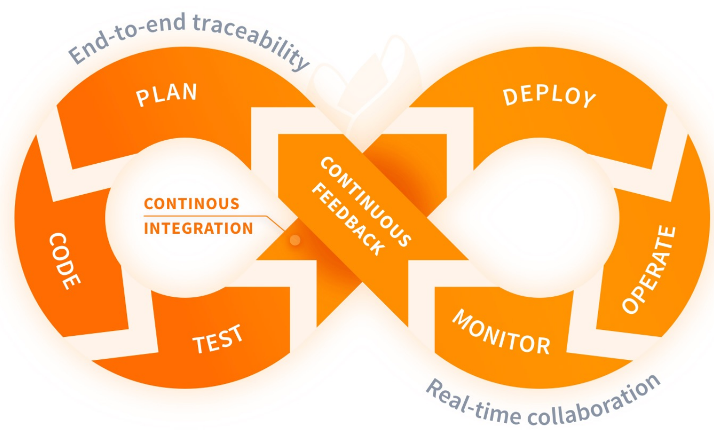
# Production Grade Machine Learning Systems




The portion of ML training code in a production-grade ML system is a lot smaller than the technologies and processes needed for supporting it.

# Software Engineering in ML Systems

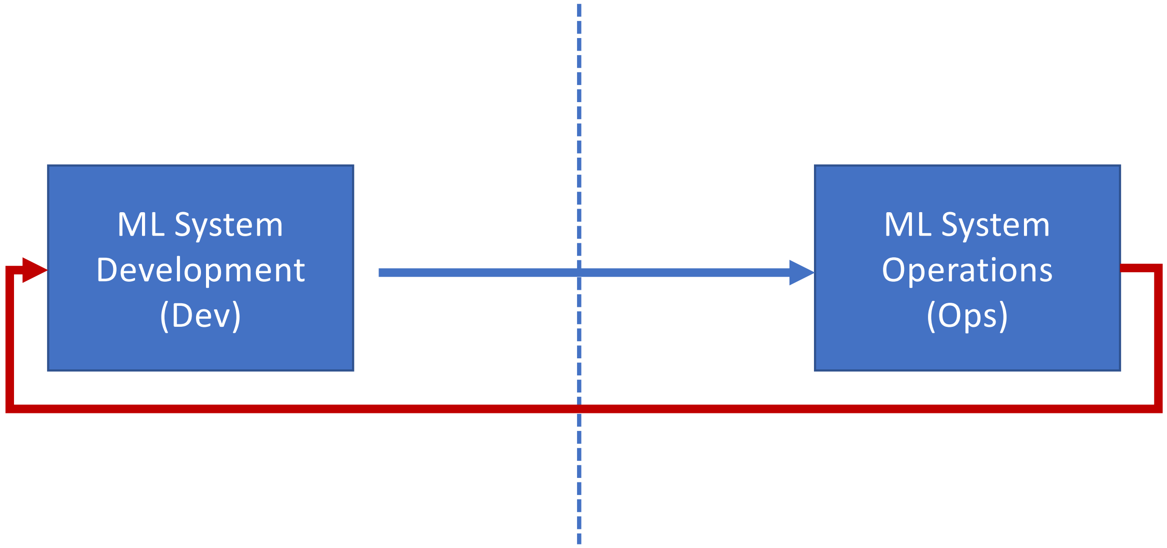
- Machine learning applications run as pipelines that ingest data, compute features, identify model(s), discover hyperparameters, train model(s), validate and deploy model(s).
- Making a model as a production-capable web service
  - Containerization (docker), cluster deployment (K8s)
  - APIs exposed as web service (Tensorflow serving/ONNX runtime)
- Workflow engines (Kubeflow) automate the ML pipeline
- Deployment monitoring and operational analytics
- Devops principles applicable to ML Systems:
  - Continuous Integration, Continuous delivery (CI/CD)
  - Predictability
    - "A model may be unexplainable—but an API cannot be unpredictable"
  - Reproducibility and Traceability
    - Provenance for Machine Learning Artifacts




ML Specific testing and monitoring apart from traditional software testing

- Data testing
- Infrastructure testing
- Model testing
- Production testing

# ModelOps: The Assembly Line for ML



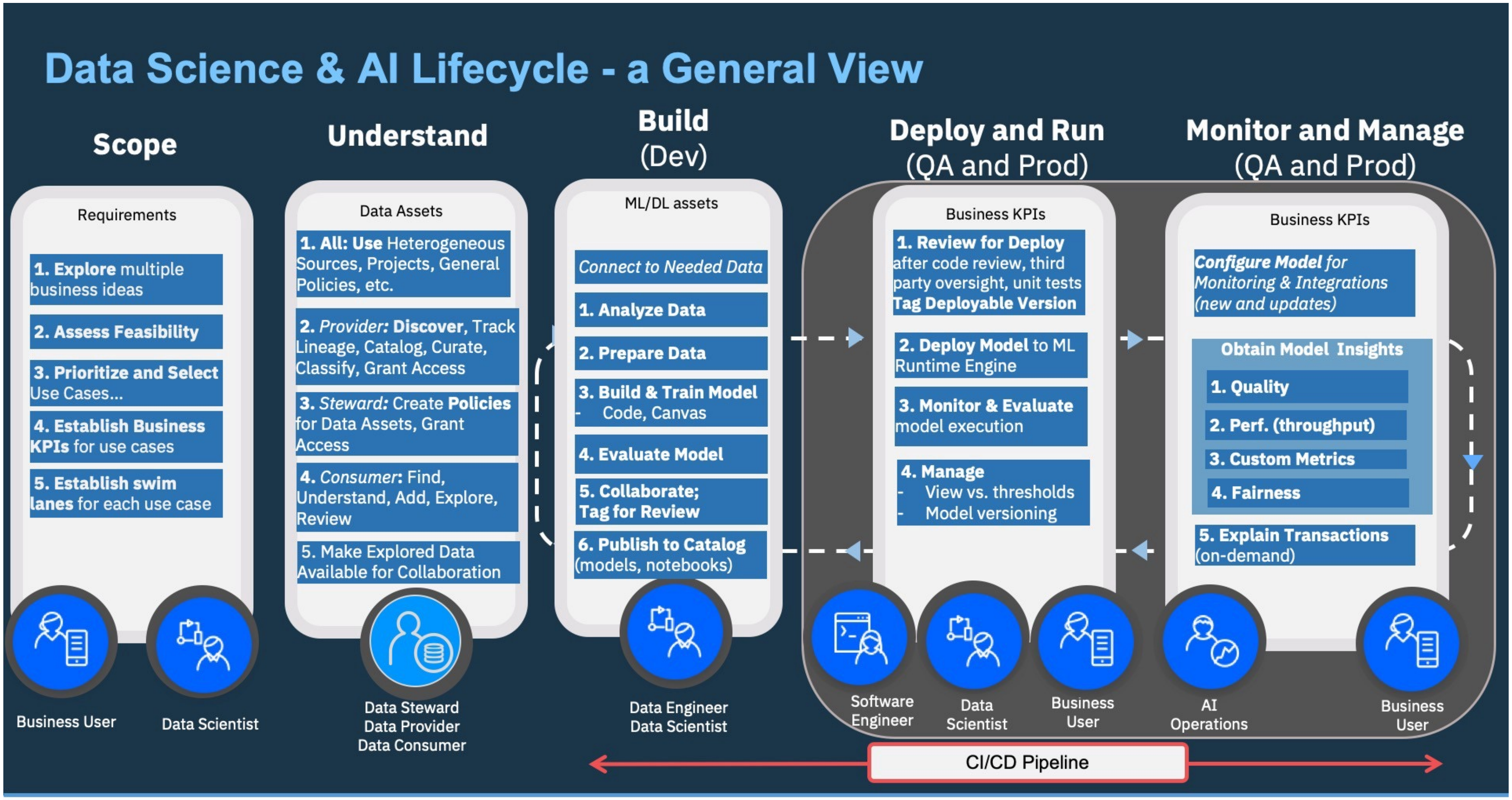

**Goal:** Accelerate model life cycle ( from development to deployment )
Maintain high quality model in production

**Approach:** automation and monitoring through development of tool-chain covering all steps of ML system construction, including development, integration, testing, releasing, deployment and infrastructure management.

**Is ModelOps same as DevOps ?**

# Operationalizing AI

ML Models → ML Systems → Deployed ML Service → Business Value




IBM

# ML specific challenges to DevOps

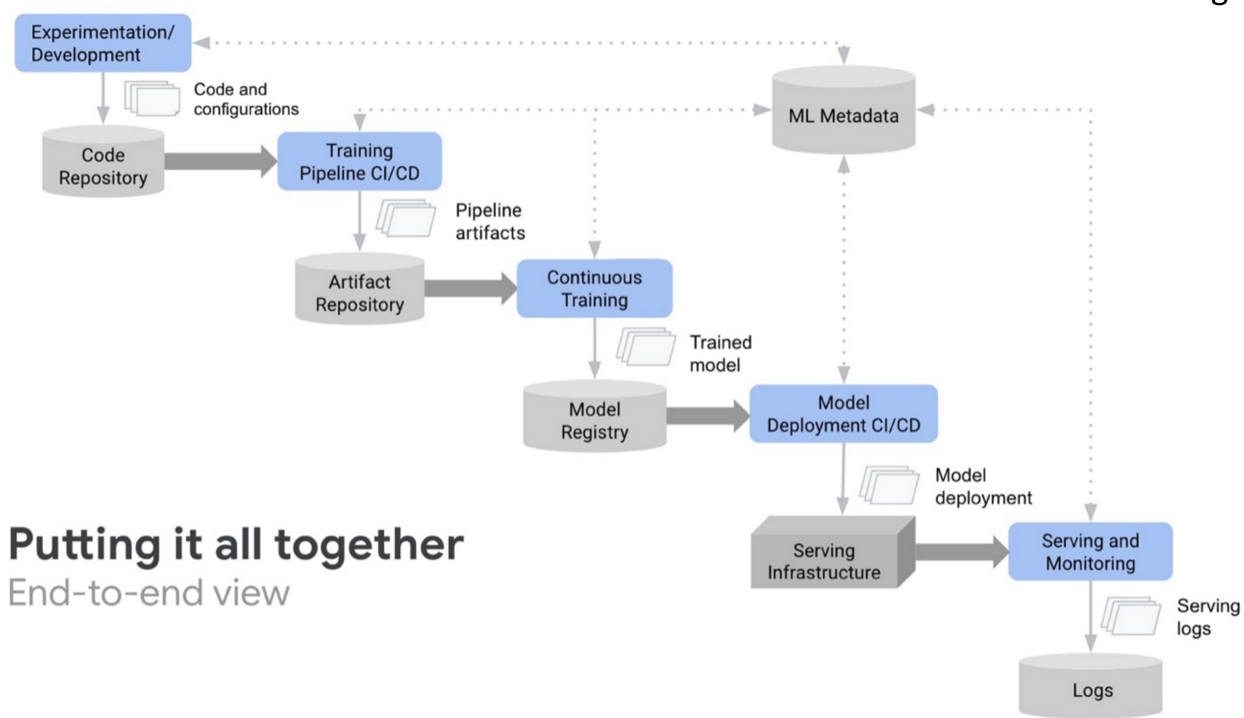
- Continuous Integration (CI) is not only about testing and validating code and components, but also testing and validating data, data schemas, and models.
- Continuous Delivery (CD) is not only about a single software package or a service, but a system (an ML training pipeline) that should automatically deploy another service (model prediction service).
- Continuous Training (CT) is a new property, unique to ML systems, that's concerned with automatically retraining candidate models for testing and serving.
- Continuous Monitoring (CM) is not only about catching errors in production systems, but also about monitoring production inference data and model performance metrics tied to business outcomes.

# Google MLOps

- *MLOps* is an ML engineering culture and practice that aims at unifying ML system development (Dev) and ML system operation (Ops)
  - An introduction to MLOps on Google Cloud

# MLOps with CI/CD

Repeatable and reliable pipelines
Lineage tracking of trained models



# IBM Cloud Pak for Data for ML Operationalization

[Managing AI Lifecycle with MLOps](#)