End to End Machine Learning Workflows

Animesh Singh
@AnimeshSingh
STSM and Chief Architect
Data and AI Open Source Platform

IBM **Developer**

IBM

Center for Open Source Data and AI Technologies (CODAIT)

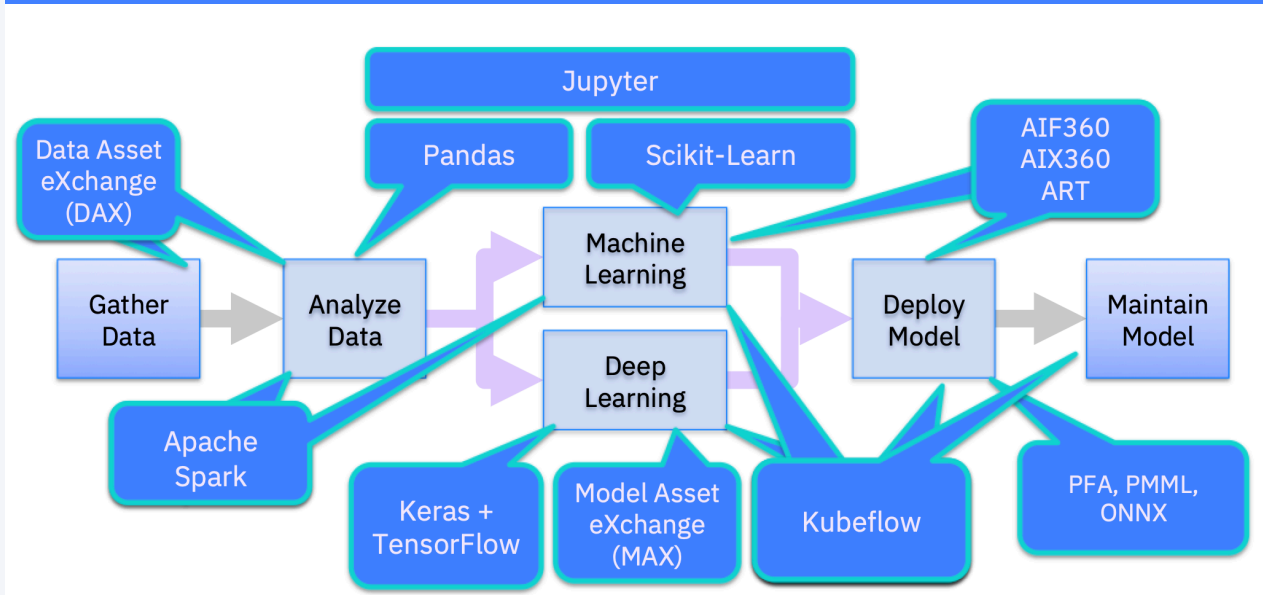
Code – Build and improve practical frameworks to enable more developers to realize immediate value.

Content – Showcase solutions for complex and real-world AI problems.

Community – Bring developers and data scientists to engage with IBM

- Team contributes to over **10 open source projects**
- **17 committers** and many contributors in Apache projects
- Over **1100 JIRAs** and **66,000 lines of code** committed to Apache Spark itself; over **65,000 LoC into SystemML**
- Over **25 product lines** within IBM leveraging Apache Spark
- Speakers at over 100 conferences, meetups, unconferences and more

Improving Enterprise AI lifecycle in Open Source

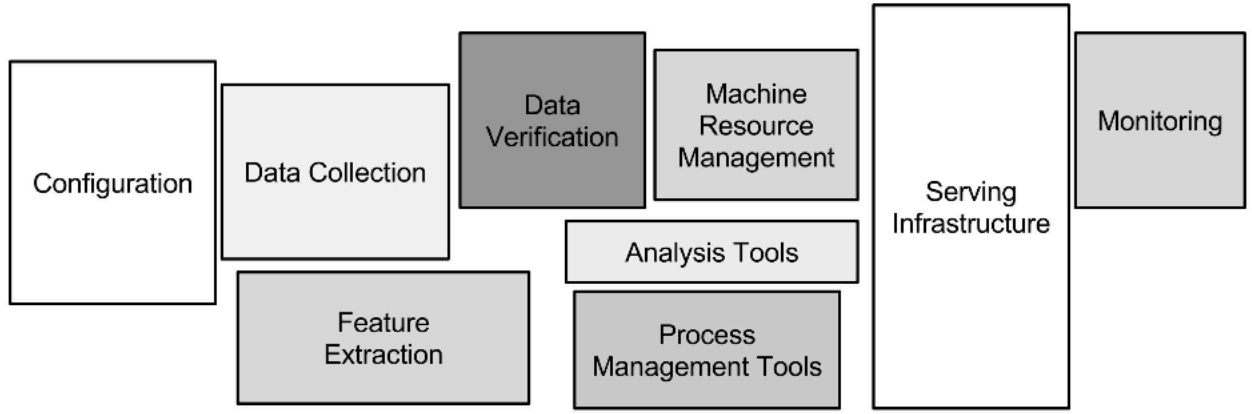


CODAIT

codait.org

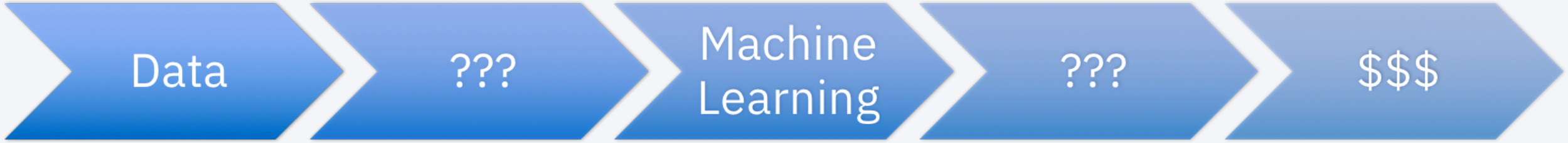
The Machine Learning Workflow

Perception

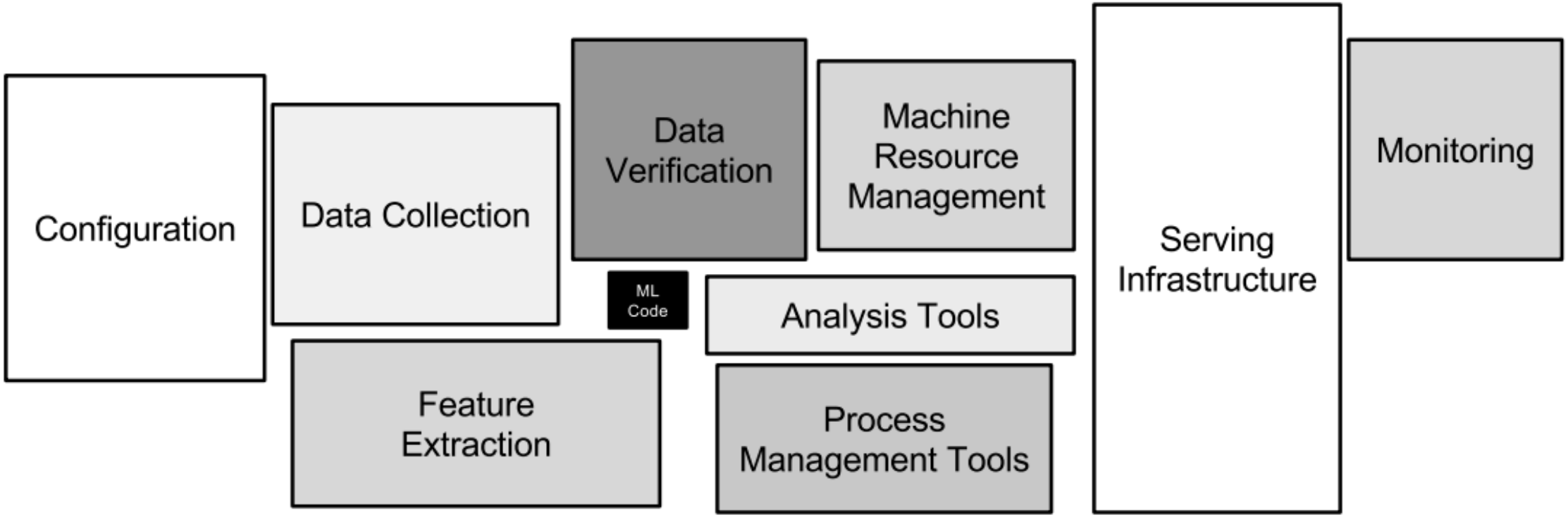


*Source: Hidden Technical Debt in Machine Learning Systems

Perception

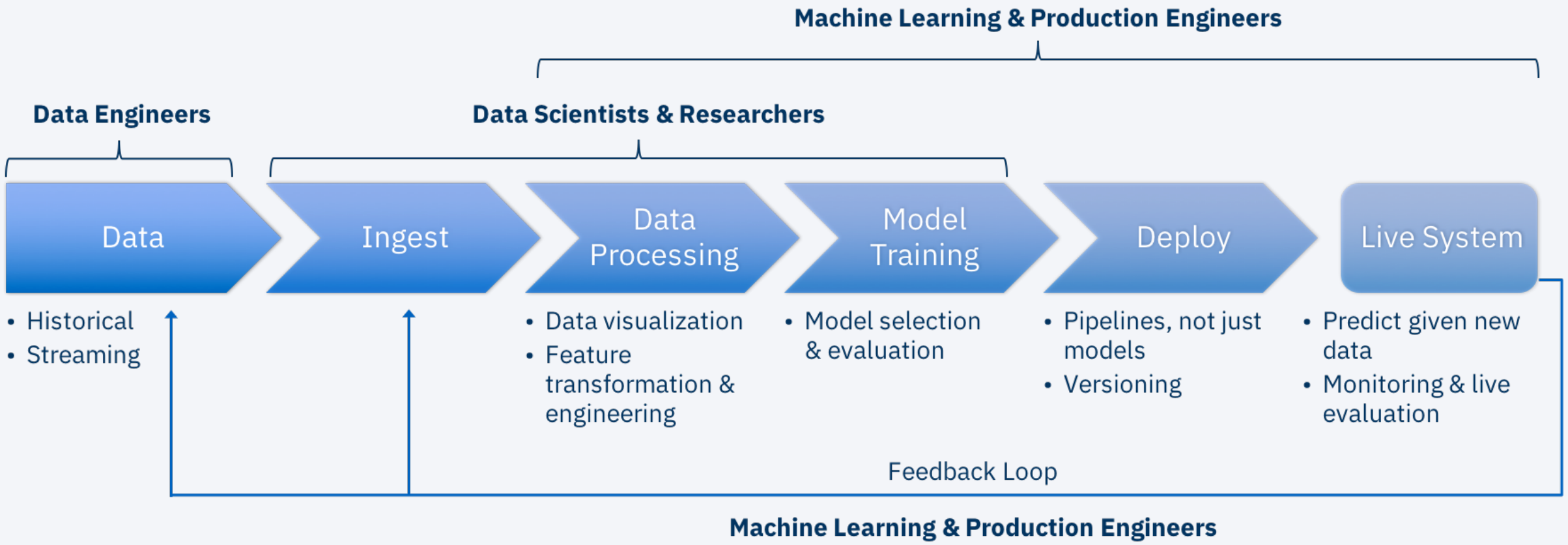


In reality…ML Code is tiny part in this overall platform

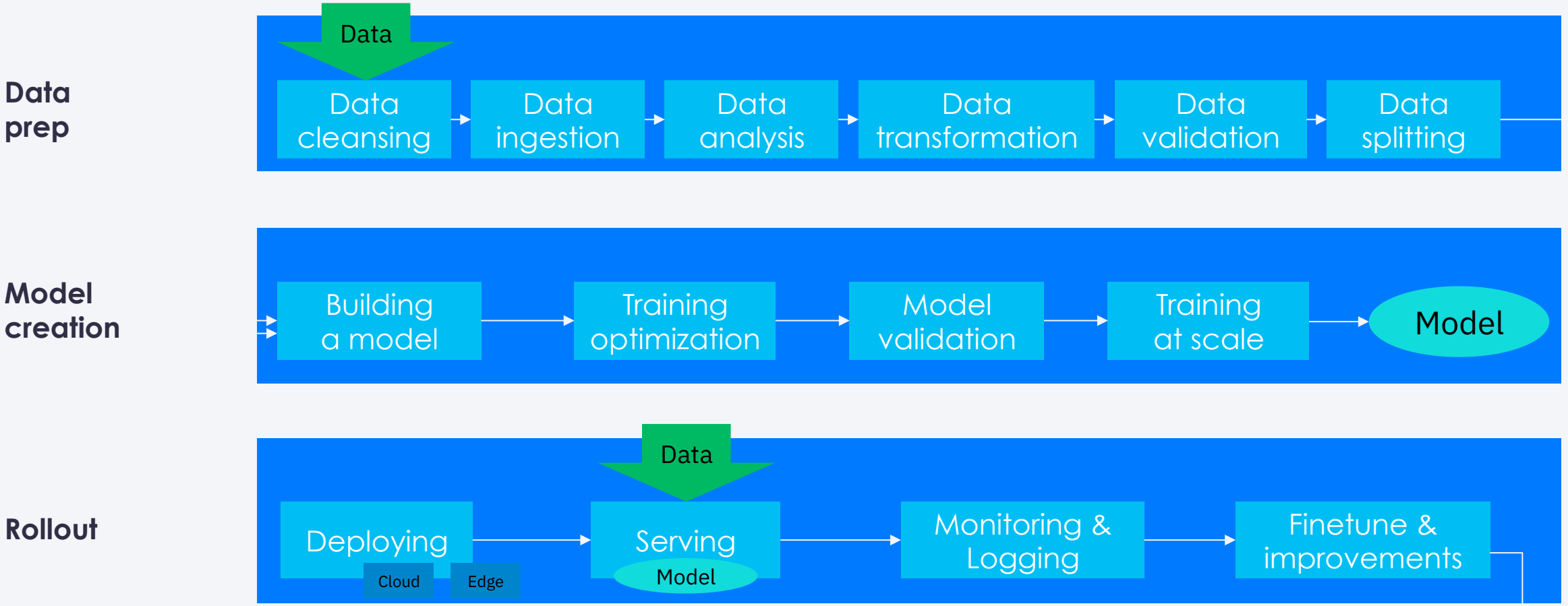


*Source: Hidden Technical Debt in Machine Learning Systems

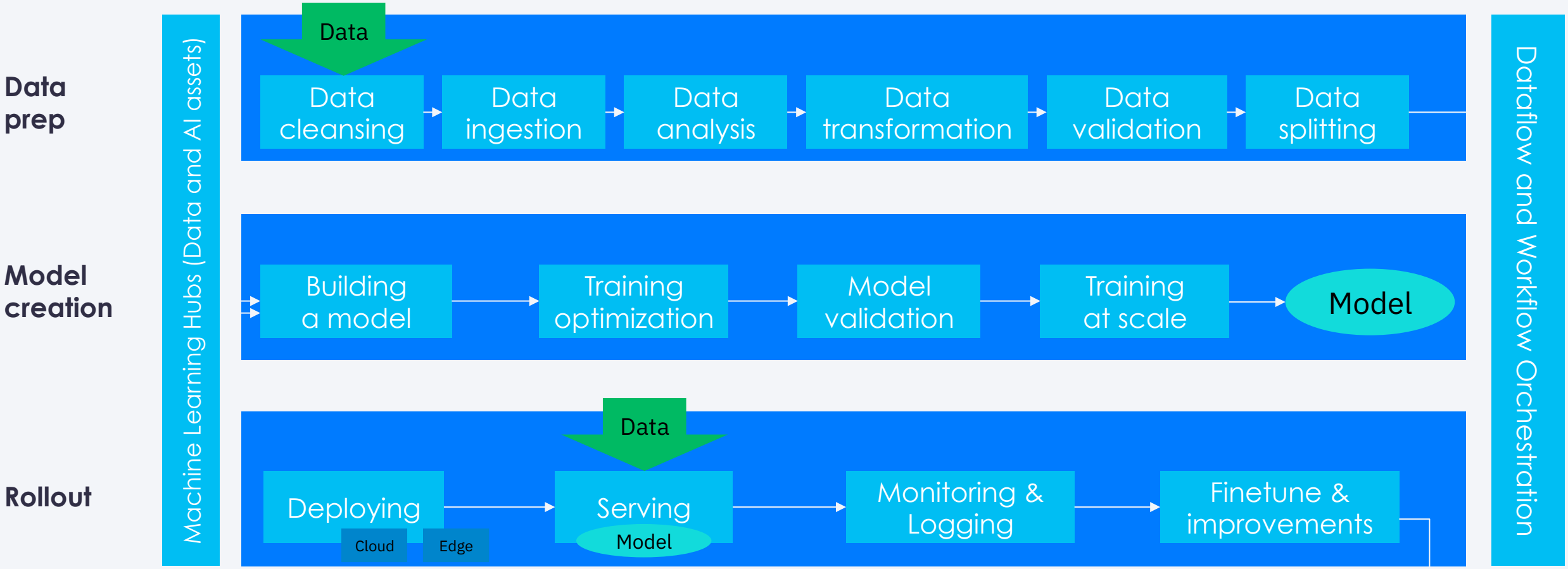
And the ML workflow spans teams …



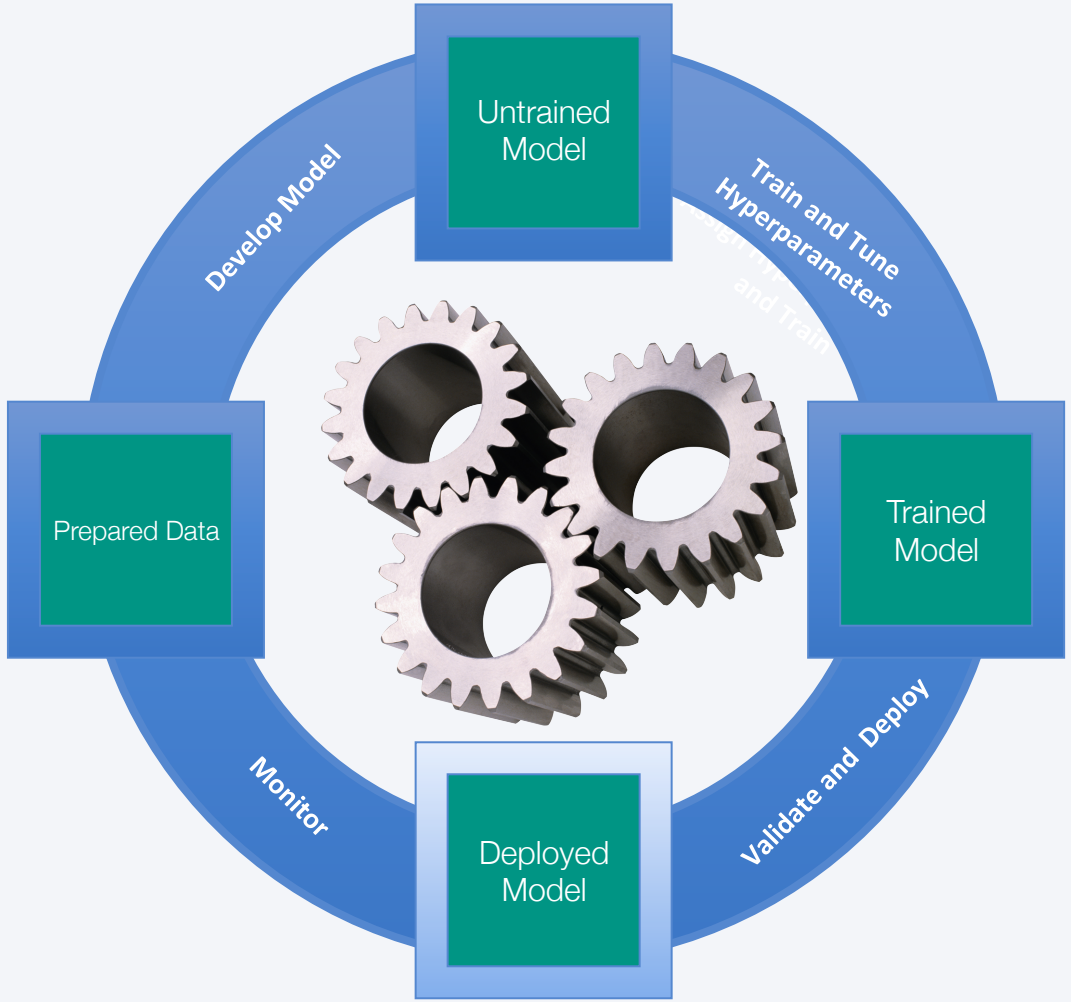
…and is much more complex….



And needs a workflow orchestrator…



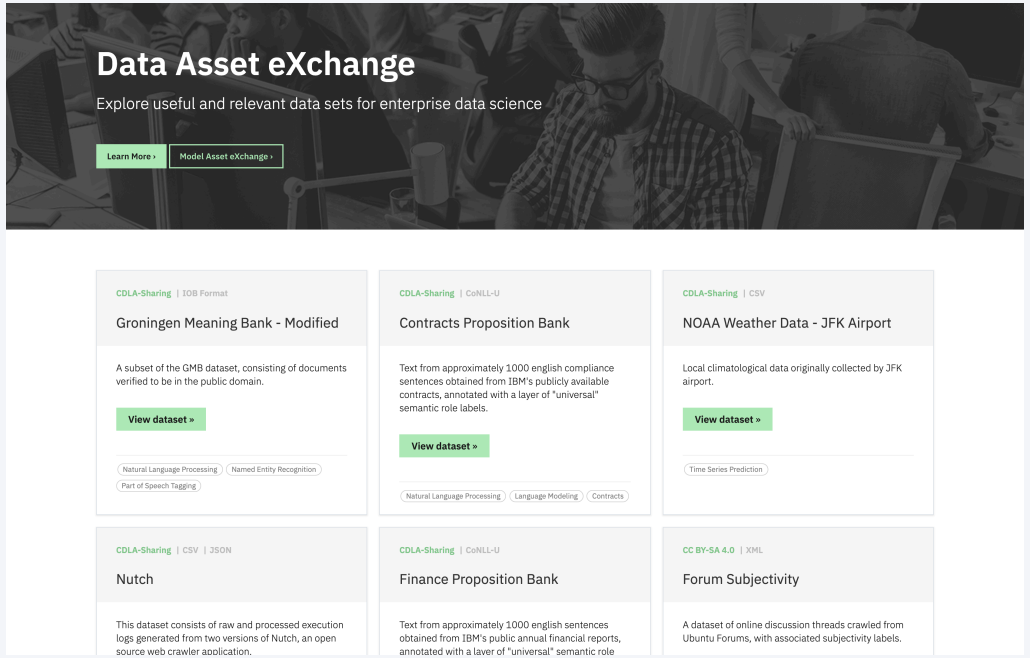
Let`s go through different phases of AI Lifecycle through our use cases…
For ML Lifecycle, starting with a quality data set is a must....

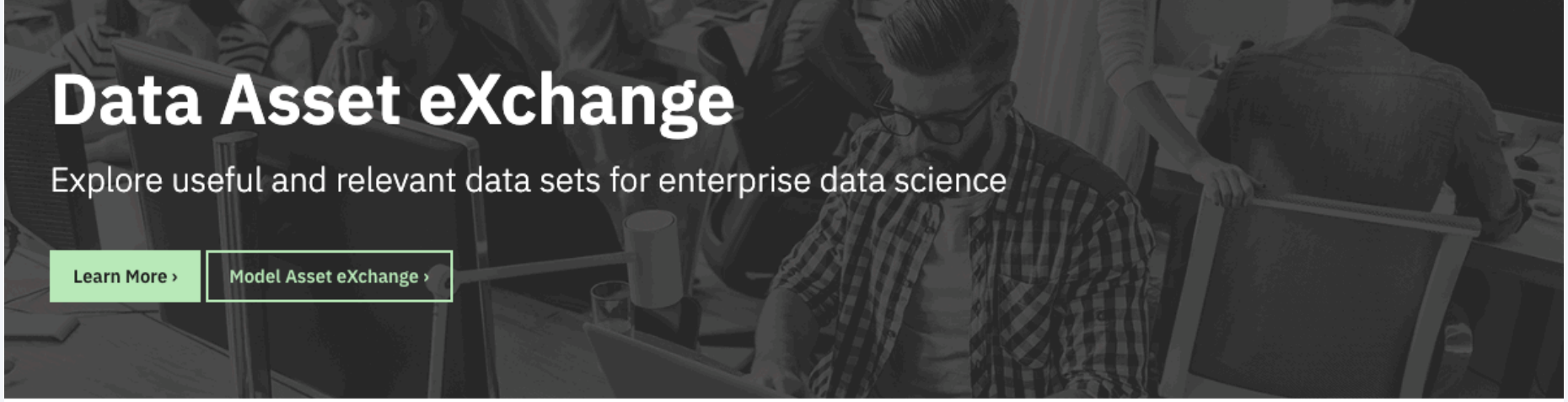


IBM Data Asset eXchange (DAX)

- Curated free and open datasets under open data licenses
- Standardized dataset formats and metadata
- Ready for use in enterprise AI applications

Data Asset eXchange
ibm.biz/data-asset-exchange





CDLA-Sharing | CoNLL-U

Contracts Proposition Bank

Text from approximately 1000 english compliance sentences obtained from IBM's publicly available contracts, annotated with a layer of "universal" semantic role labels.

View dataset »

Natural Language Processing
Language Modeling
Contracts

CDLA-Sharing | CSV

NOAA Weather Data - JFK Airport

Local climatological data originally collected by JFK airport.

View dataset »

Time Series Prediction

CDLA-Sharing | CSV | H.264

Double Pendulum Chaotic

Videos of the chaotic motion of a double pendulum, in addition to the positions of the datums of the pendulum.

View dataset »

Time Series
Video Prediction

CDLA-Sharing | CSV | JSON

Nutch

CDLA-Sharing | CoNLL-U

Finance Proposition Bank

CC BY-SA 4.0 | XML

Forum Classify

CDLA-Sharing | CoNLL-U

Contracts Proposition Bank

Text from approximately 1000 english compliance sentences obtained from IBM's publicly available contracts, annotated with a layer of "universal" semantic role labels.

View dataset »

Natural Language Processing
Language Modeling
Contracts

CDLA-Sharing | CSV | JSON

Nutch

CDLA-Sharing | CoNLL-U

Contracts Proposition Bank

Text from approximately 1000 english compliance sentences obtained from IBM's publicly available contracts, annotated with a layer of "universal" semantic role labels.

Get this dataset

By Sanjana Sahayaraj, Yunyao Li, Huaiyu Zhu, Marina Danilevsky, Poornima Chozhiyath Raman, Ramiya Venkatachalam | Published July 16, 2019

Natural Language Processing | Language Modeling | Contracts

Overview

ConProp version 1.0 was developed by researchers at IBM Almaden Research Center, San Jose, CA, USA. It consists of proposition bank-style annotations from approximately 1000 english compliance sentences obtained from IBM's publicly available contracts. These sentences were extracted from contract sections such as Business Partner descriptions, Agreement Terms / Structure, Intellectual Property Protection, Limitation of Liability, Warranty Terms, General Principles of Relationship, Terms of Agreement Termination, Withdrawal of Service, Third Party Claims, Charges, Service Level Agreement and many more.

To suit the need in compliance domain, a list of 60 predicates specific to the domain were

CONTENTS

Overview
Dataset Metadata
Example Records
Citation

RESOURCES

Data Asset eXchange (DAX)
Explore useful and relevant data sets for enterprise data science.

Model Asset eXchange (MAX)
A place for developers to find and use free and open source deep learning models.

Center for Open-Source Data & AI Technologies (CODAIT)
Improving the Enterprise AI Lifecycle in Open Source.

CDLA-Sharing | CoNLL-U

Contracts Proposition Bank

Text from approximately 1000 english compliance sentences obtained from IBM's publicly available contracts, annotated with a layer of "universal" semantic role labels.

Get this dataset

IBM Cloud Object Storage

CDLA-Sharing | CoNLL-U

Contracts Proposition Bank

Text from approximately 1000 english compliance sentences obtained from IBM's publicly available contracts, annotated with a layer of "universal" semantic role labels.

View dataset »

Natural Language Processing
Language Modeling
Contracts

CDLA-Sharing | CSV | JSON

Nutch

By Sanjana Sahayaraj, Yunyao Li, Huaiyu Zhu, Marina Danilevsky, Poornima Chozhiyath Raman, Ramiya Venkatachalam | Published July 16, 2019

Natural Language Processing
Language Modeling
Contracts

Overview

ConProp version 1.0 was developed by researchers at IBM Almaden Research Center, San Jose, CA, USA. It consists of proposition bank-style annotations from approximately 1000 english compliance sentences obtained from IBM's publicly available contracts. These sentences were extracted from contract sections such as Business Partner descriptions, Agreement Terms / Structure, Intellectual Property Protection, Limitation of Liability, Warranty Terms, General Principles of Relationship, Terms of Agreement Termination, Withdrawal of Service, Third Party Claims, Charges, Service Level Agreement and many more.

To suit the need in compliance domain, a list of 60 predicates specific to the domain were

CONTENTS

Overview
Dataset Metadata
Example Records
Citation

RESOURCES

Data Asset eXchange (DAX)
Explore useful and relevant data sets for enterprise data science.

Model Asset eXchange (MAX)
A place for developers to find and use free and open source deep learning models.

Center for Open-Source Data & AI Technologies (CODAIT)
Improving the Enterprise AI Lifecycle in Open Source.

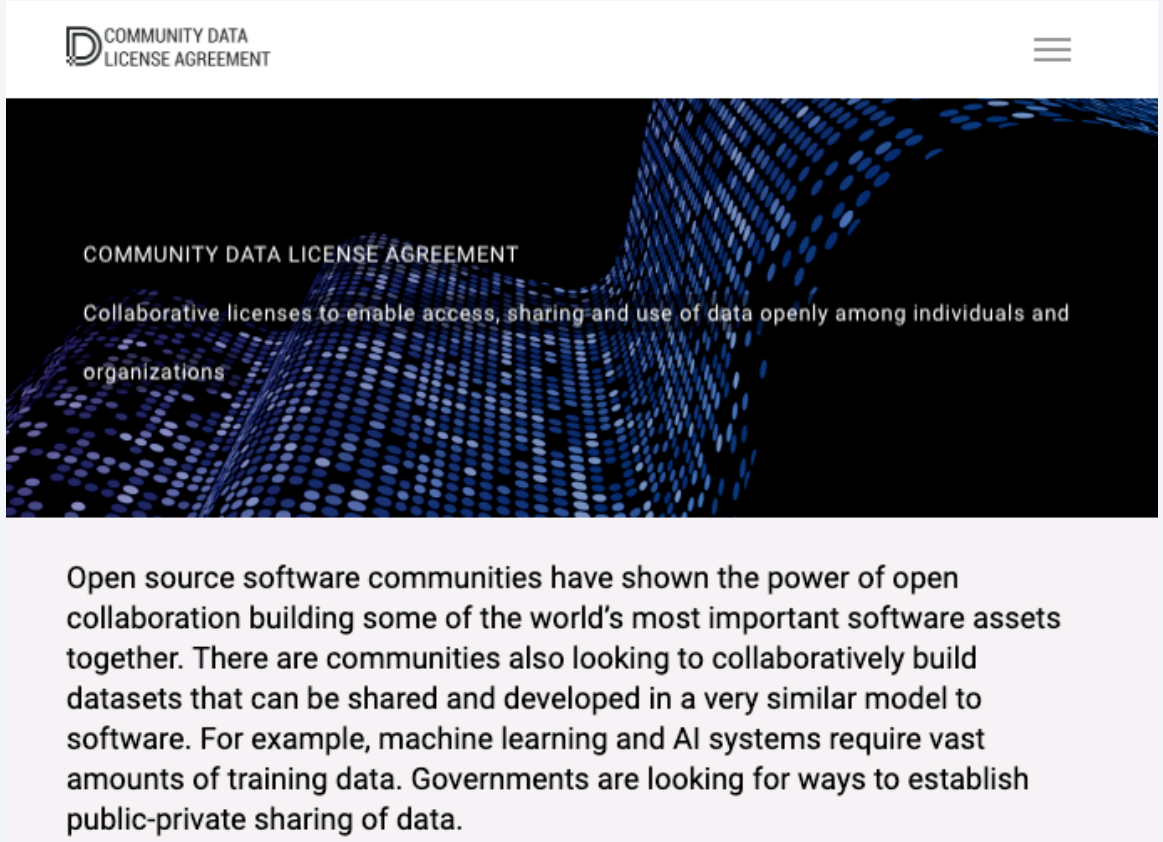
The Community Data License Agreement, driven by Linux Foundation

http://cdla.io

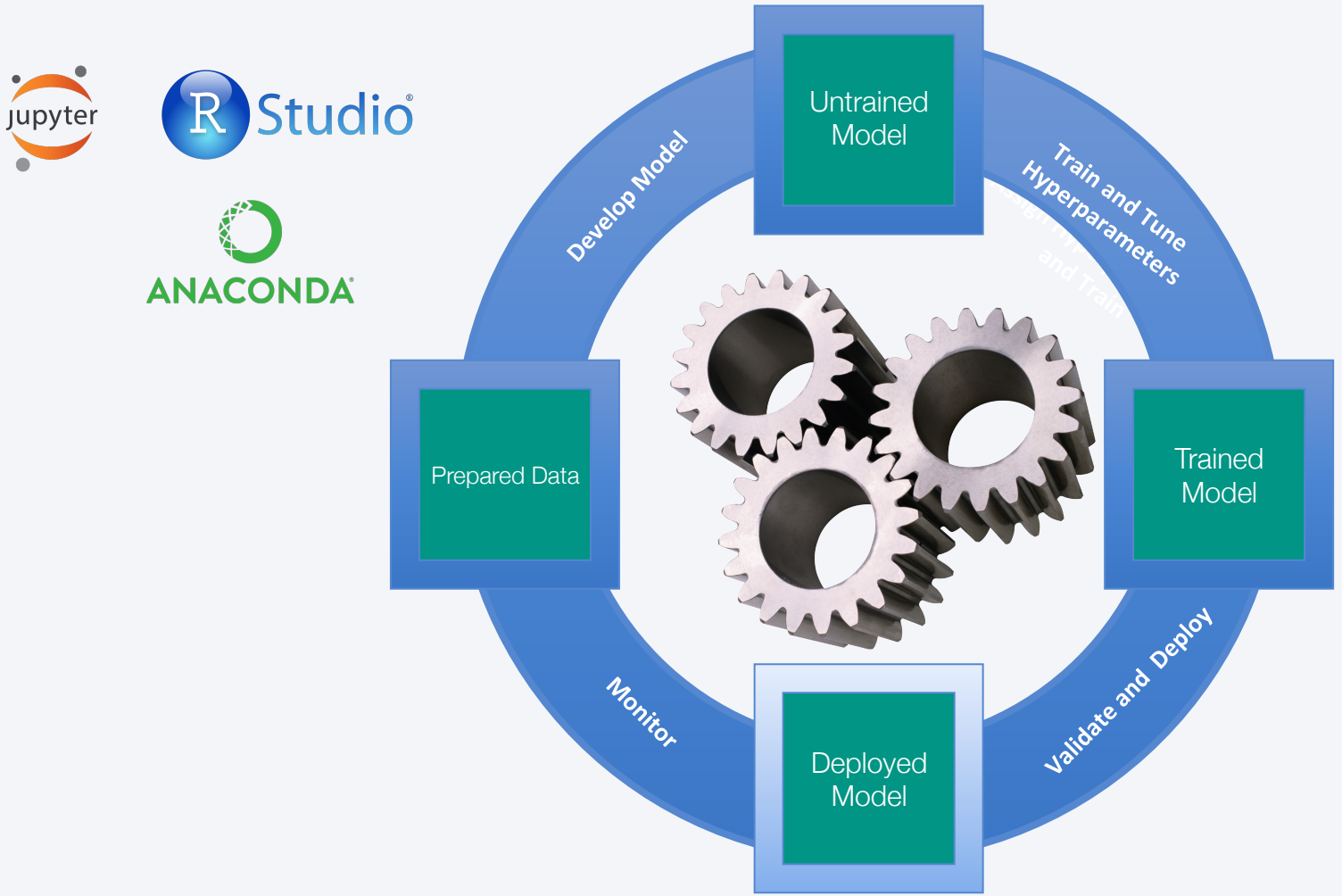
Two licenses written specifically for AI data

- CDLA-Sharing: “Copyleft” license analogous the GPL
- CDLA-Permissive: Similar to BSD license

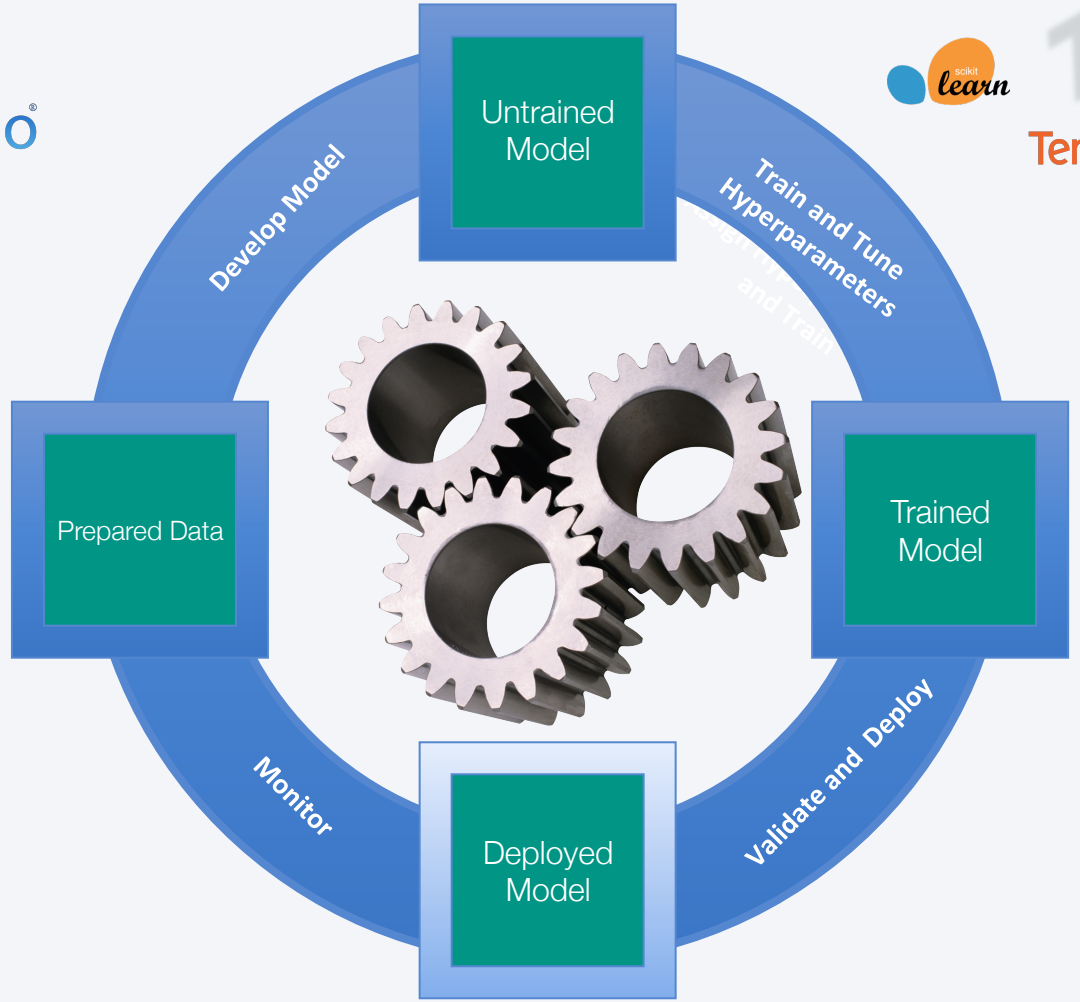
Both licenses distinguish clearly between *use* (analysis, modeling) and *modification* of the data set.



Many tools available to build initial models



Many tools to train machine learning and deep learning models



jupyter

RStudio

ANACONDA

scikit learn

TensorFlow

PaddlePaddle

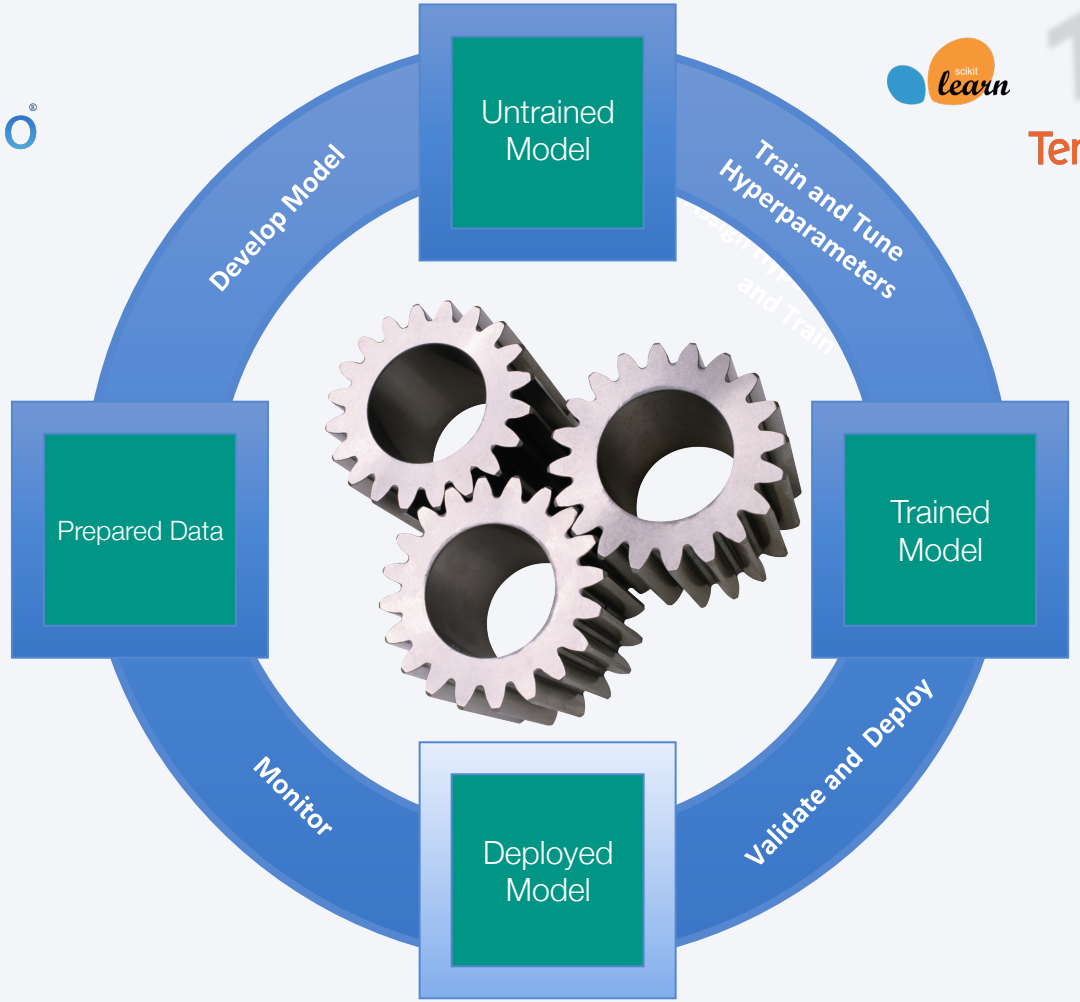
PyTorch

Keras

dmlc XGBoost

kubernetes

We need a multi framework ML platform. And it should be cloud native… but why?



e.g. Tensorflow is awesome , but had static graphs so PyTorch's dynamic graphs became more popular.

e.g. Tensorflow Parameter server approach is great but Horovod ring-all reduce algorithm performs better?

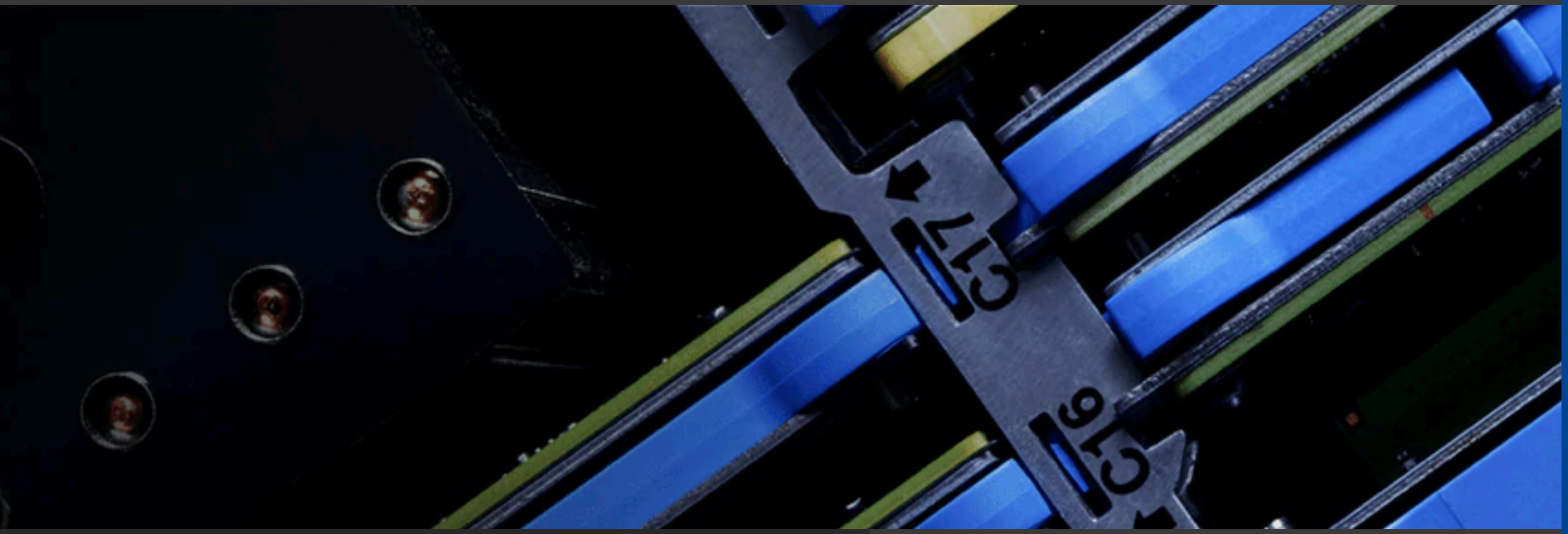
1.How do we handle multiple Open Source Machine Learning and Deep Learning frameworks in a consistent way?

2. Do we need the power of cloud for distributed computing?

ML requires the strength of HPC & GPUs

Ability to scale ML workloads on demand

Ability to utilize various technologies and achieve high performance computing.



1. Model/Data Parallelism

2. MPI/NCCL

NCCL (pronounced "Nickel") is a stand-alone library of standard collective communication routines for GPUs, implementing all-reduce, all-gather, reduce, broadcast, and reduce-scatter. It has been optimized to achieve high bandwidth on platforms using PCIe, NVLink, NVswitch, as well as networking using InfiniBand Verbs or TCP/IP sockets.

NCCL supports an arbitrary number of GPUs installed in a single node or across multiple nodes, and can be used in either single- or multi-process (e.g., MPI) applications.

To scale we need to go Cloud native for AI….we need to go to Kubernetes

Microservices

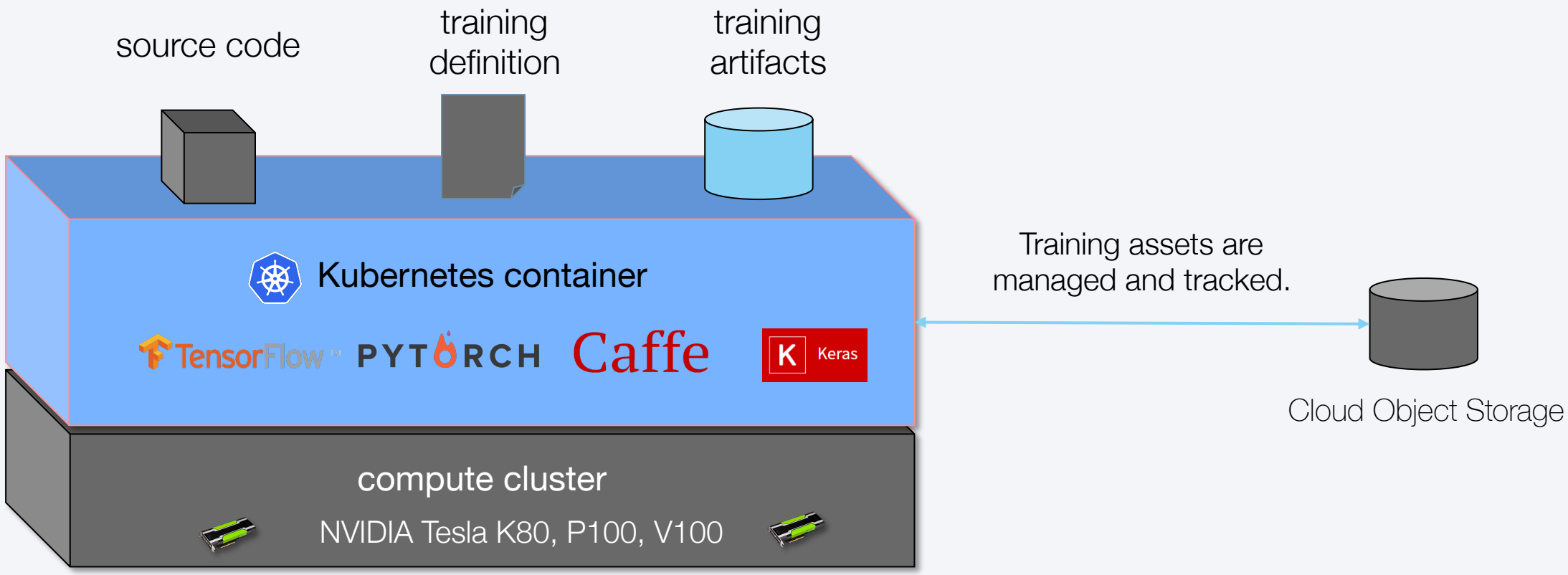
Containers

DevOps automation

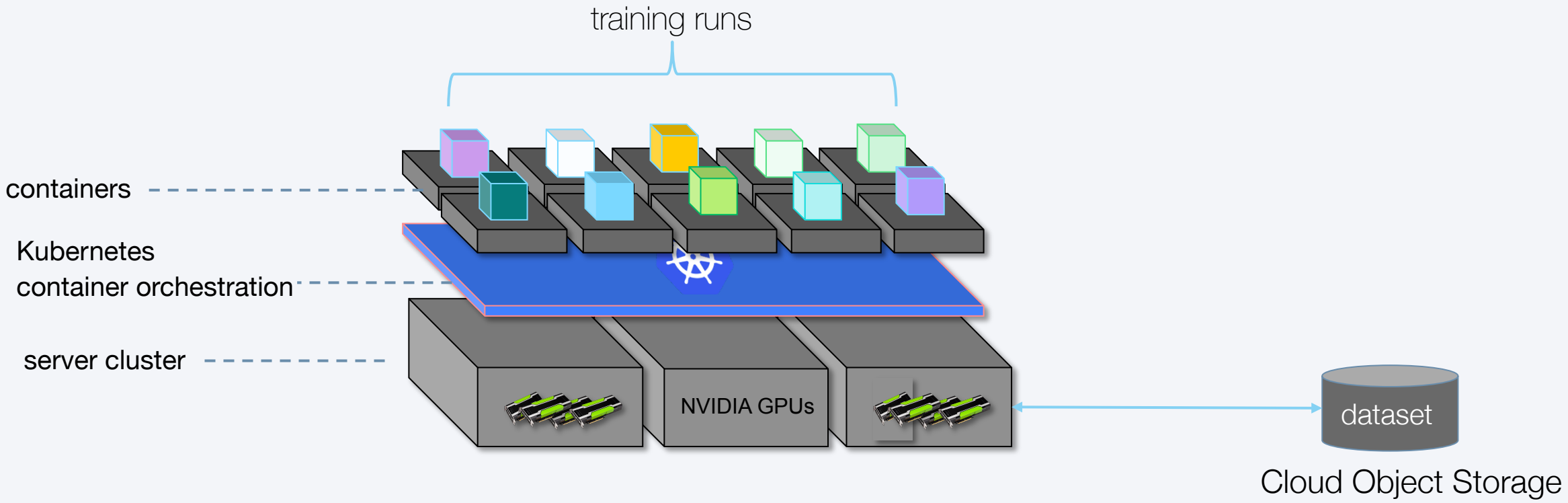
IBM CODE

Access to elastic compute leveraging Kubernetes

Auto-allocation means infrastructure is used only when needed

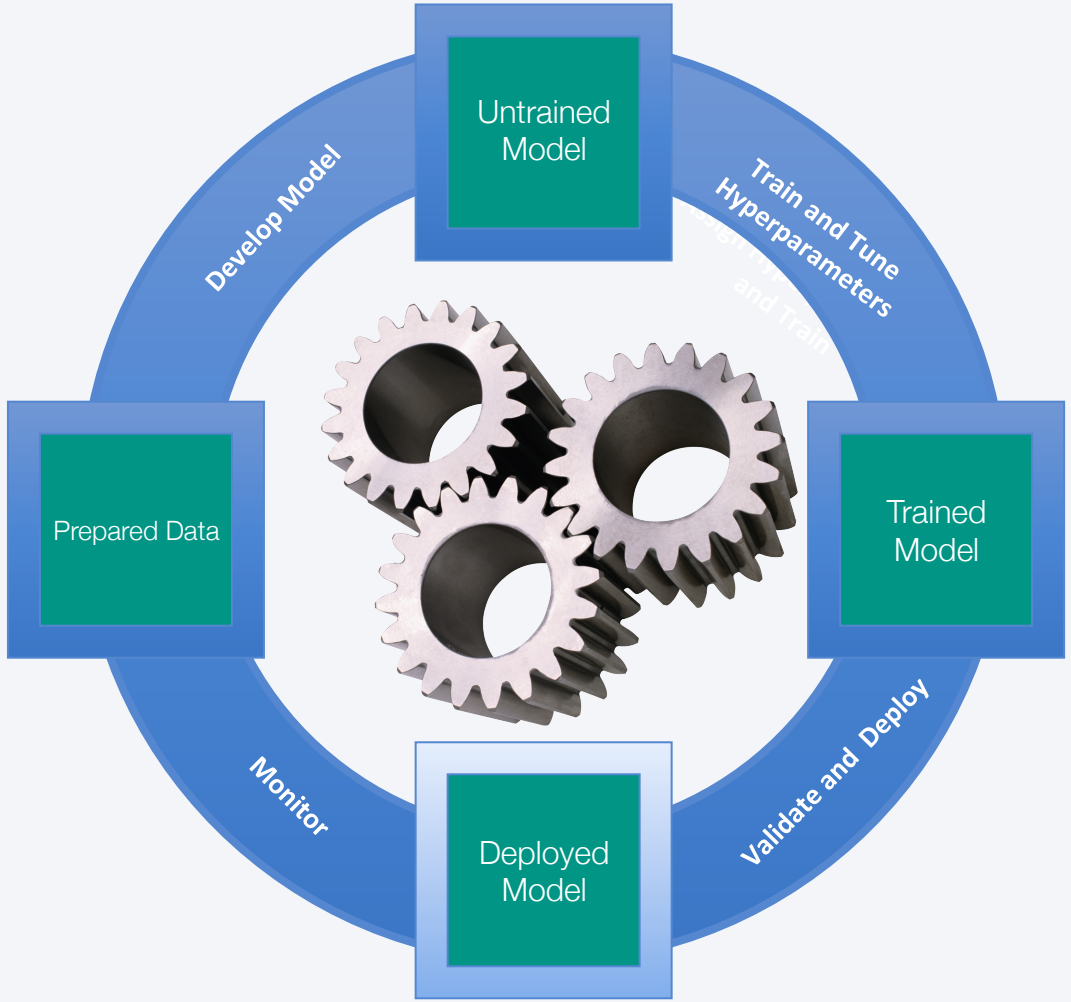


Model training distributed across containers



So we need Kubernetes, but Kubernetes is not the end game

We need a platform. Enter Kubeflow



Kubeflow

Kubeflow

https://github.com/kubeflow

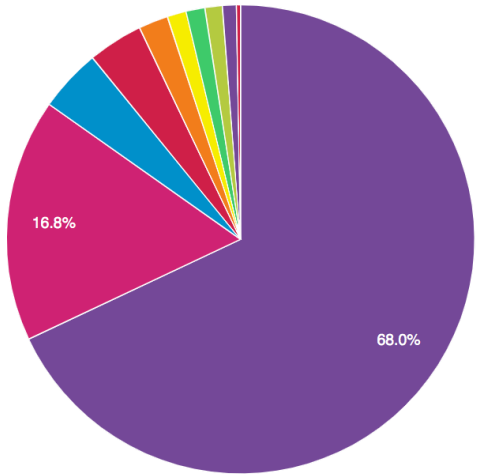
- End to end ML Platform on Kubernetes. Focused on multiple aspects of Model Lifecycle
- Originated at Google, and has grown to have a large community of developers
- Google, IBM, Cisco, RedHat, Intel, Microsoft and others contributing
- IBM is the 2nd largest contributor in terms of overall commits. IBM maintainers (committers/reviewers) in Katib (HPO+Training), Kubeflow Serving, Manifests, Pipelines etc.

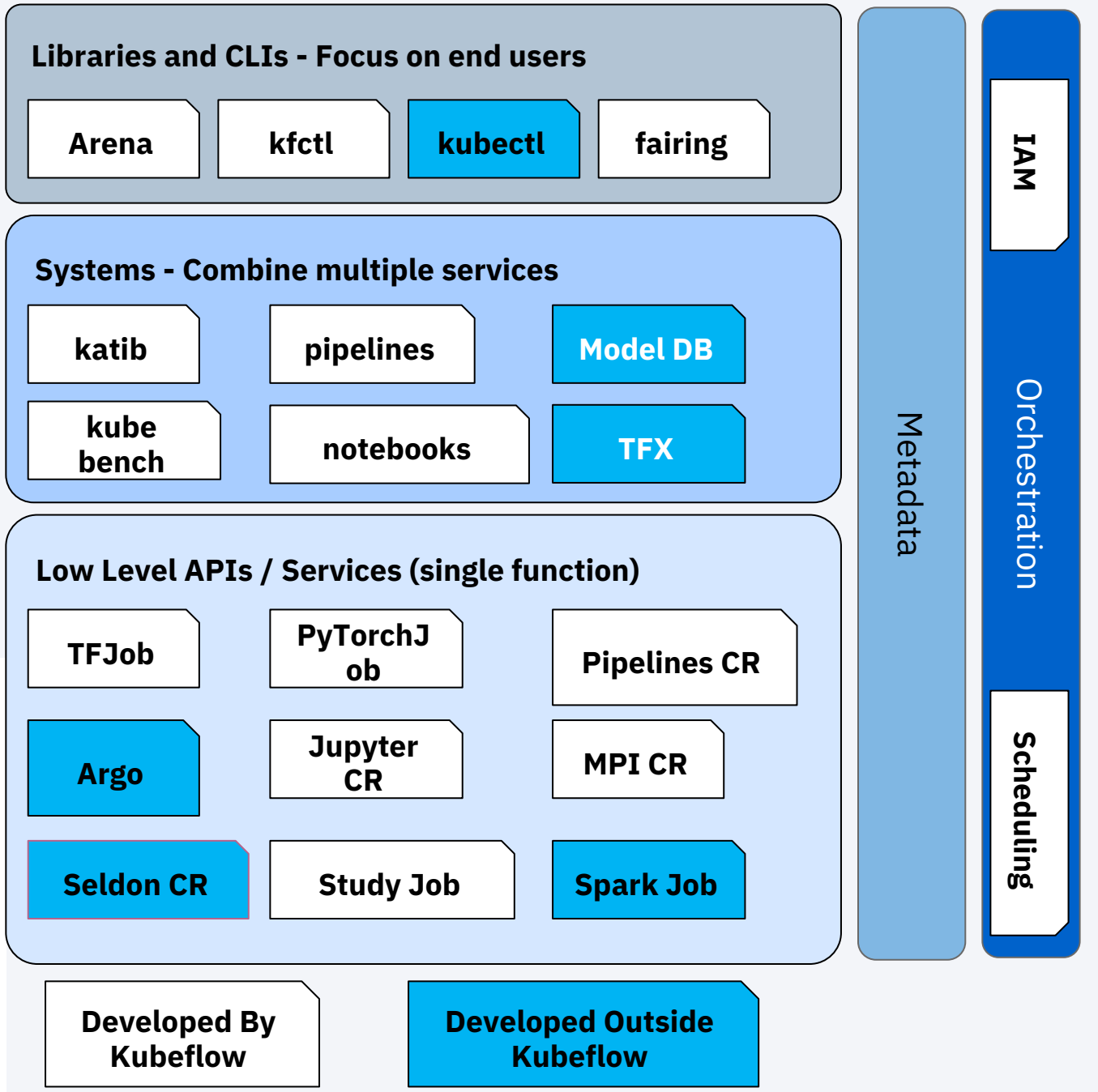
Commits by Company

Show 10 entries Search

#	Company	Commits
	*independent	3669
1	Google	904
2	IBM	234
3	Caicloud	205
4	Alibaba	110
5	Red Hat	71
6	Intel	65
7	Huawei	52
8	Cisco Systems	15
9	VA Linux	12

Showing 1 to 10 of 29 entries Previous Next





* Not all components shown

https://ibm.biz/bootstrap

Distributed Model Training and HPO (Katib, TFJob, PyTorch Job…)

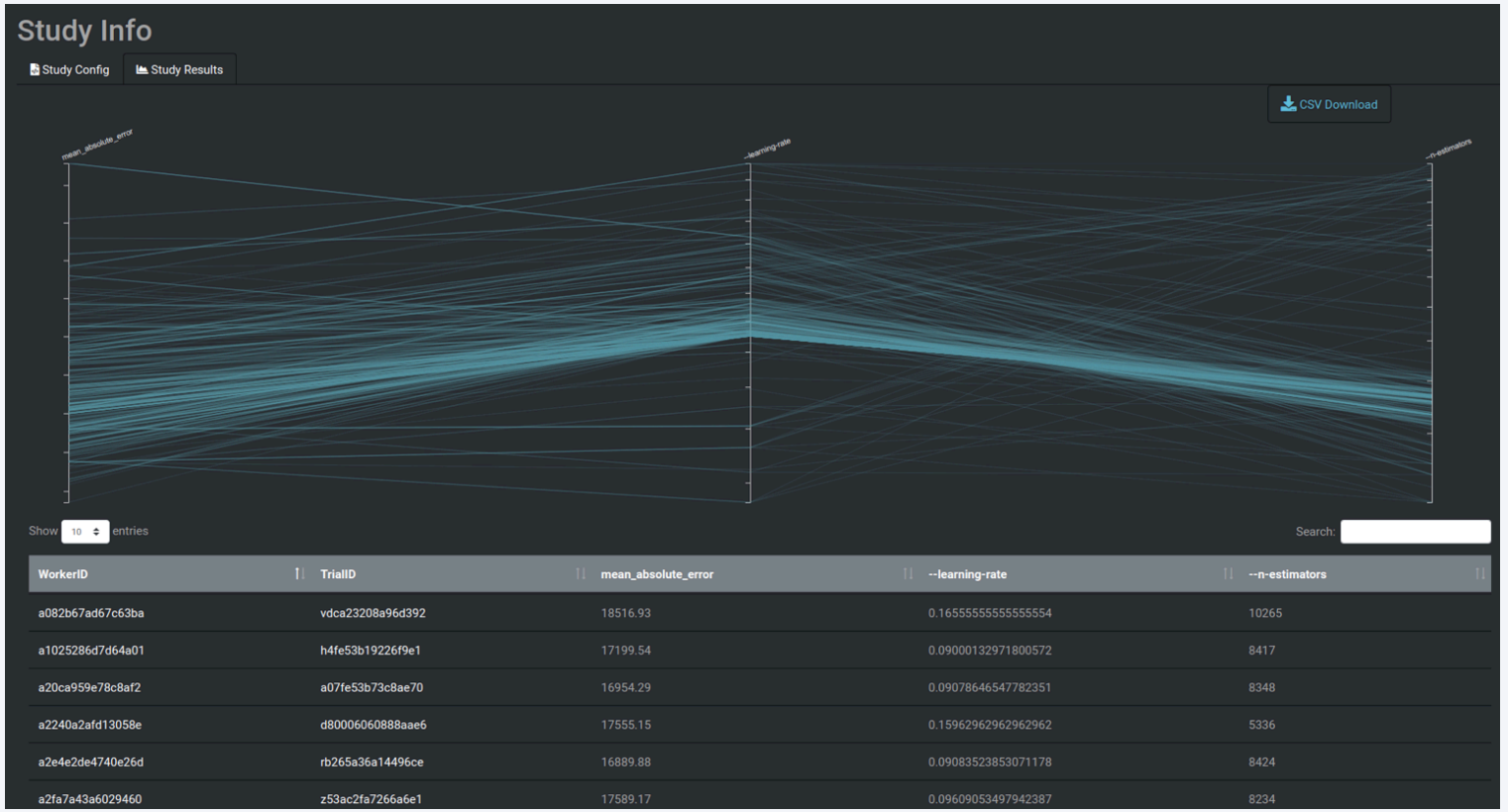
- Addresses One of the key goals for model builder persona:

  Distributed Model Training and Hyper parameter optimization for **Tensorflow, PyTorch etc.**

- Common problems in HP optimization
  - Overfitting
  - Wrong metrics
  - Too few hyperparameters

- Katib: a fully open source, Kubernetes-native hyperparameter tuning service
  - Inspired by Google Vizier
  - Framework agnostic
  - Extensible algorithms
  - Simple integration with other Kubeflow components

- Kubeflow also supports distributed MPI based training using Horovod

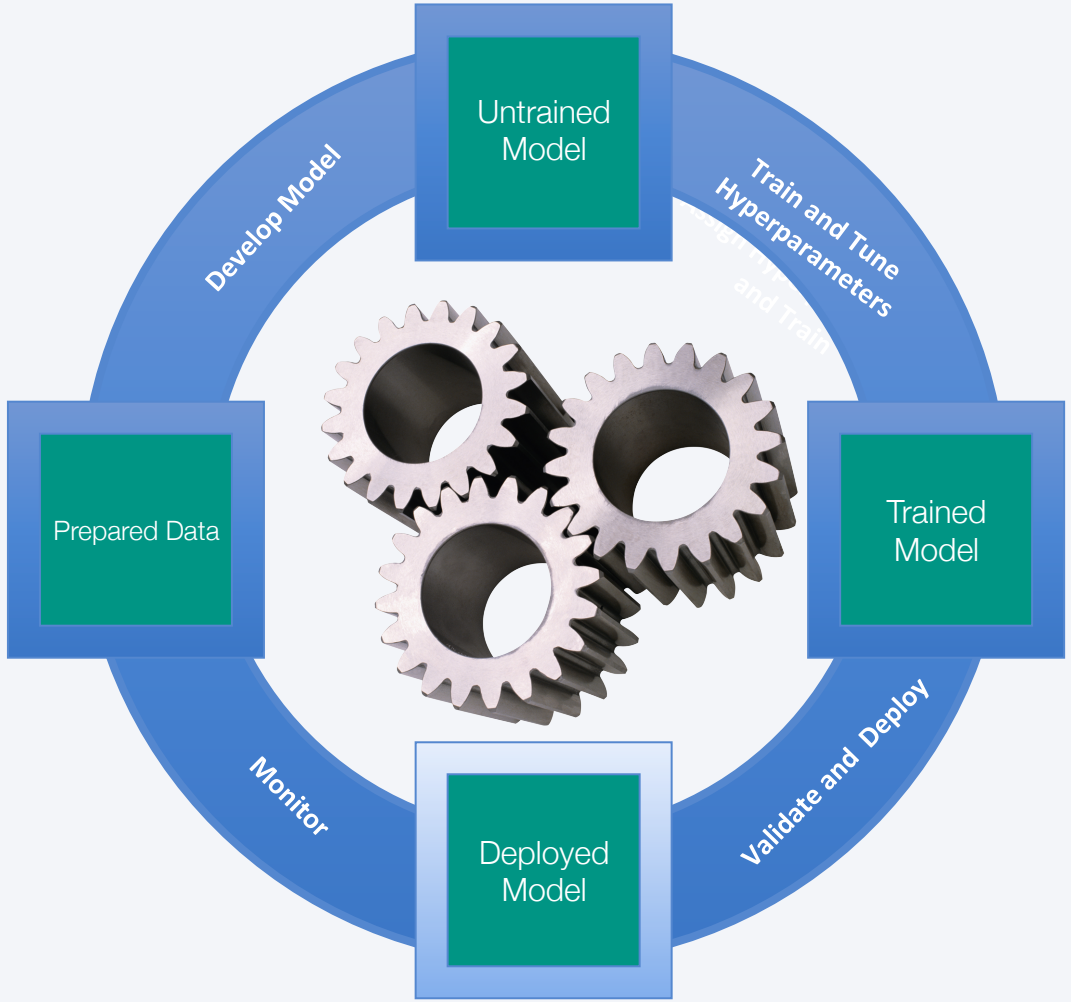


TensorFlow PYTORCH

HOROVOD

Model is ready for deployment: Just wrap in containers and push on Kube?

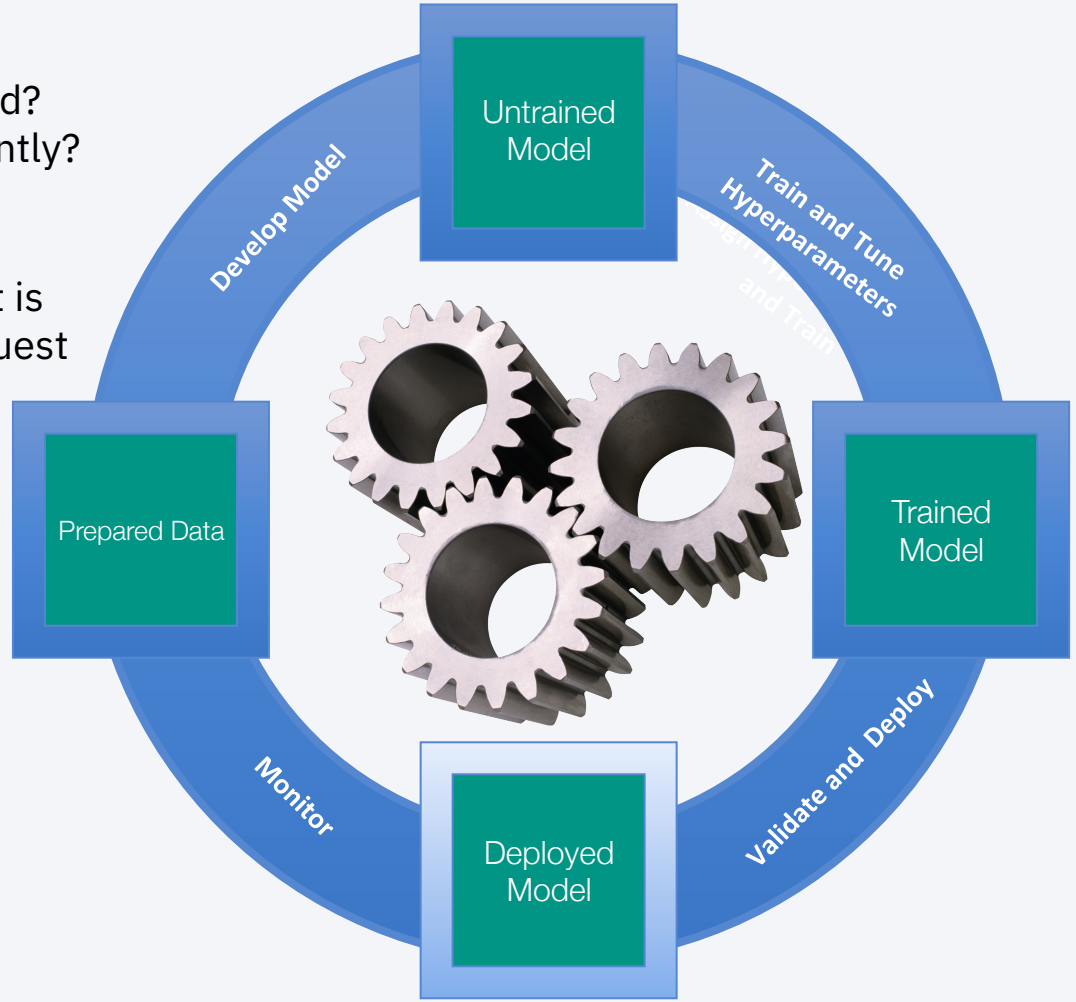
Anything else needed?



Production ML Model Serving? How hard could it be?

- **Cost**:
  Is the model over or under scaled?
  Are resources being used efficiently?

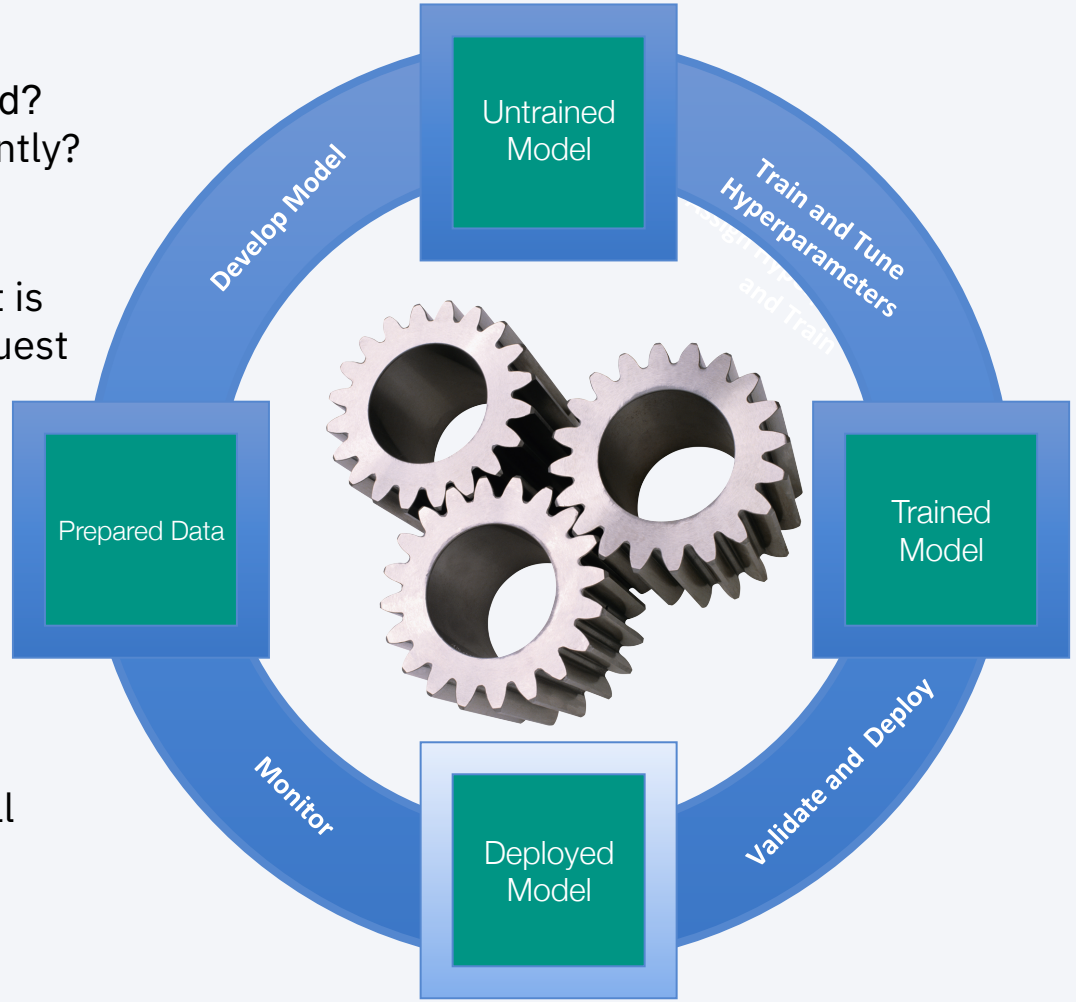
- **Monitoring**:
  Are the endpoints healthy? What is the performance profile and request trace?



Production ML Model Serving? How hard could it be?

- ❑ Cost:
  Is the model over or under scaled?
  Are resources being used efficiently?

- ❑ Monitoring:
  Are the endpoints healthy? What is the performance profile and request trace?



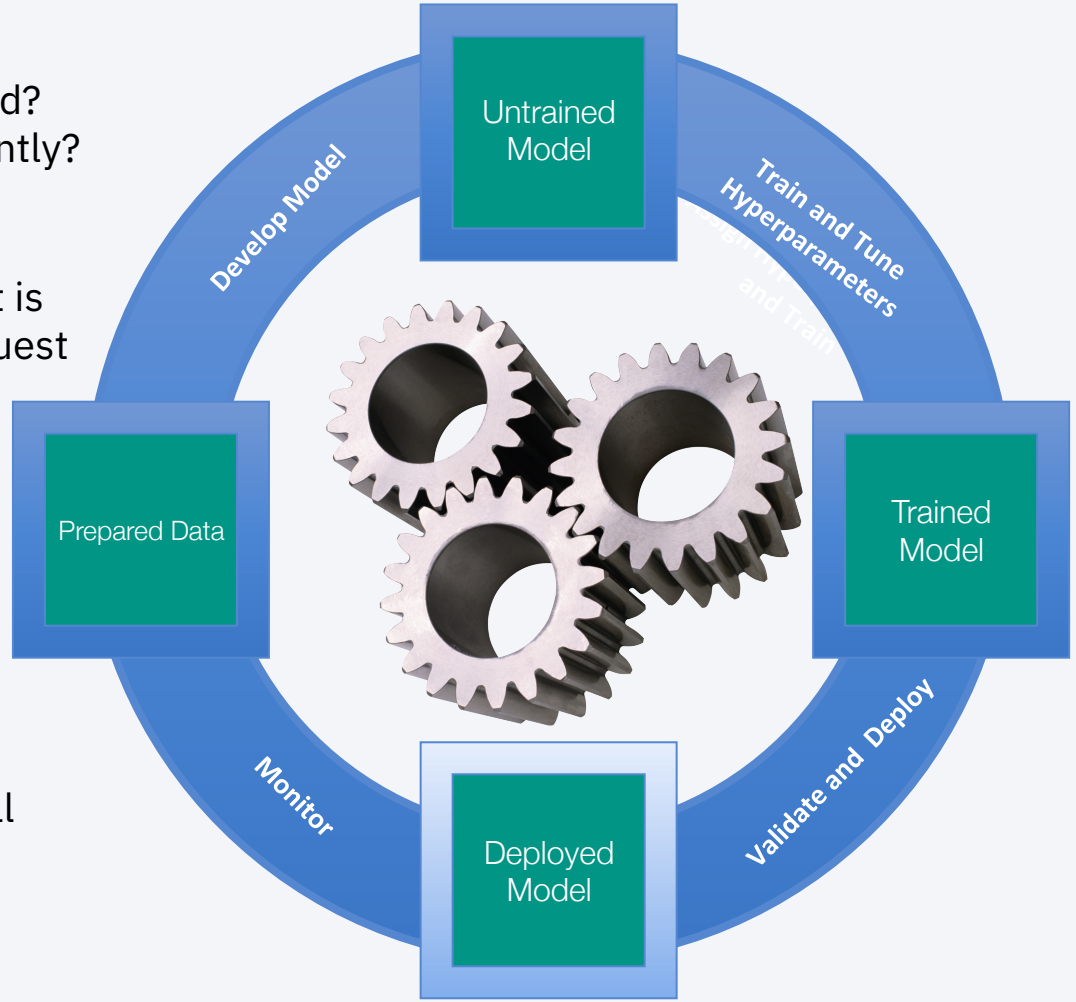
- ❑ Rollouts:
  Is this rollout safe? How do I roll back? Can I test a change without swapping traffic?

- ❑ Protocol Standards:
  How do I make a prediction? GRPC? HTTP? Kafka?

Production ML Model Serving? How hard could it be?

- **Cost:**
  Is the model over or under scaled?
  Are resources being used efficiently?

- **Monitoring:**
  Are the endpoints healthy? What is the performance profile and request trace?



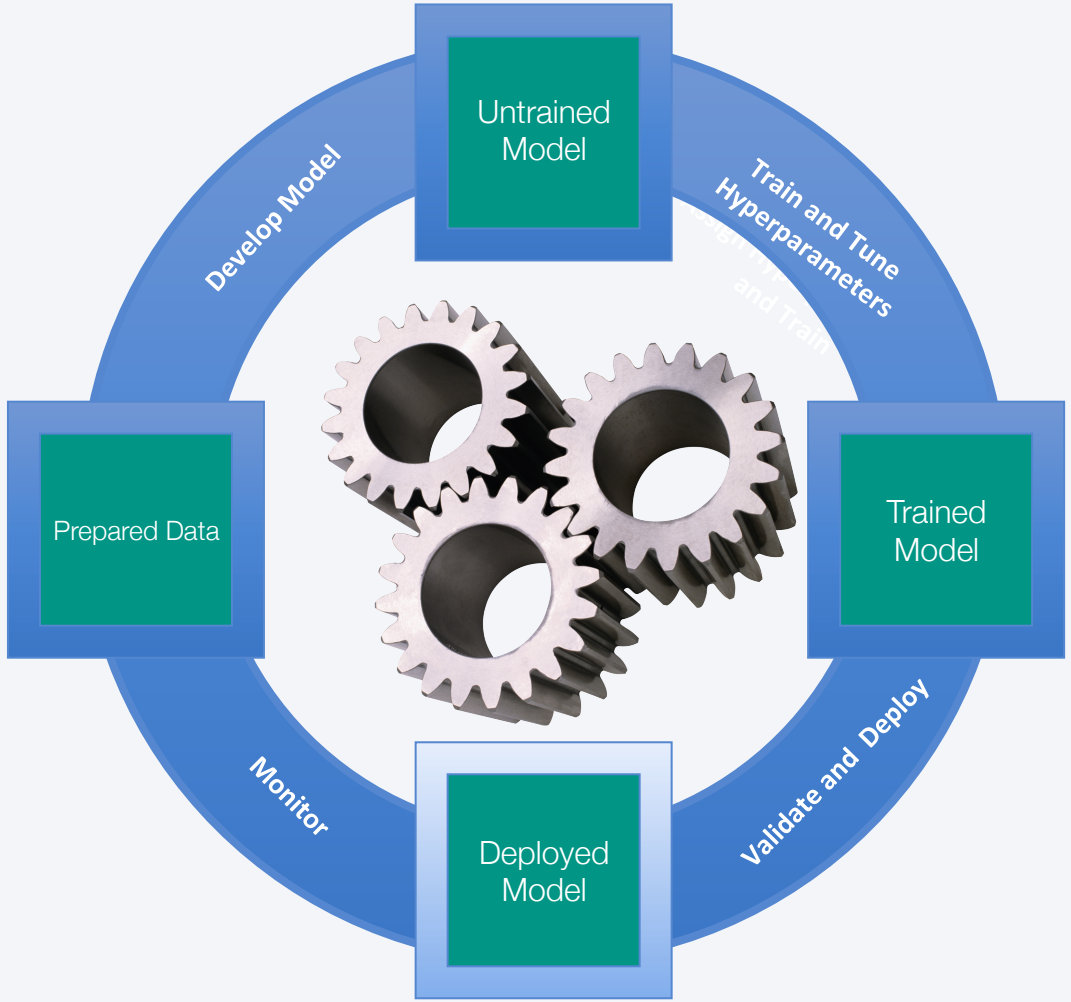
- **Rollouts:**
  Is this rollout safe? How do I roll back? Can I test a change without swapping traffic?

- **Protocol Standards:**
  How do I make a prediction? GRPC? HTTP? Kafka?

- **Frameworks:**
  How do I serve on Tensorflow? XGBoost? Scikit Learn? Pytorch? Custom Code?

- **Features:**
  How do I explain the predictions? What about detecting outliers and skew? How do I wire up custom pre and post processing?

Enter: KFServing (Kubeflow Serving)



KFServing

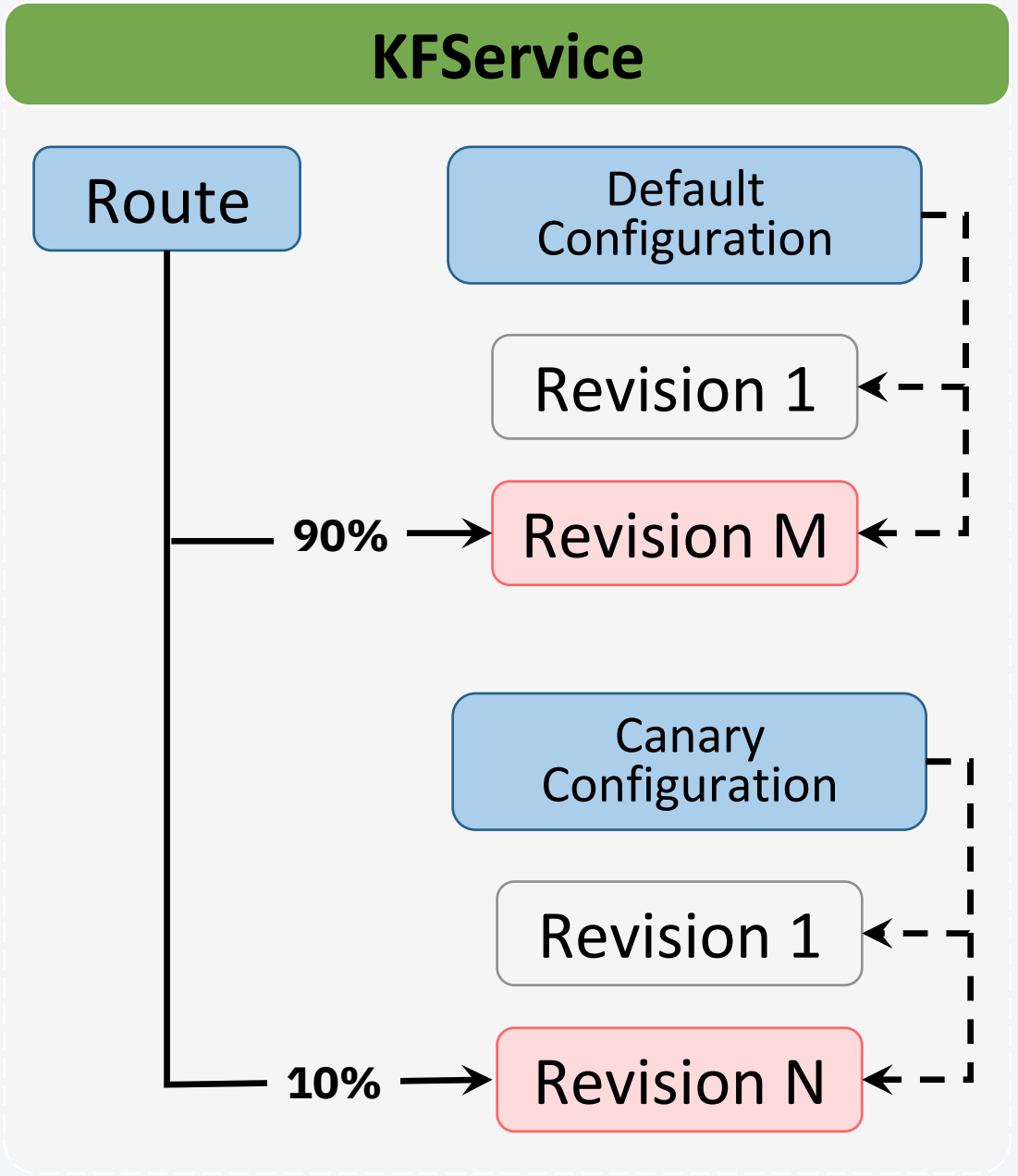
KFServing: Model Serving and Management

Bringing the power of Knative and Istio for serverless Model deployments

KFServing: Default and Canary Configurations

Manages the hosting aspects of your models

- **KFService** - manages the lifecycle of models
- **Configuration** - manages history of model deployments. Two configurations for default and canary.
- **Revision** - A snapshot of your model version
  - Config and image
- **Route** - Endpoint and network traffic management



KFServing

Now (0.1.0) Support

Model Source
- s3
- gs
- pvc
- file in image

Model Framework
- tensorflow
- tensorRT
- Pytorch
- Sklearn
- XGBoost
- custom

KFServing Roadmap

Multi-Model Serving.

- Multiple KFServices share resources in the backend.
- GPU Sharing.

Flexible Inference Graphs MLGraph CRD.

- Model Experimentation.
- Ensembling.
- Multi Arm Bandit.

Trusted Model Serving: Bias, Skew, and Outlier Detection.

- Online support in graph.
- Offline support with Payload Logging.

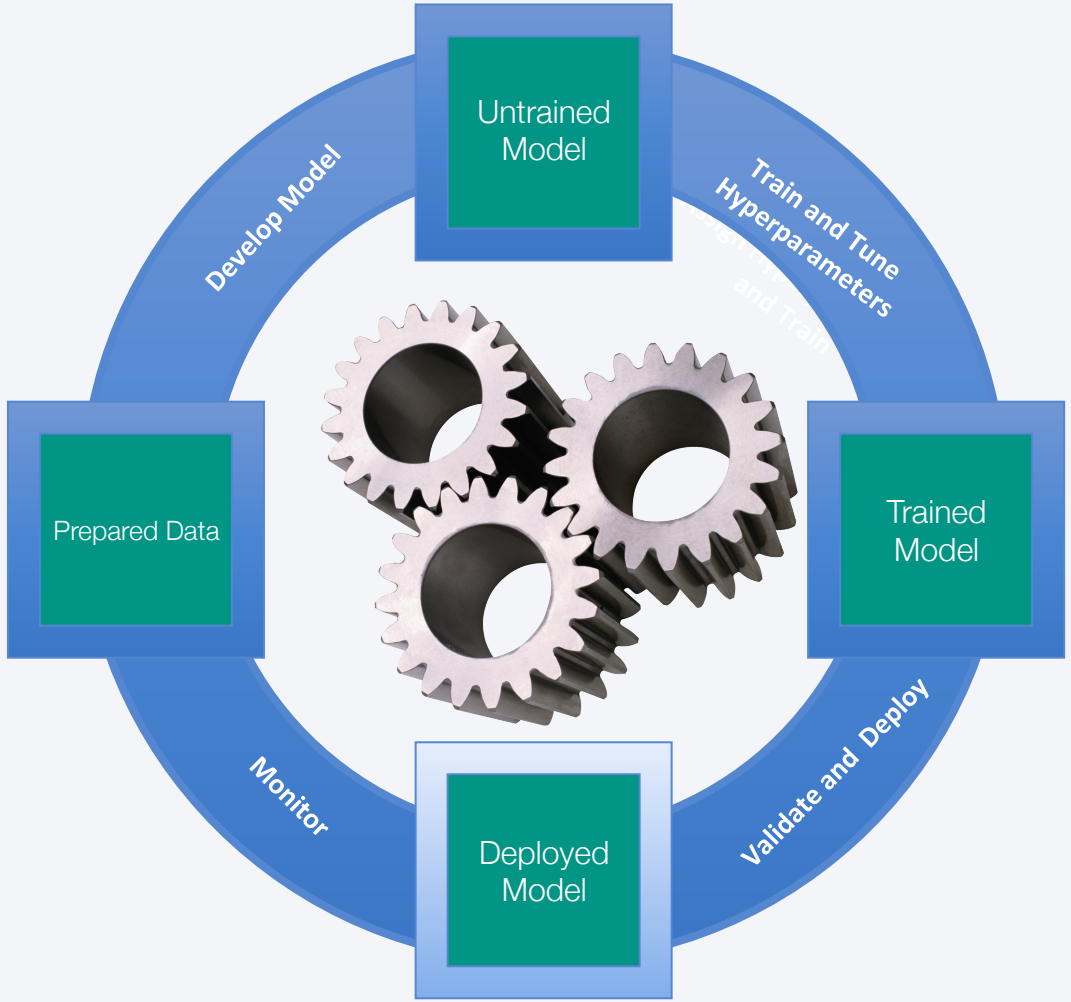
Meta-Protocol Definition.

- Unify disparate protocols across frameworks.

Adaptive batching support

- Queue and batch requests to increase throughput.

So Model is Trained, Tested and Deployed. How can we orchestrate his lifecycle together? We need AI Pipelines

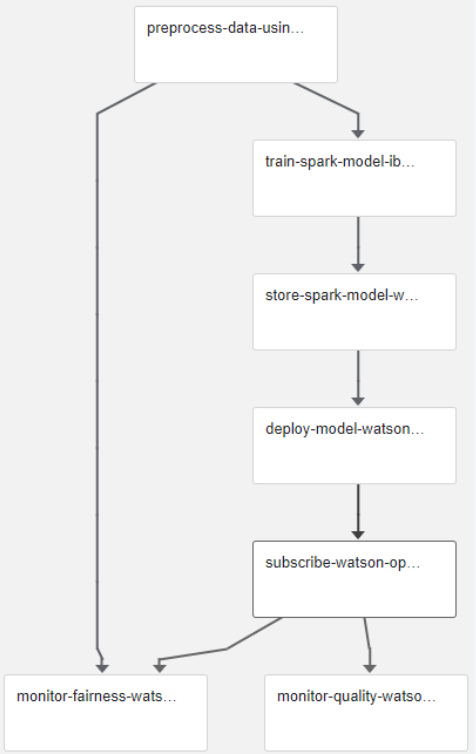


AI Pipelines

Pipelines

- Covers the whole lifecycle from data and source code to the application of models
  - ➢ end-to-end experiments – effective, efficient and reliable
  - ➢ lifecycle management – data engineering, train, validate, store, deploy, serve, inference, monitor, log, …
  - ➢ share and reuse
- Usually represented as DAGs (Directed Acyclic Graph)
- Blocks or steps have input and output dependencies
- Triggers and eventing
- Stores artifacts and metadata throughout pipeline

⇒ help solving “the last mile” problem



Infrastructure to support AI Pipelines

Functions

- compose, create, orchestrate, schedule, eventing, …
  - API and SDK
- deploy, manage, …
  - Kubernetes or other base infrastructure
  - portable, scalable, hybrid
- monitor, logging, visualization, …
  - UI, experiment tracking, metrics and other artifacts logging

Infrastructure to support AI Pipelines

Desired

- Easy deployment
  - One click deploy
  - One click to install components / operators (for example, on OpenShift)
  - Cloud agnostic (any Kubernetes clusters)
- Easy maintenance

  - Not to add the complexity of workflow
  - CI/CD pipelines
- Data and model versioning (b/c consistent reproducibility)
  - metadata, storage (S3, github repo, ...)
  - portable model packaging for easy serving

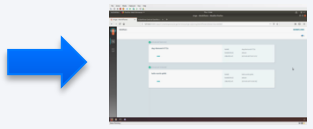
Infrastructure to support AI Pipelines

- More important to run on container platform
  - Container – packaging the execution environment (libraries, binaries, configure) with the application abstracted in a pre-built image (eg. Docker)
    - ✓ hybrid target environments: personal laptop, private or public cloud
    - ✓ run on hosts, VMs, Kubernetes alike clusters
    - ✓ self-contained/isolation
    - ✓ lightweight than VM
  - With Kubernetes
    - ✓ container orchestration
    - ✓ scalable
    - ✓ portable
    - ✓ service health monitoring

Open source projects for Pipelines

Argo Workflows

- https://argoproj.github.io/argo, https://github.com/argoproj/argo
- container-native workflow engine for Kubernetes, each step is a container
- workflow as CRD (Custom Resource Definition, .yaml)
- can run on any Kubernetes clusters
- orchestrate highly parallel tasks
- pipelines represented as DAGs
  - compose a workflow spec in yaml, run with Argo CLI or kubectl
  - input/output (artifacts/parameters), loops, conditionals, exit handler, daemon/sidecar container, and more

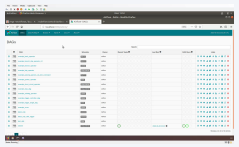


```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: dag-diamond-
spec:
  entrypoint: diamond
  templates:
  - name: diamond
    dag:
      tasks:
      - name: A
        template: echo
        arguments:
          parameters: [{name: message, value: A}]
      - name: B
        dependencies: [A]
        template: echo
        arguments:
          parameters: [{name: message, value: B}]
      - name: C
        dependencies: [A]
        template: echo
        arguments:
          parameters: [{name: message, value: C}]
      - name: D
        dependencies: [B, C]
        template: echo
        arguments:
          parameters: [{name: message, value: D}]

  - name: echo
    inputs:
      parameters:
      - name: message
    container:
      image: alpine:3.7
```

Open source projects for Pipelines

Airflow

- https://airflow.apache.org/, https://github.com/apache/airflow
- A platform to programmatically author, schedule and monitor workflows
  - DAGs, Operators, Tasks, TaskInstances, …
- Rich command line utilities and user interface
- Pipeline definitions written in Python
- KubernetesOperator, KubernetesPodOperator – each TaskInstance runs in a new pod
- (possibly used by mlflow)

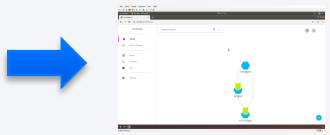


```python
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta


default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2015, 6, 1),
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    # 'queue': 'bash_queue',
    # 'pool': 'backfill',
    # 'priority_weight': 10,
    # 'end_date': datetime(2016, 1, 1),
}

dag = DAG('tutorial', default_args=default_args, schedule_interval=timedelta(days=1))

# t1, t2 and t3 are examples of tasks created by instantiating operators
t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag)

t2 = BashOperator(
    task_id='sleep',
    bash_command='sleep 5',
    retries=3,
    dag=dag)

templated_command = """
    {% for i in range(5) %}
        echo "{{ ds }}"
        echo "{{ macros.ds_add(ds, 7)}}"
        echo "{{ params.my_param }}"
    {% endfor %}
"""

t3 = BashOperator(
    task_id='templated',
    bash_command=templated_command,
    params={'my_param': 'Parameter I passed in'},
    dag=dag)

t2.set_upstream(t1)
t3.set_upstream(t1)
```

Open source projects for Pipelines

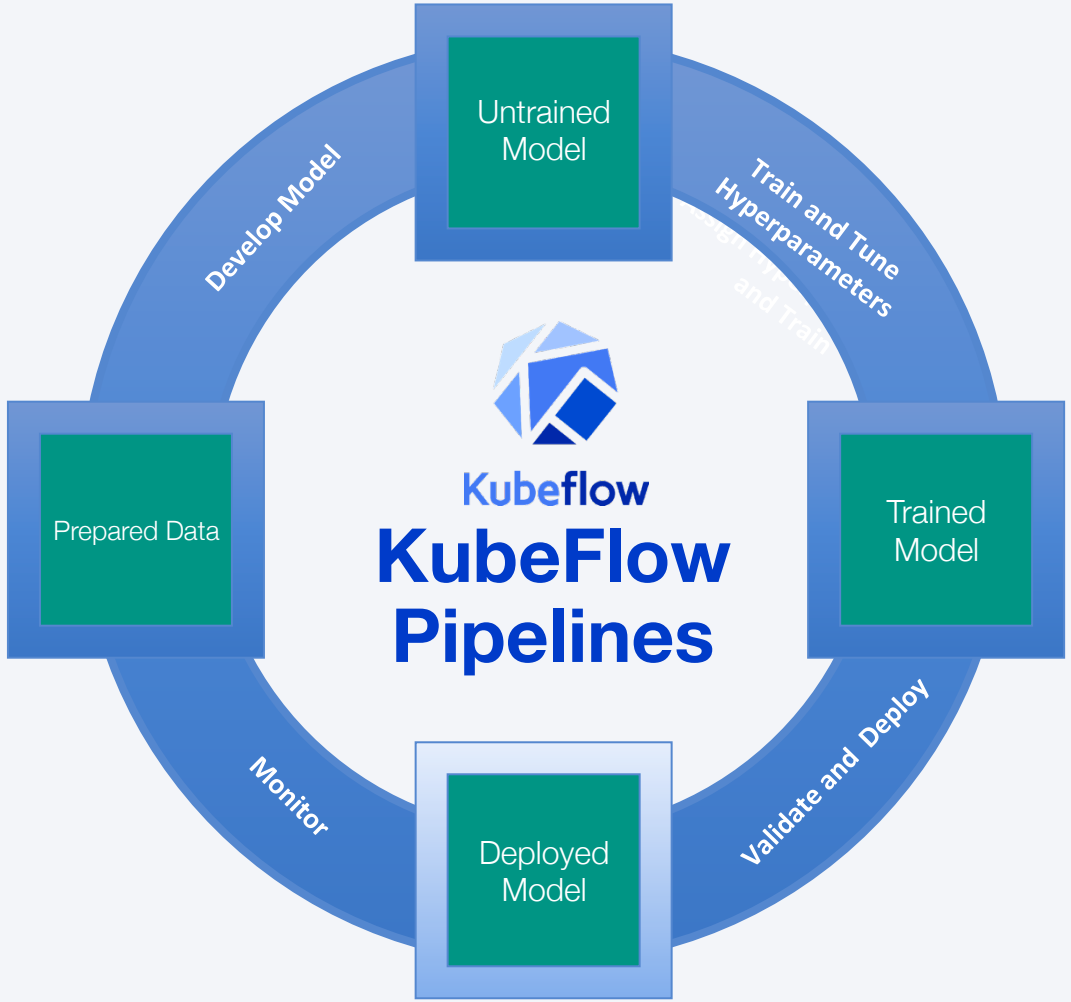
Data Pipelines – Pachyderm

- version control for data and other artifacts
- run on container platform
- data provenance
  - track any result all the way back to its raw input data, including all analysis, code, and intermediate results
- parallelism (datum ~ worker)
- create-repo, put-file, create-pipeline
- add more data → trigger new pipelines job
- CLI, python-pachyderm client, …
- pachyderm + kubeflow pipelines ?



```
# edges.json
{
  "pipeline": {
    "name": "edges"
  },
  "transform": {
    "cmd": [ "python3", "/edges.py" ],
    "image": "pachyderm/opencv"
  },
  "input": {
    "pfs": {
      "repo": "images",
      "glob": "/*"
    }
  }
}

# montage.json
{
  "pipeline": {
    "name": "montage"
  },
  "input": {
    "cross": [ {
      "pfs": {
        "glob": "/",
        "repo": "images"
      }
    },
    {
      "pfs": {
        "glob": "/",
        "repo": "edges"
      }
    } ]
  },
  "transform": {
    "cmd": [ "sh" ],
    "image": "v4tech/imagemagick",
    "stdin": [
      "montage -shadow -background SkyBlue -geometry 300x300+2+2 $(find /pfs -type f | sort) /pfs/out/montage.png"
    ]
  }
}
```

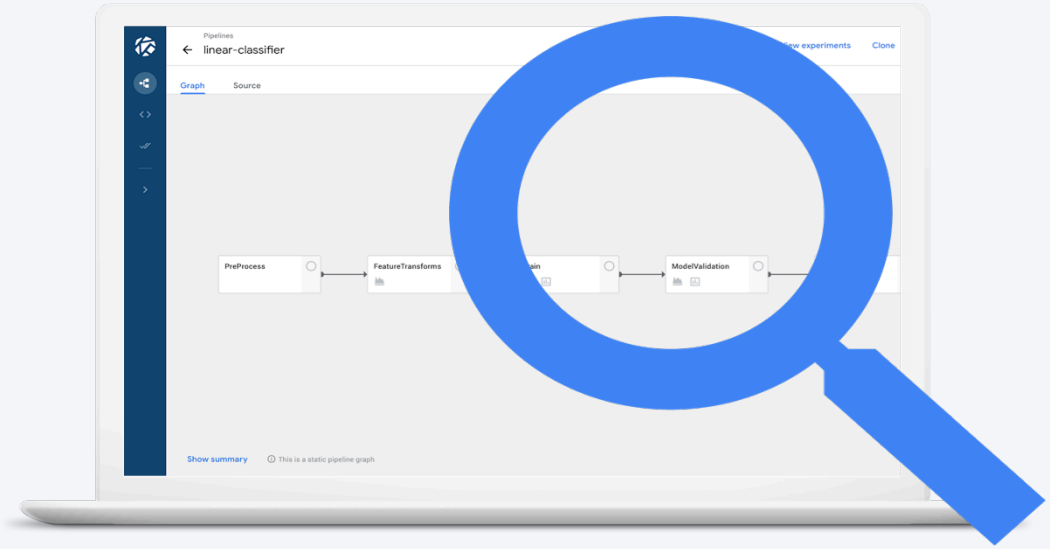
Last but not the least: Kubeflow Pipelines



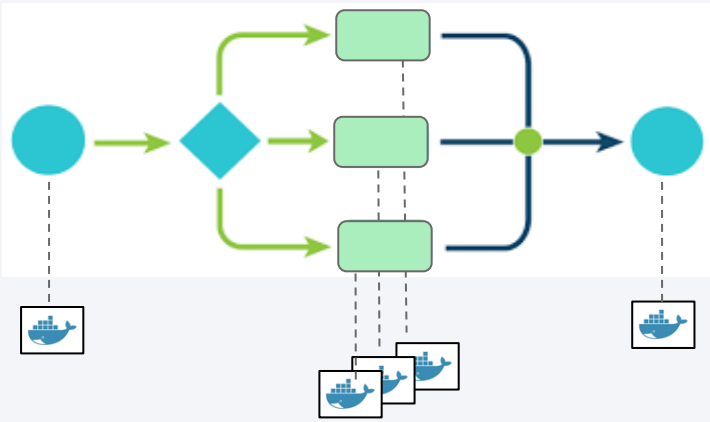
Kubeflow Pipelines

- Released to Kubeflow in Nov 2018, integrated into KF deployment CLI and 1-click-deploy-app
- Aimed to bring
  - Orchestration for complex ML workflows
  - Reproducible and reliable experimentation
  - Bridging experimentation and operationalization
  - Composition and reusable ML components and pipelines

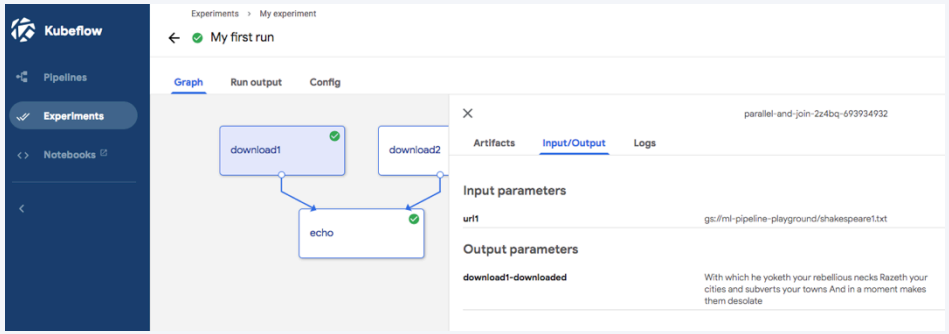
Kubeflow Pipelines

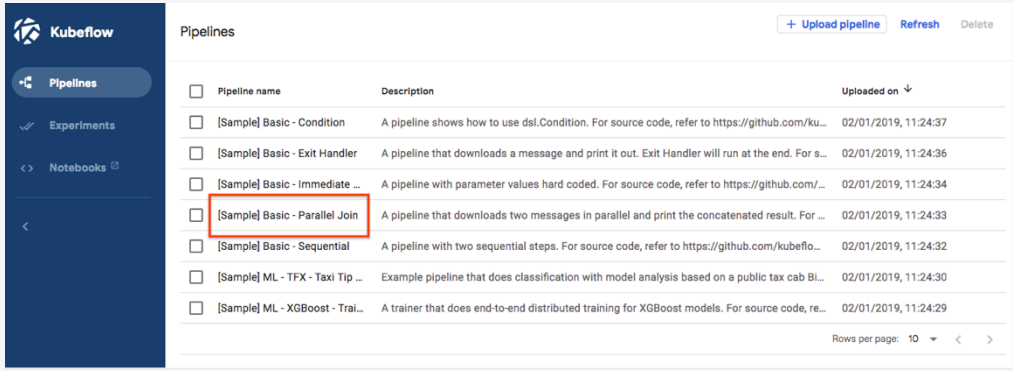


What constitutes a Kubeflow ML Pipeline

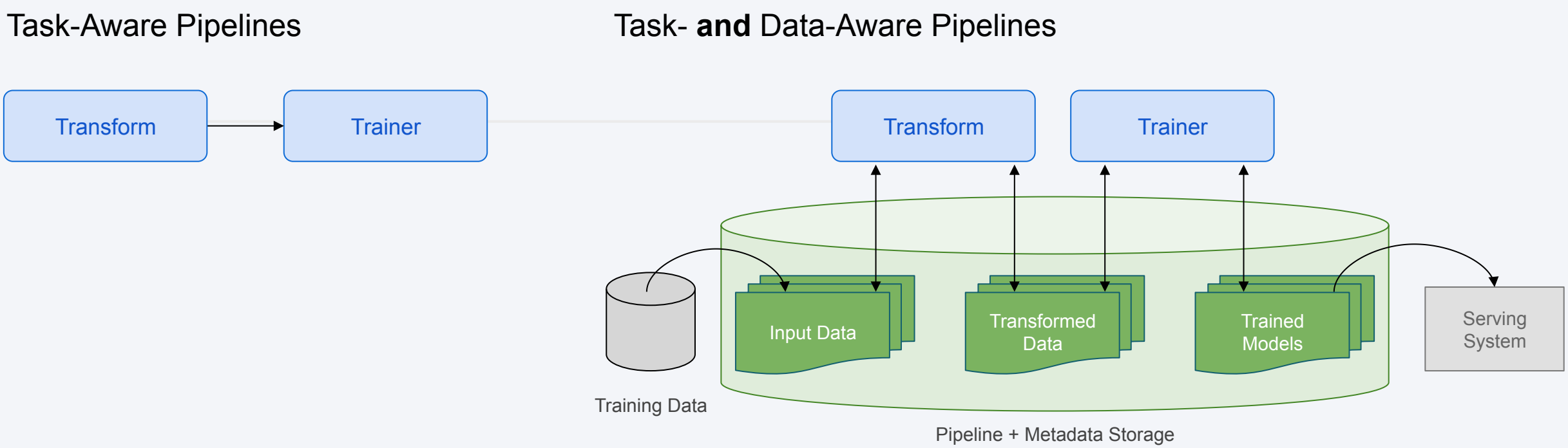


- Containerized implementations of ML Tasks
  - Pre-built components: Just provide params or code snippets (e.g. training code)
  - Create your own components from code or libraries
  - Use any runtime, framework, data types
  - Attach k8s objects - volumes, secrets
- Specification of the sequence of steps
  - Specified via Python DSL
  - Inferred from data dependencies on input/output
- Input Parameters
  - A “Run” = Pipeline invoked w/ specific parameters
  - Can be cloned with different parameters
- Schedules
  - Invoke a single run or create a recurring scheduled pipeline

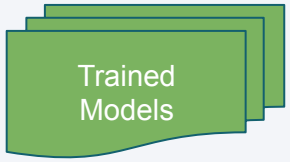




Metadata and Pipeline Execution



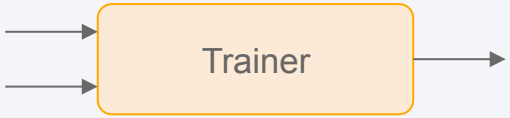
What's in the Metadata Store?



Type definitions of Artifacts and their Properties

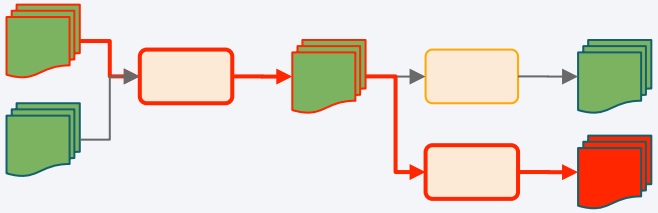
Models, Data, Evaluation Metrics, Visualizations

(core types come out of the box, custom types can be defined)



Execution Records (Runs)

Runtime configuration params, Inputs + Outputs references, versioning of binaries/images

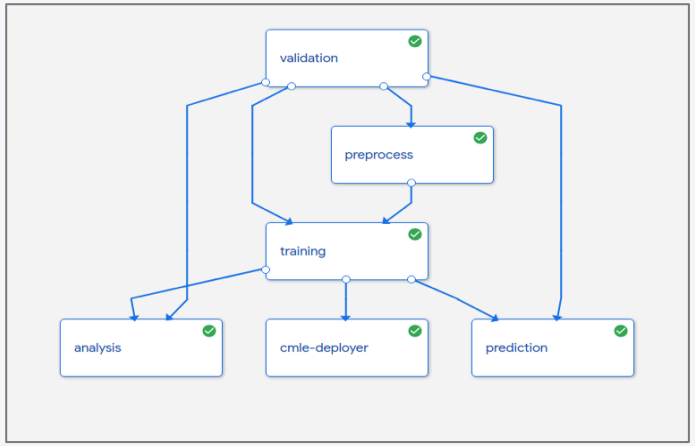


Lineage Tracking Across All Executions

To recurse back to all inputs of a specific artifact.

Find out how a given dataset was used? What was used to create a given model?

Define Pipeline with Python SDK



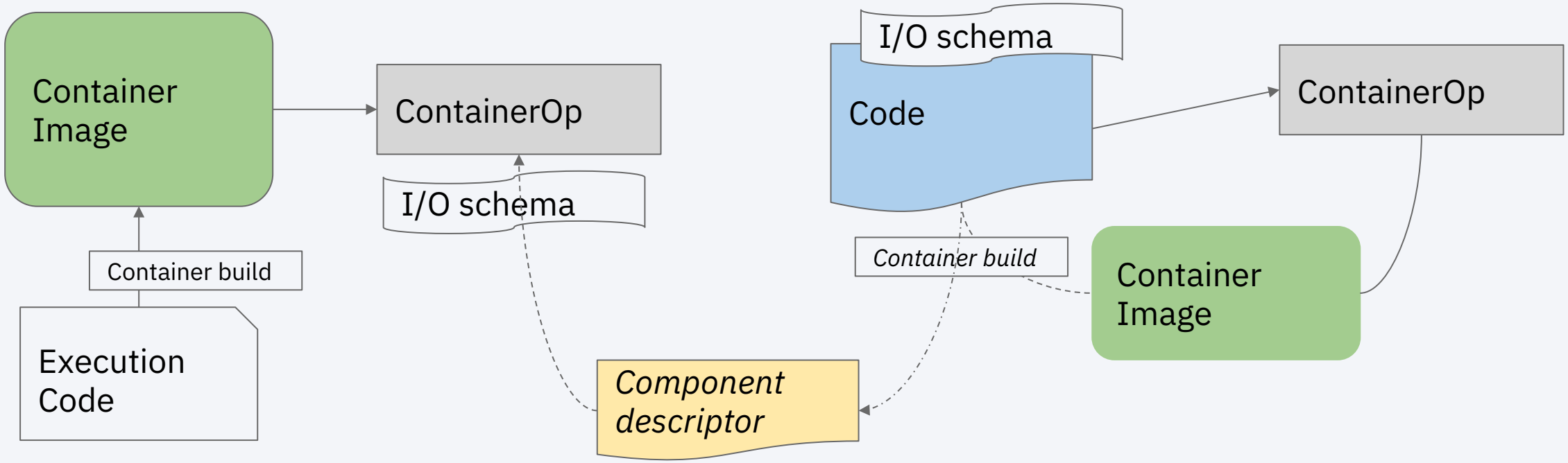

```
@dsl.pipeline(name='Taxi Cab Classification Pipeline Example’)
def taxi_cab_classification(
    output_dir,
    project,
    Train_data      = 'gs://bucket/train.csv',
    Evaluation_data = 'gs://bucket/eval.csv',
    Target          = 'tips',
    Learning_rate   = 0.1, hidden_layer_size = '100,50’, steps=3000):

        tfdv                = TfdvOp(train_data, evaluation_data, project, output_dir)
        preprocess          = PreprocessOp(train_data, evaluation_data, tfdv.output[“schema”], project, output_dir)
        training  = DnnTrainerOp(preprocess.output, tfdv.schema, learning_rate, hidden_layer_size, steps,
                                 target, output_dir)
        tfma                = TfmaOp(training.output, evaluation_data, tfdv.schema, project, output_dir)
        deploy    = TfServingDeployerOp(training.output)
```

Compile and Submit Pipeline Run

```
dsl.compile(taxi_cab_classification,  'tfx.tar.gz')
run = client.run_pipeline(
      'tfx_run', 'tfx.tar.gz', params={'output': ‘gs://dpa22’, 'project': ‘my-project-33’})
```

Creating your own components

- Ways to build reusable components for pipelines
  - Create a container with your code and write either a ContainerOp() or shareable component descriptor
  - Turn your python code into a component directly in the notebook (with or without building a container)
    - These components can be exported into a shareable format



Simply Create Component from a Python function..

```
@dsl.python_component(
    name='cmle_deployer',
    description='deploys a model to GCP CMLE',
    base_image='python:3.5')
def deploy_model(model_dot_version: str, model_path: str, gcp_project: str):
  # code to deploy a model
  …
  …
```

Locally test it and convert to a component with a container image

```
MyDeployerOp = compiler.build_python_component(
      component_func=deploy_model,
      target_image=gcr.io/myproject33/mydeployer-image:1.0')
```

Use it in a Pipeline

```
@dsl.pipeline(
  name='Taxi Cab Classification Pipeline Example',
)
def taxi_cab_classification():

    ...
    deploy = TfServingDeployerOp(training.output)  ⇒ deploy = MyDeployerOp(name, training.output, project)
```

Create Component from a Python function without building container

In [ ]:
```
import kfp.components as comp
```

Simple function that just add two numbers:

In [ ]:
```
#Define a Python function
def add(a: float, b: float) -> float:
   '''Calculates sum of two arguments'''
   return a + b
```

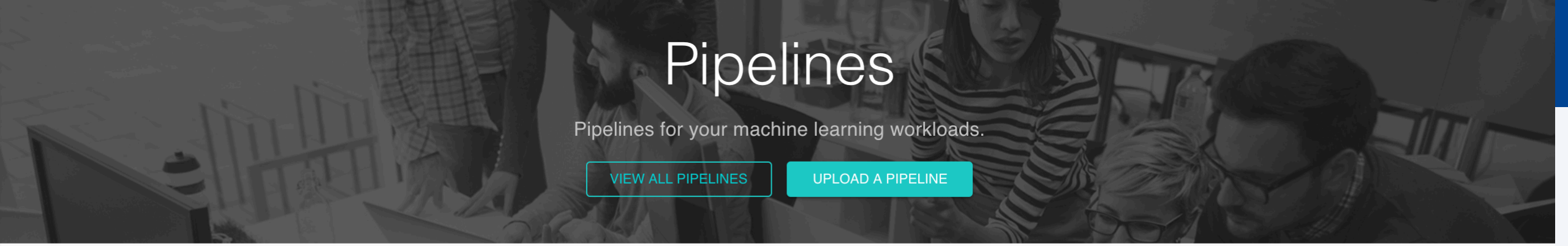
Convert the function to a pipeline operation

In [ ]:
```
add_op = comp.func_to_container_op(add)
```

Can export it to a file with component definition from here

Use it in a Pipeline

In [ ]:
```
import kfp.dsl as dsl
@dsl.pipeline(
   name='Calculation pipeline',
   description='A toy pipeline that performs arithmetic calculations.'
)
def calc_pipeline(
   a='a',
   b='7',
   c='17',
):
    #Passing pipeline parameter and a constant value as operation arguments
    add_task = add_op(a, 4) #Returns a dsl.ContainerOp class instance.

    #Passing a task output reference as operation arguments
    #For an operation with a single return value, the output reference can be accessed using `tas
k.output` or `task.outputs['output_name']` syntax
    divmod_task = divmod_op(add_task.output, b, '/')

    #For an operation with a multiple return values, the output references can be accessed using `
task.outputs['output_name']` syntax
    result_task = add_op(divmod_task.outputs['quotient'], c)
```



Pipelines

Pipelines for your machine learning workloads.

VIEW ALL PIPELINES | UPLOAD A PIPELINE

Train with FfDL and canary testing with Knative

This pipeline trains 2 models, does fairness and robustness checking, then does canary test with Knative for deployment.

Open Source

Train and deploy with FfDL and Seldon (OSS)

A simple IBM OSS pipeline demonstrates how to train a model using Fabric for Deep Learning and then deploy it with Seldon.

Open Source

Train and deploy with Watson Machine Learning

A sample pipeline for training, storing and deploying a Tensorflow model with MNIST handwriting recognition on IBM Watson Machine Learning service.

IBM Watson

Model analysis and operations using AI OpenScale

IBM AI OpenScale to monitor and analyze ML models.

Open Source

End-to-end pipeline with fairness and robustness test.

This AI pipeline uses FfDLto train a model, does fairness checking and robustness checking. For successful checking results, it does the model deployment with A/B testing.

IBM Watson

Kubernetes model deployment

Deploy a container to a Kuberentes platform with its own service.

IBM Watson

Knative model deployment

Deploy a container to a Knative platform with traffic splitting.

Open Source

[Sample] Basic - Condition

A pipeline shows how to use dsl.Condition. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/basic/condition.py.

IBM Watson

[Sample] Basic - Parallel Join

A pipeline that downloads two messages in parallel and print the concatenated result. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/basic/parallel_join.py.

IBM Watson

[Sample] ML - XGBoost - Training with Confusion Matrix

A trainer that does end-to-end distributed training for XGBoost models. For source code, refer to https://github.com/kubeflow/pipelines/tree/master/samples/xgboost-spark.

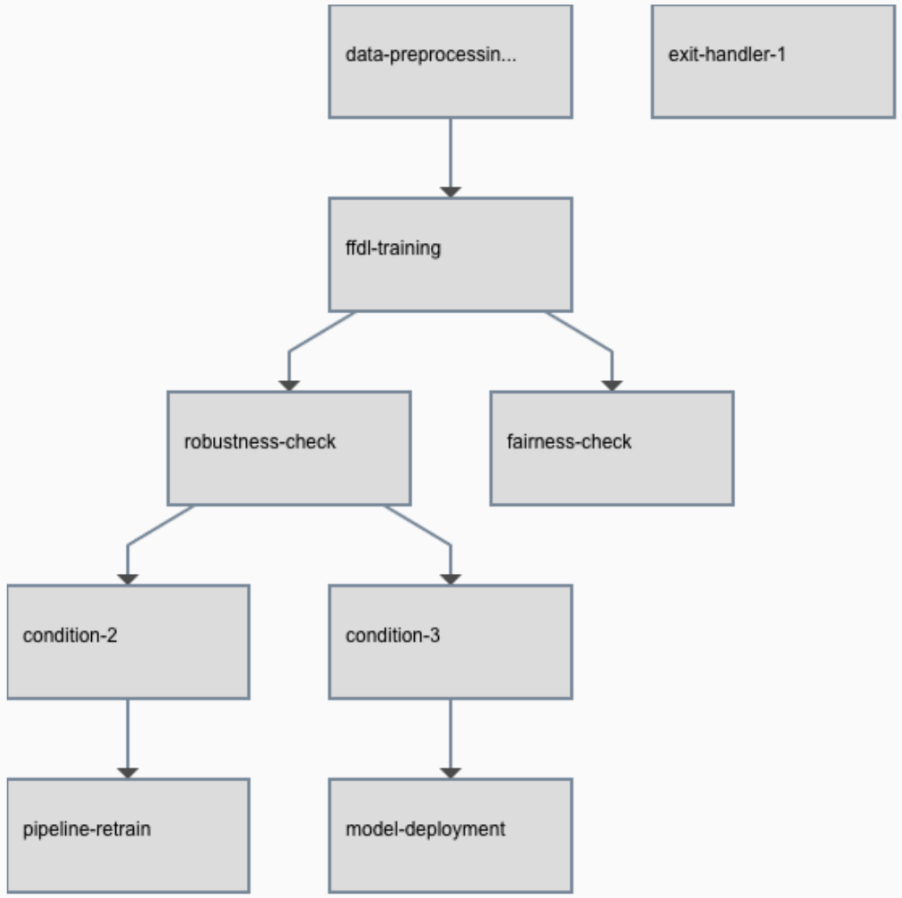
Open Source

Pipelines

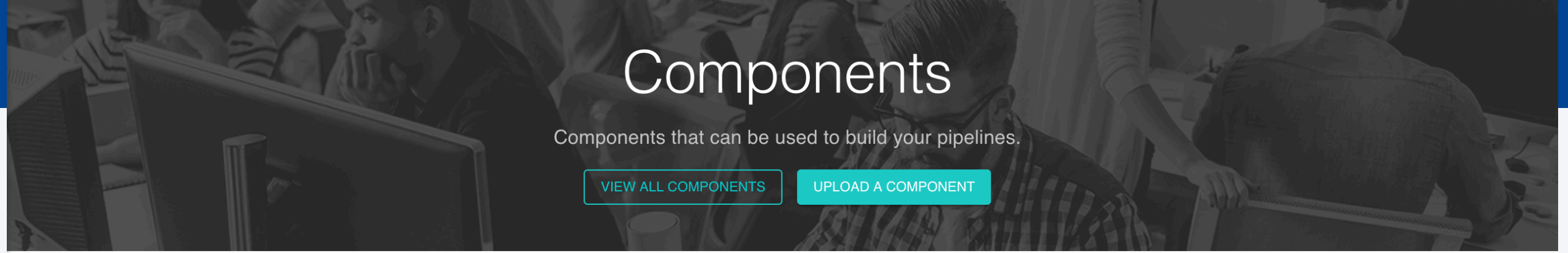
Train a model, do fairness and robustness checks, and deploy it with A/B testing.

PIPELINES | GET CODE

GRAPH | DETAILS | PIPELINE YAML




```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: ffdl-pipeline-
spec:
  arguments:
    parameters:
    - name: model-def-file-path
      value: gender_classification_training.zip
    - name: manifest-file-path
      value: manifest_training.yml
    - name: preprocessing-def-file-path
      value: preprocessing.zip
    - name: preprocessing-manifest-file-path
      value: preprocessing_manifest.yml
    - name: fgsm-attack-epsilon
      value: '0.2'
    - name: retrain-pipe-exp-id
      value: 5adacf87-48aa-4c6e-9aad-9a9cf933b9a6
    - name: retrain-pipeline-id
      value: f246bd8a-832e-4560-a8db-3cbee851c3fd
    - name: pipeline-url
      value: http://169.48.196.99/pipeline
    - name: primary-model-revision
      value: model-serving-00001
    - name: new-model-traffic-percentage
      value: '5'
    - name: model-class-file
      value: gender_classification_training.py
    - name: model-class-name
      value: ThreeLayerCNN
```



Components

Components that can be used to build your pipelines.

VIEW ALL COMPONENTS | UPLOAD A COMPONENT

Deploy Model - Watson Machine Learning

Deploy stored model on Watson Machine Learning as a web service.

IBM Watson Machine Learning

Subscribe - Watson OpenScale

Binding deployed models and subscribe them to Watson OpenScale service.

IBM Watson OpenScale

Store model - Watson Machine Learning

Store and persistent trained model on Watson Machine Learning.

IBM Watson Machine Learning

Knative model deploy

Deploy AI models using Knative serving.

OpenSource

Fabric for Deep Learning - Train Model

Train Machine Learning and Deep Learning Models remotely using Fabric for Deep Learning

OpenSource

Model Robustness Check

Perform a robustness check using fast gradient attack with ART to make sure the model is robust against simple gradient changes.

OpenSource

Kubernetes model deploy

Deploy AI models using Kubernetes deployment.

OpenSource

Train Model - Watson Machine Learning

Train Machine Learning and Deep Learning Models in the Cloud using Watson Machine Learning

IBM Watson Machine Learning

Seldon Core - Serve PyTorch Model

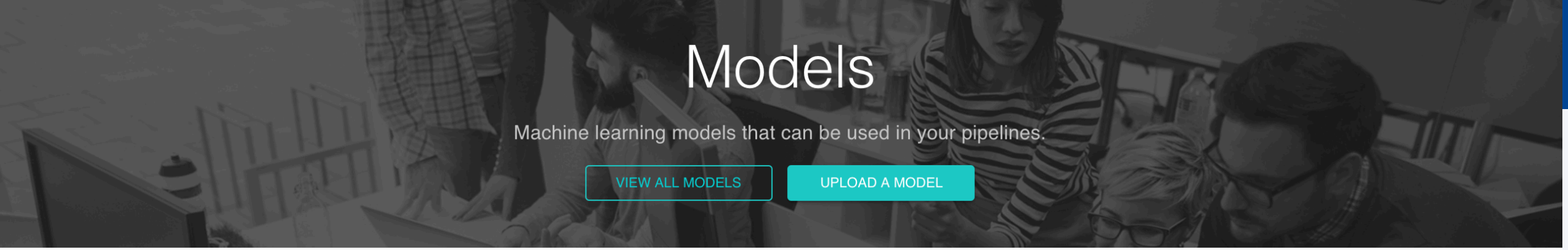
Serve PyTorch Models remotely as web service using Seldon Core

OpenSource

Model Fairness Check

Perform a fairness check on a certain attribute using AIF360 to make sure the model is fair and ethical

OpenSource



Models

Machine learning models that can be used in your pipelines.

VIEW ALL MODELS | UPLOAD A MODEL

MAX Audio Classifier

IBM Model Asset eXchange(MAX) model that dentifies sounds in short audio clips

Audio Classification

MAX Breast Cancer Mitosis Detector

IBM Model Asset eXchange(MAX) model that detects whether a mitosis exists in an image of breast cancer tumor cells

Image Classification

MAX Facial Age Estimator

IBM Model Asset eXchange(MAX) model that recognizes faces in an image and estimate the age of each face

Facial Recognition

MAX Image Caption Generator

IBM Model Asset eXchange(MAX) model that generates captions from a fixed vocabulary describing the contents of images in the COCO dataset

Image-To-Text Translation

MAX Image Segmenter

IBM Model Asset eXchange(MAX) model that identifies objects in an image, additionally assigning each pixel of the image to a particular object

Object Detection in image

MAX Named Entity Tagger

IBM Model Asset eXchange(MAX) model that locates and tags named entities in text

Natural Language Processing

MAX News Text Generator

IBM Model Asset eXchange(MAX) model that generates English-language text similar to the news articles in the One Billion Words data set

Language Modeling

MAX Object Detector

IBM Model Asset eXchange(MAX) model that localizes and identifies multiple objects in a single image

Object detection in images

MAX Sports Video Classifier

IBM Model Asset eXchange(MAX) model that classifies sporting activities in videos

Video Classification

MAX Weather Forecaster

IBM Model Asset eXchange(MAX) model that predicts hourly weather features given historical data for a specific location

Time Series Prediction

MAX Facial Age Estimator

IBM Model Asset eXchange(MAX) model that recognizes faces in an image and estimate the age of each face

MODELS | GET CODE

DETAILS | CREATE RUN | YAML DEFINITION | SAMPLE SERVING CODE

About:

model identifier	max-facial-age-estimator
domain	Facial Recognition
description	IBM Model Asset eXchange(MAX) model that recognizes faces in an image and estimate the age of each face

Usage Info:

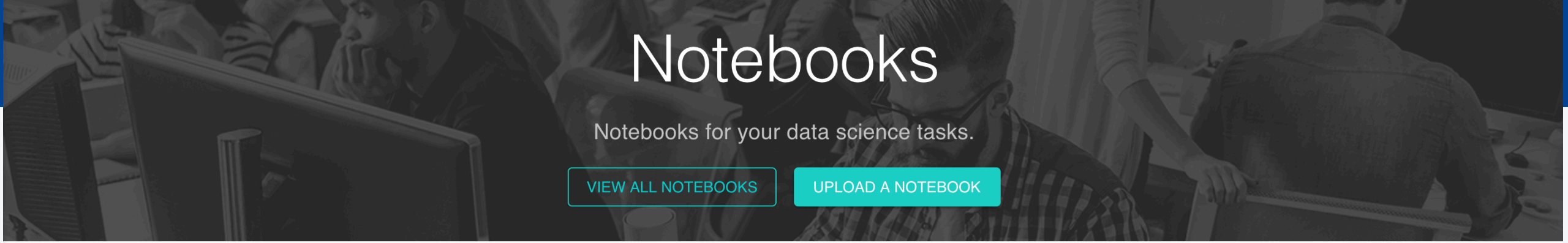
framework name	Keras	v2.2.2
license	Apache 2.0	
website	https://developer.ibm.com/exchanges/models/all/max-facial-age-estimator	

Serving Details:

tested platforms	kubernetes, knative
container image	codait/max-facial-age-estimator:latest

KUBERNETES | KNATIVE

```
@dsl.pipeline(
    name='Model Deployment pipeline',
    description='A pipeline for ML/DL model deployment on Kubernetes.'
)
def model_pipeline(model_id='max-facial-age-estimator'):
    """A pipeline for ML/DL model deployment."""

    from kfp import dsl
    from ai_pipeline_params import use_ai_pipeline_params

    secret_name = 'e2e-creds'

    model_config = dsl.ContainerOp(
        name='model_config',
        image='tomcli/model-config',
        command=['python'],
        arguments=[
            '-u', 'model-config.py',
            '--secret_name', secret_name,
            '--model_id', model_id
        ],
        file_outputs={
            'model_serving_image': '/tmp/model_serving_image',
            'deployment_name': '/tmp/deployment_name'
        }
    )

    model_deployment = dsl.ContainerOp(
```



Notebooks

Notebooks for your data science tasks.

VIEW ALL NOTEBOOKS | UPLOAD A NOTEBOOK

AIF360 Gender Classification

Notebook to train AIF360 Gender Classification with reweighing.

OpenSource

ART detector model

Notebook to train ART detector model to dector possible adversarial attack.

OpenSource

ART poisoning attack

Use Notebook to leverage ART for poisoning training data and learn how to defends it.

OpenSource

ART with FfDL

Notebook to do Adversarial training using Fabrics for Deep Learning.

OpenSource

Train with FfDL and canary testing with Knative.

This notebook trains a model, does fairness and robustness checking, then does canary test with Knative for deployment.

OpenSource

End-to-end with fairness and robustness test

This notebook uses FfDL to train a model, does fairness checking and robustness checking.

OpenSource

Watson OpenScale

A notebook showcases how to train custom Spark model on the Cloud and use IBM WatsonOpenScale to monitor and analyze ML model.

Watson OpenScale

Watson OpenScale walk through

End to End Watson OpenScale walk through.

Watson OpenScale

ART poisoning attack

Use Notebook to leverage ART for poisoning training data and learn how to defends it

NOTEBOOKS | GET CODE

DETAILS | CREATE RUN | YAML DEFINITION | NOTEBOOK CODE

About:

name	ART poisoning attack
description	Use Notebook to leverage ART for poisoning training data and learn how to defends it
platform	OpenSource

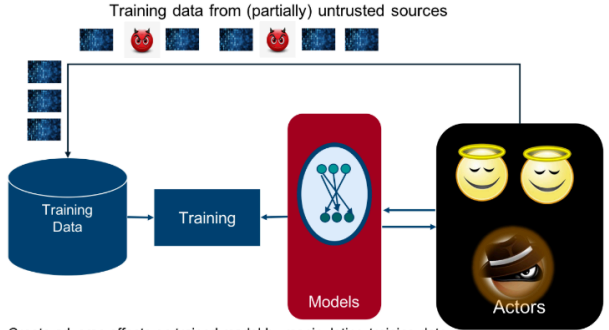
Implementation:

github	https://github.com/IBM/adversarial-robustness-toolbox/blob/master/notebooks/mnist_poisoning_demo.ipynb

jupyter nbviewer

adversarial-robustness-toolbox / notebooks

Poisoning Attacks



Backdoor attacks:

Recent work (Gu et al. 2017; Liu et al. 2017; Liu et al. 2017b) demonstrated that adversaries can generate “backdoors” into neural networks by poisoning the training data

- Models with backdoors perform well on standard training and validation samples
- but behaves badly on specific attacker-chosen inputs

In [1]:

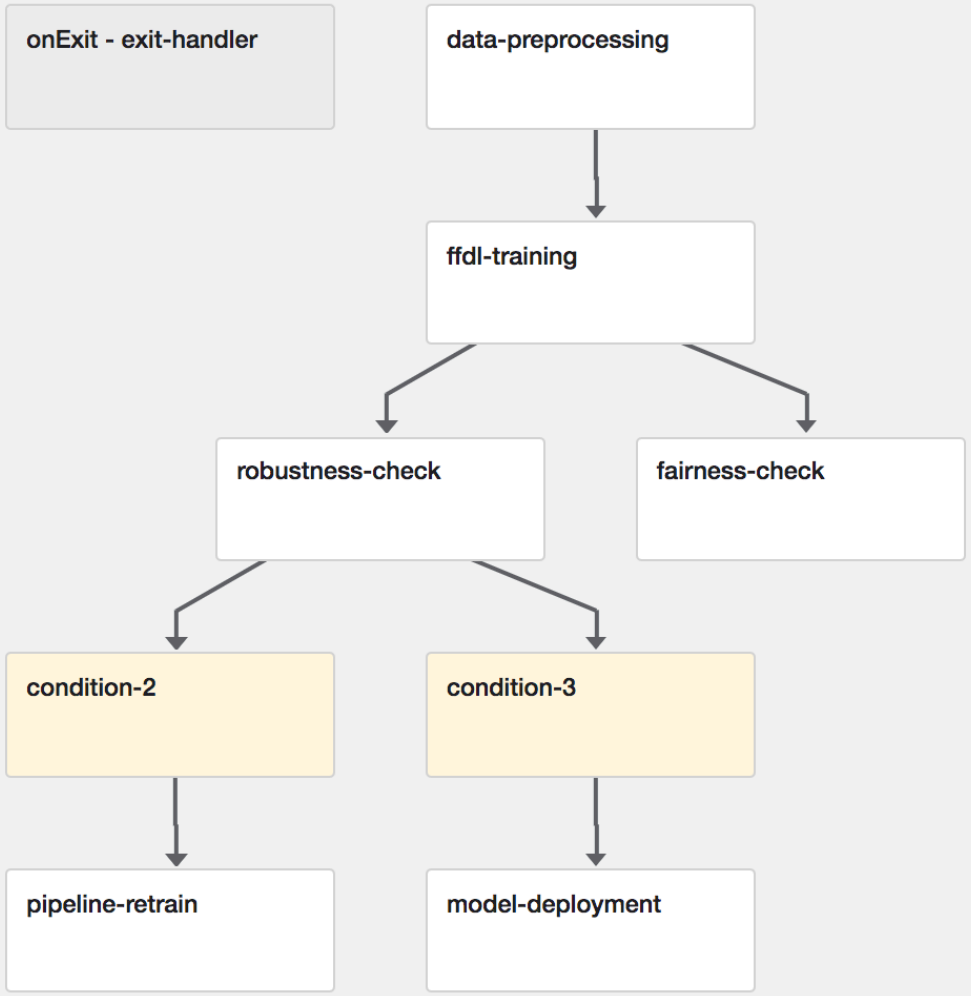
```
from __future__ import absolute_import, division, print_function, u

import os, sys
from os.path import abspath

module_path = os.path.abspath(os.path.join('..'))
```

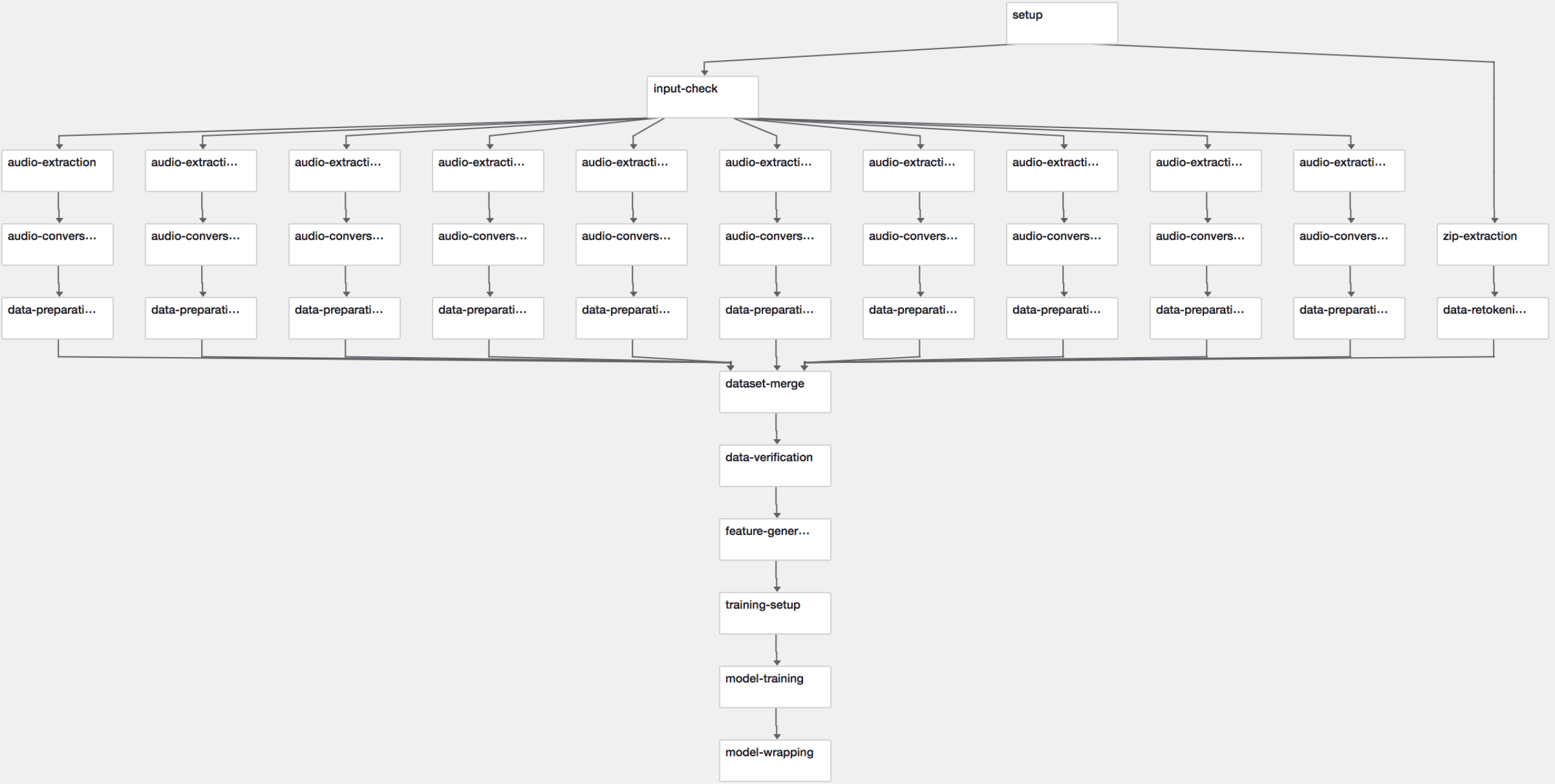
Pipelines

End-to-end pipeline with fairness and robustness test

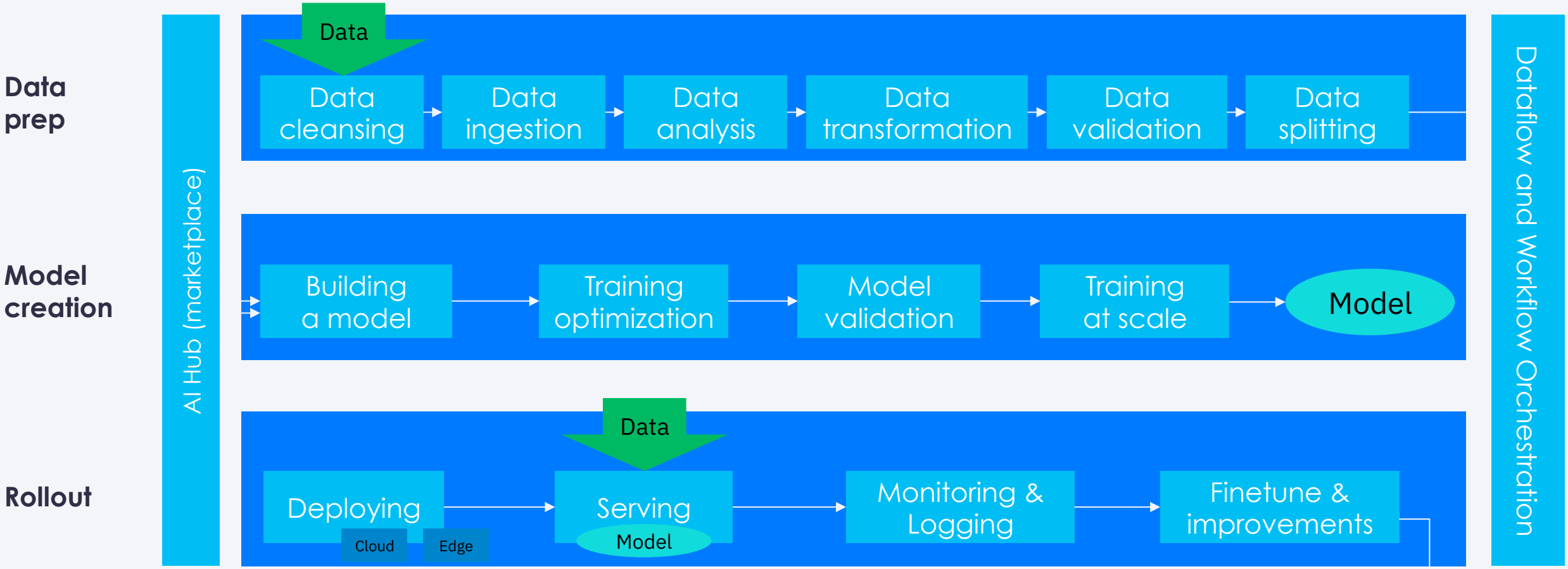
+ Create run + Create an experiment

Graph Source

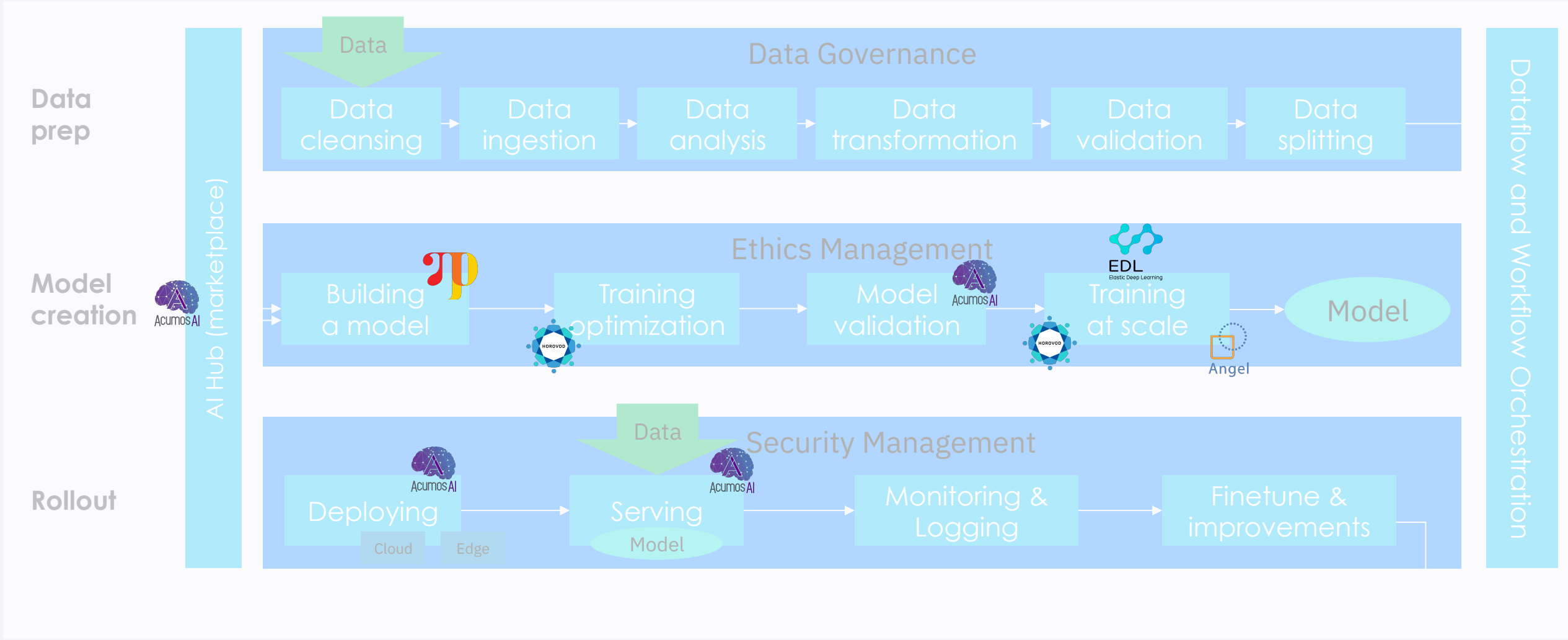



setup
input-check
audio-extraction
audio-extracti…
audio-extracti…
audio-extracti…
audio-extracti…
audio-extracti…
audio-extracti…
audio-extracti…
audio-extracti…
audio-extracti…
audio-convers…
audio-convers…
audio-convers…
audio-convers…
audio-convers…
audio-convers…
audio-convers…
audio-convers…
audio-convers…
audio-convers…
zip-extraction
data-preparati…
data-preparati…
data-preparati…
data-preparati…
data-preparati…
data-preparati…
data-preparati…
data-preparati…
data-preparati…
data-preparati…
data-retokeni…
dataset-merge
data-verification
feature-gener…
training-setup
model-training
model-wrapping

Machine Learning Stack



LFAI Machine Learning Stack



Open Source Machine Learning Stack