

Your Speaker Today:

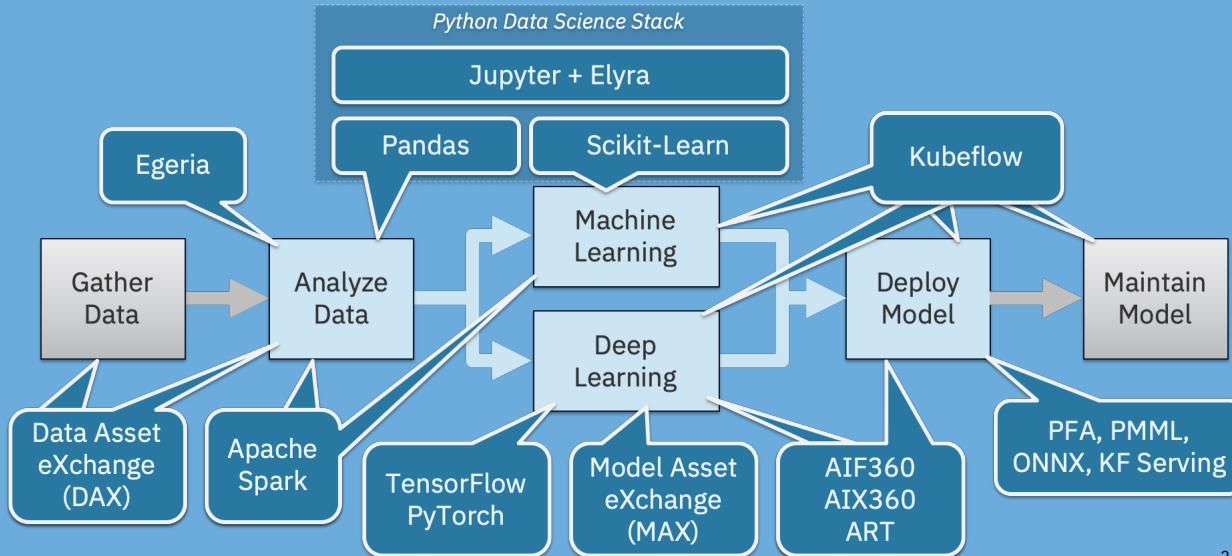


Animesh Singh

STSM and Chief Architect - Data and AI Open Source Platform

- CTO, IBM RedHat Data and AI Open Source Alignment
- IBM Kubeflow Engagement Lead, Kubeflow Committer
- Chair, Linux Foundation AI - Trusted AI
- Chair, CD Foundation MLOps Sig
- Ambassador, CNCF
- Member of IBM Academy of Technology (IBM AoT)

CODAIT



Kubeflow
github.com/kubeflow



Kubeflow: Current IBM Contributors



Christian Kadner



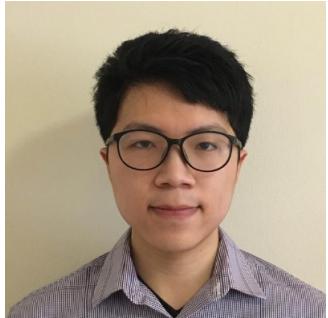
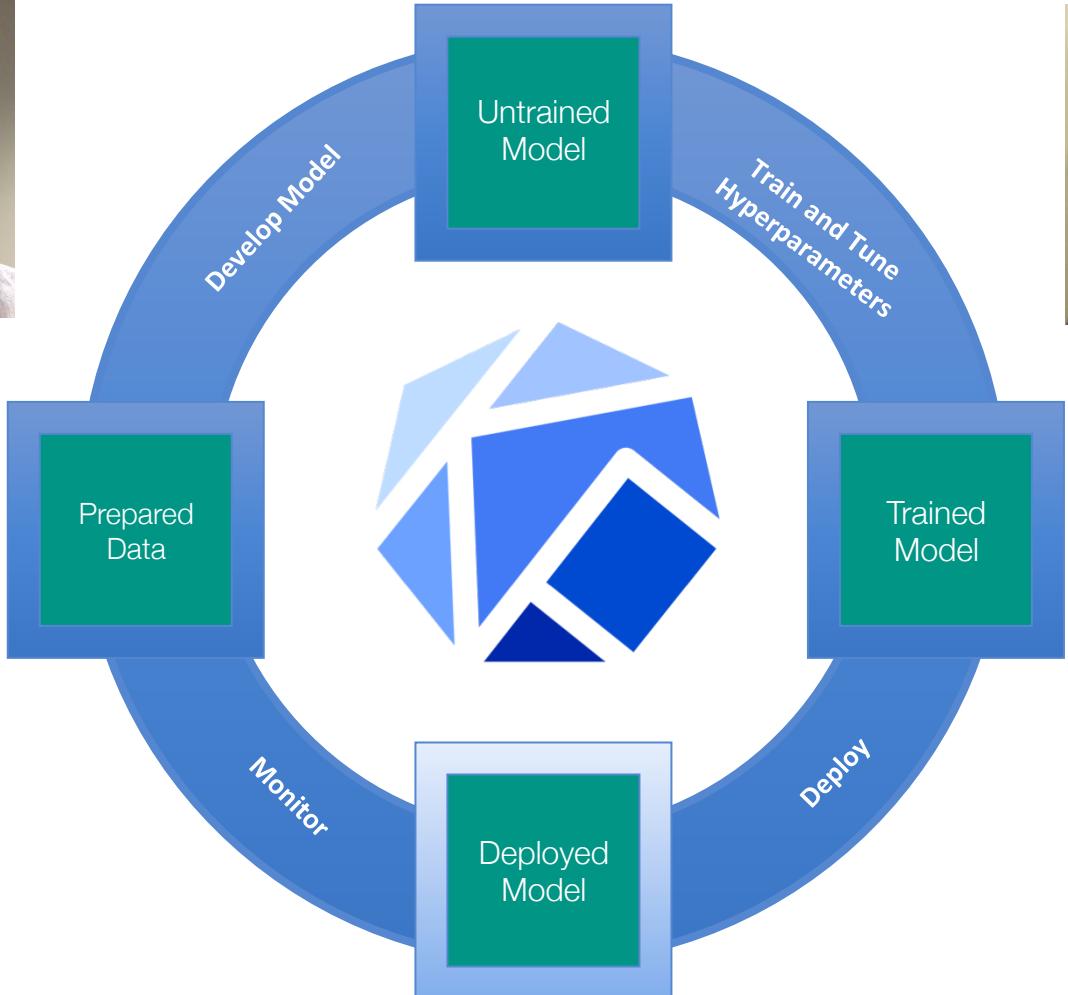
Weiqiang Zhuang



Jin Chi He



Feng Li



Tommy Li



Andrew Butler



Ke Zhu



Kevin Yu

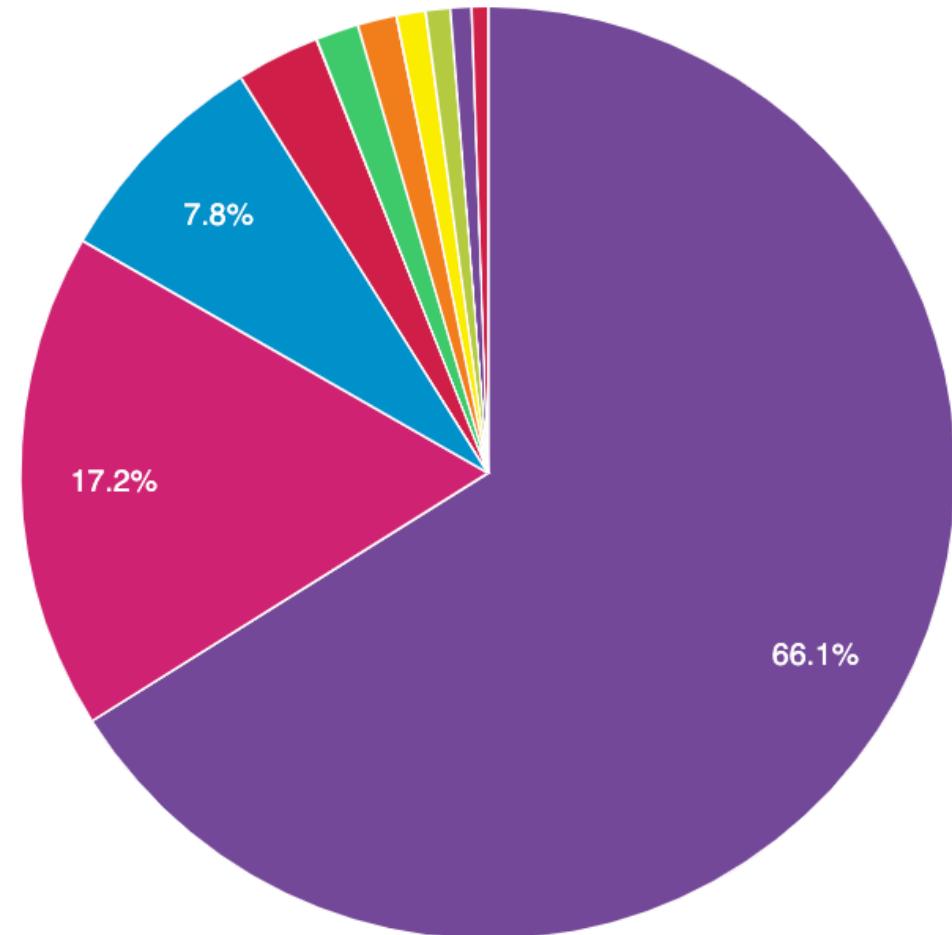


Commits by Company

Show 10 entries

Search

#	Company	Commits
	*independent	6882
1	Google	1792
2	IBM	816
3	Caicloud	301
4	Alibaba	141
5	Intel	105
6	Bloomberg LP	89
7	Red Hat	75
8	Huawei	59
9	Amazon	27



Showing 1 to 10 of 40 entries

Previous

Next

IBM is the 2nd Largest Contributor

Rank	Contributor	Count
1	Google	22064
2	IBM	4727
3	Cisco	4009
4	Caicloud	1865
5	Amazon	1425
6	Microsoft	553
7	Seldon	449
8	Net EASE	266
9	NetEase	260
10	Arrikto	213
11	DaoCloud	143
12	Huawei	139
13	NVidia	80
14	Oracle	78
15	Alibaba	70
16	Dell	63
17	Red Hat	52
18	Intel	50



Show 10 entries

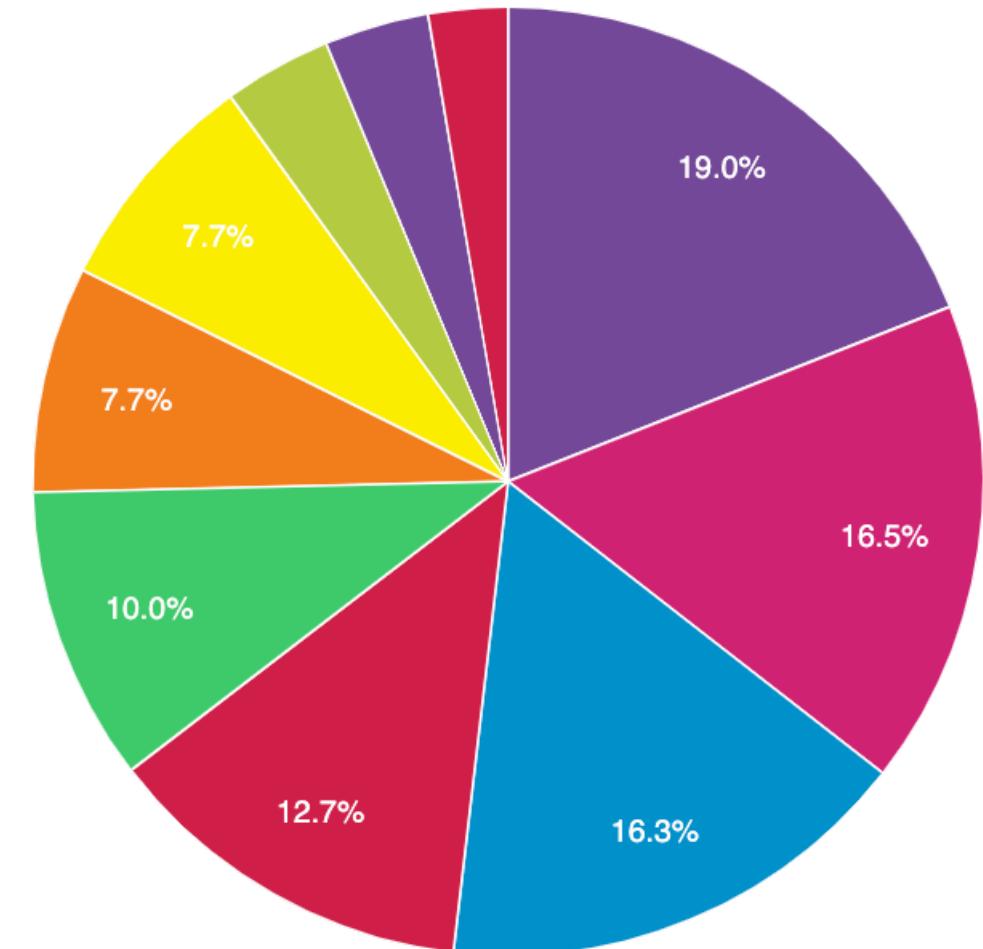
Search

#	Module	Commits
1	kfserving@kubeflow	155
2	kfp-tekton@kubeflow	135
3	katib@kubeflow	133
4	website@kubeflow	104
5	fairing@kubeflow	63
6	pipelines@kubeflow	63
7	examples@kubeflow	30
8	kfctl@kubeflow	29
9	kubeflow	22
10	manifests@kubeflow	21

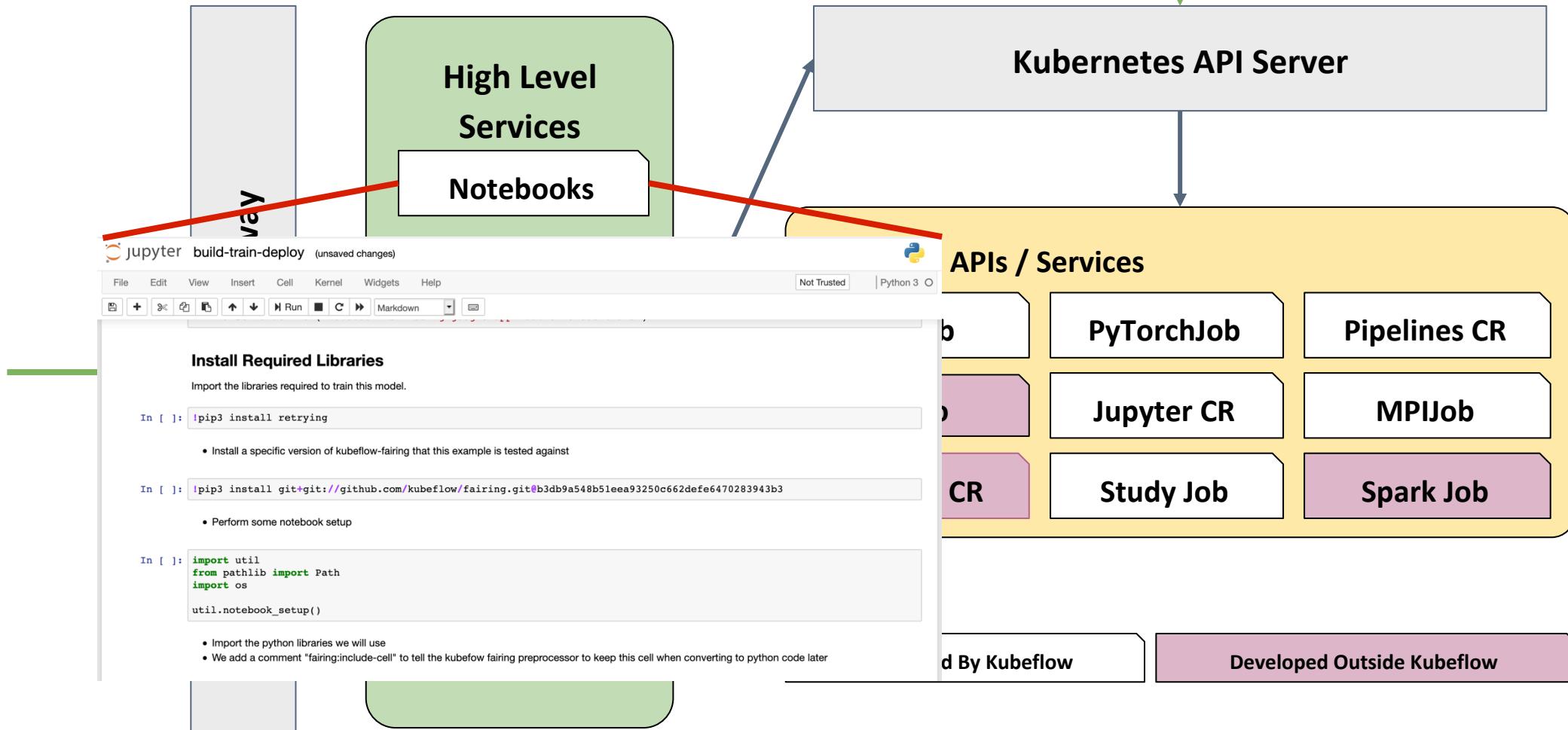
Showing 1 to 10 of 18 entries

Previous

Next



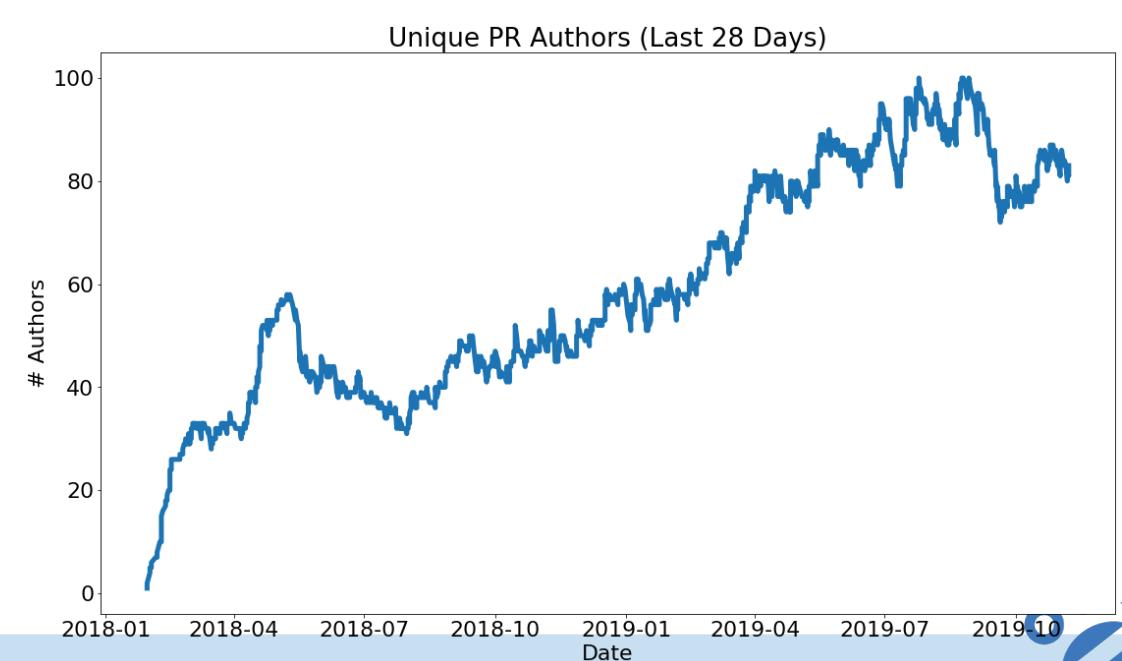
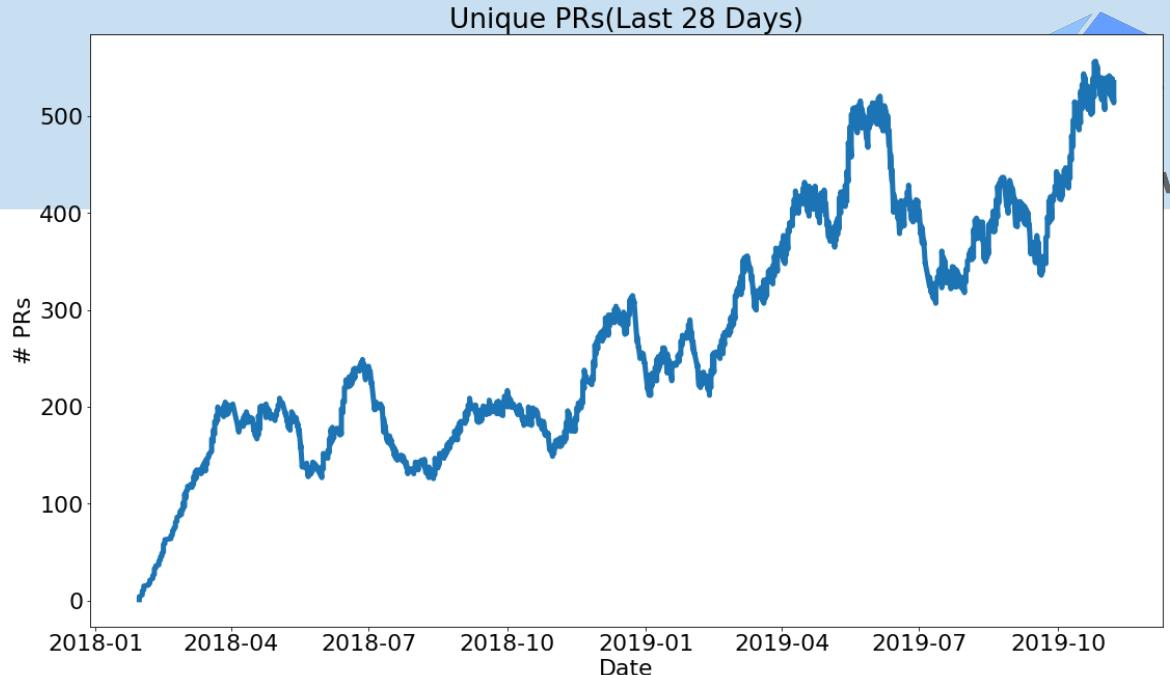
kubectl apply -f tfjob



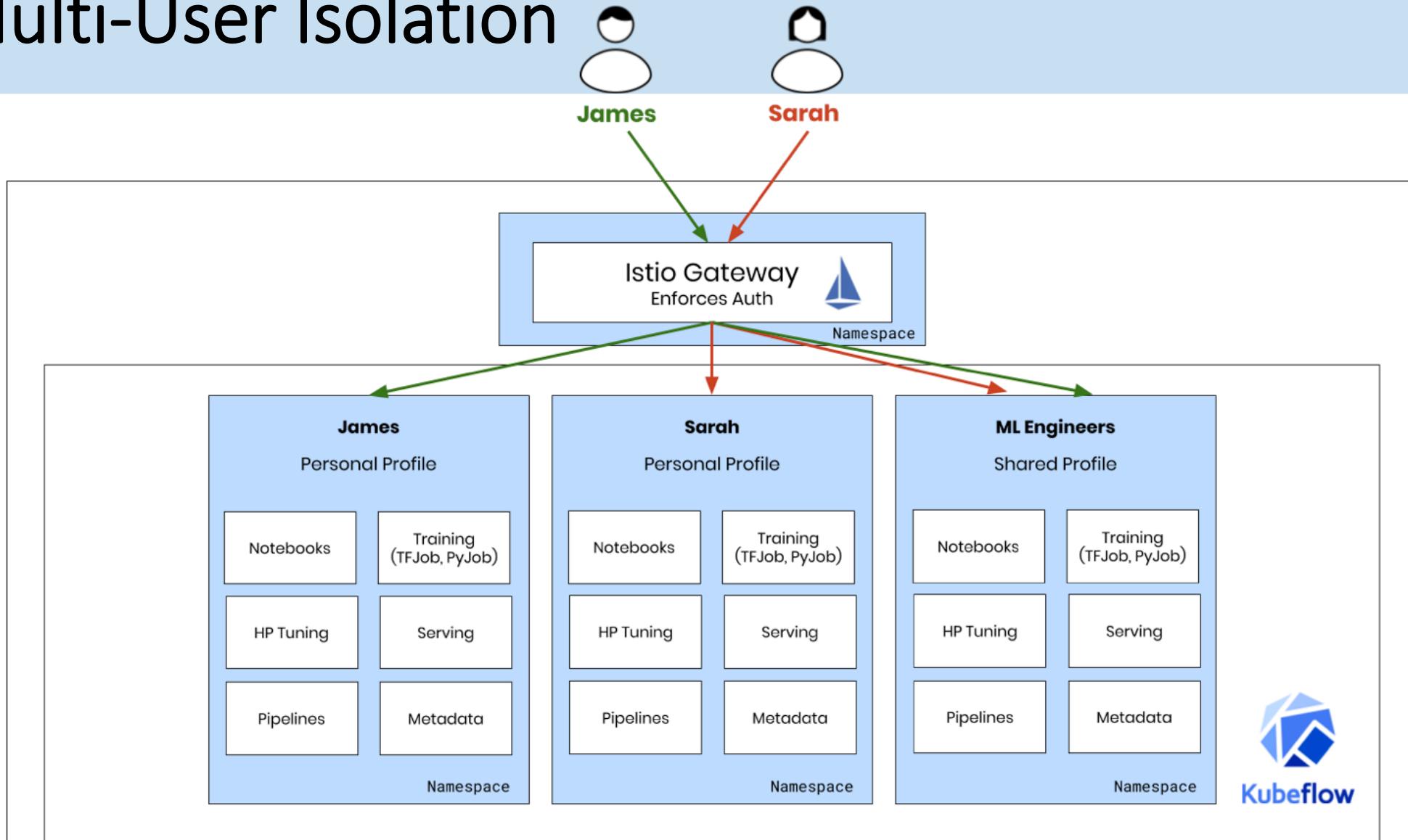
Adapted from Kubeflow Contributor Summit 2019 talk: Kubeflow and ML Landscape (Not all components are shown)



Community is growing!



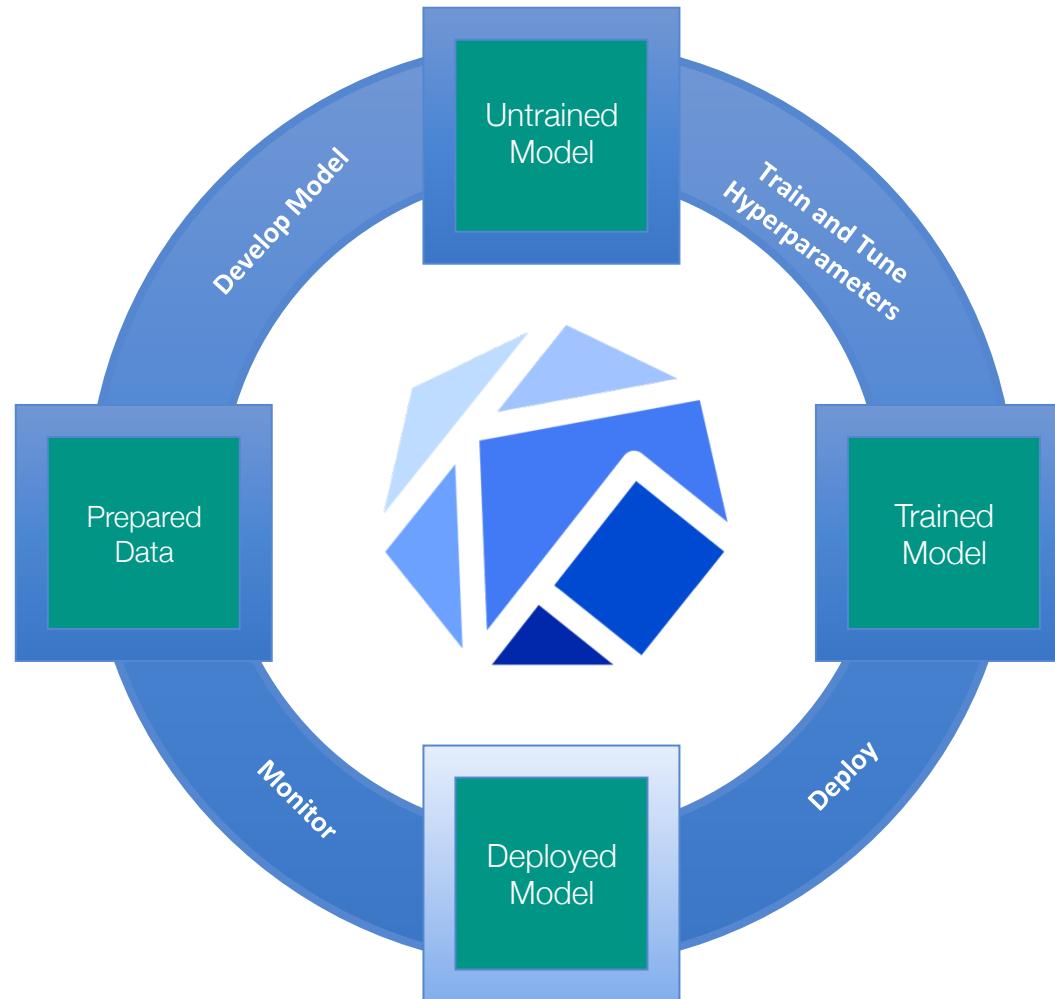
Multi-User Isolation



kubernetes



ML Lifecycle: Build: Development, Training and HPO



- Data Scientist
 - Self-service Jupyter Notebooks provide faster model experimentation
 - Simplified configuration of CPU/GPU, RAM, Persistent Volumes
 - Faster model creation with training operators, TFX, magics, workflow automation (Kale, Fairing)
 - Simplify access to external data sources (using stored secrets)
 - Easier protection, faster restoration & sharing of “complete” notebooks

- IT Operator
 - Profile Controller, Istio, Dex enable secure RBAC to notebooks, data & resources
 - Smaller base container images for notebooks, fewer crashes, faster to recover



Kubeflow

Home

Pipelines

Notebook Servers

Katib

Artifact Store

GitHub

Documentation

Privacy • Usage Reporting

Select namespace ▾

Dashboard Activity

Quick shortcuts

-  **Upload a pipeline**
Pipelines
-  **View all pipeline runs**
Pipelines
-  **Create a new Notebook server**
Notebook Servers
-  **View Katib Studies**
Katib
-  **View Metadata Artifacts**
Artifact Store

Recent Notebooks

Choose a namespace to see Notebooks

Recent Pipelines

-  **refarch-reefer-ml**
Created 6/29/2020, 10:04:11 AM
-  **[Tutorial] DSL - Control structures**
Created 6/10/2020, 2:24:18 PM
-  **[Tutorial] Data passing in python components**
Created 6/10/2020, 2:24:17 PM
-  **[Demo] TFX - Taxi Tip Prediction Model Trainer**
Created 6/10/2020, 2:24:16 PM
-  **[Demo] XGBoost - Training with Confusion Matrix**
Created 6/10/2020, 2:24:15 PM

Recent Pipeline Runs

Documentation

Getting Started with Kubeflow
Get your machine-learning workflow up and running on Kubeflow 

MinikF
A fast and easy way to deploy Kubeflow locally 

Microk8s for Kubeflow
Quickly get Kubeflow running locally on native hypervisors 

Minikube for Kubeflow
Quickly get Kubeflow running locally 

Kubeflow on GCP
Running Kubeflow on Kubernetes Engine and Google Cloud Platform 

Kubeflow on AWS
Running Kubeflow on Elastic Container Service and Amazon Web Services 

Requirements for Kubeflow
Get more detailed information about using Kubeflow and its components 

	TF Operator	PyTorch Operator	MPI Operator
Framework Support	 TensorFlow	 PyTorch	 TensorFlow/Keras Apache MXNet/PyTorch/OpenMPI
Distribution Strategy & Backend	<code>tf.distribute</code> MPI/NCCL/PS/TPU	<code>torch.distributed</code> Gloo/MPI/NCCL	<code>horovod</code> DistributedOptimizer Gloo/MPI/NCCL



Distributed Training Operators



tf-operator

Tools for ML/Tensorflow on Kubernetes.

● Jsonnet ⚙ Apache-2.0 323 ⭐

pytorch-operator

PyTorch on Kubernetes

● Jsonnet ⚙ Apache-2.0 87 ⭐ 19

mpi-operator

Kubernetes Operator for Allreduce-style

kubernetes tensorflow mpi dist

horovod kubeflow

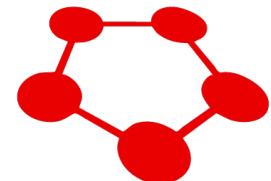
● Go ⚙ Apache-2.0 83 ⭐ 125

xgboost-operator

Incubating project for xgboost operator

● Go ⚙ Apache-2.0 23 ⭐ 41

XGBoost



Chainer

mxnet-operator

A Kubernetes operator for mxnet jobs

● Go ⚙ Apache-2.0 20 ⭐ 50

chainer-operator

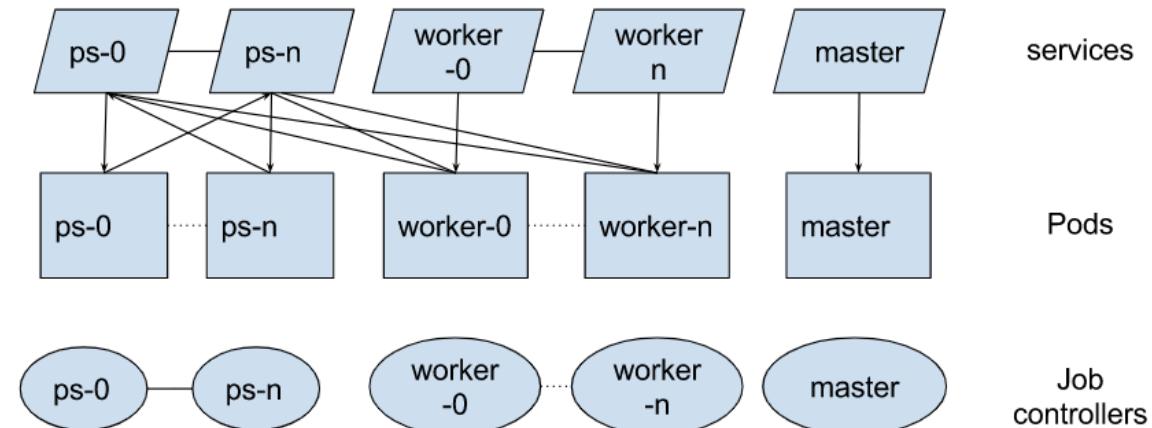
Repository for chainer operator

● Go ⚙ Apache-2.0 9 ⭐ 12



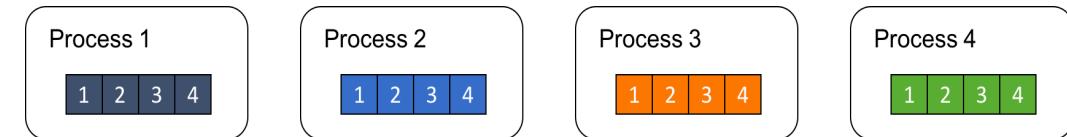
Distributed Tensorflow Operator

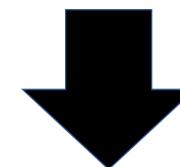
- A distributed Tensorflow Job is collection of the following processes
 - Chief – The chief is responsible for orchestrating training and performing tasks like checkpointing the model
 - Ps – The ps are parameters servers; the servers provide a distributed data store for the model parameters to access
 - Worker – The workers do the actual work of training the model. In some cases, worker 0 might also act as the chief
 - Evaluator - The evaluators can be used to compute evaluation metrics as the model is trained

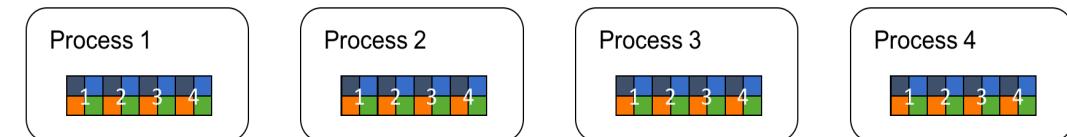


Distributed MPI Operator - AllReduce

- AllReduce is an operation that reduces many arrays spread across multiple processes into a single array which can be returned to all the processes
- This ensures consistency between distributed processes while allowing all of them to take on different workloads
- The operation used to reduce the multiple arrays back into a single array can vary and that is what makes the different options for AllReduce



 AllReduce



IBM Hyper Parameter Optimization and Neural Architecture Search - Katib

- Katib: Kubernetes Native System for Automated tuning of machine learning model's Hyperparameter Tuning and Neural Architecture Search.
- Github Repository:
<https://github.com/kubeflow/katib>
- Hyperparameter Tuning
 - [Random Search](#)
 - [Tree of Parzen Estimators \(TPE\)](#)
 - [Grid Search](#)
 - [Hyperband](#)
 - [Bayesian Optimization](#)
 - [CMA Evolution Strategy](#)
- Neural Architecture Search
 - [Efficient Neural Architecture Search \(ENAS\)](#)
 - [Differentiable Architecture Search \(DARTS\)](#)





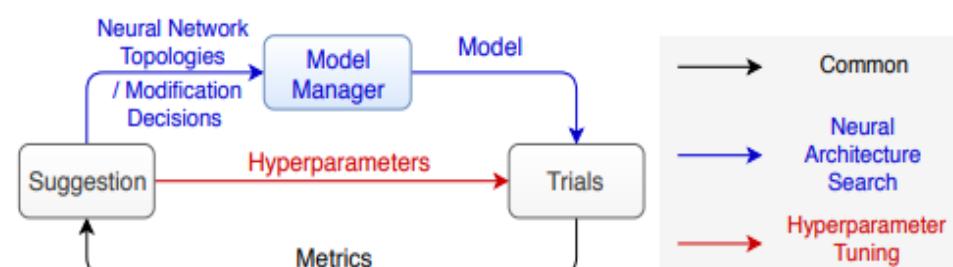
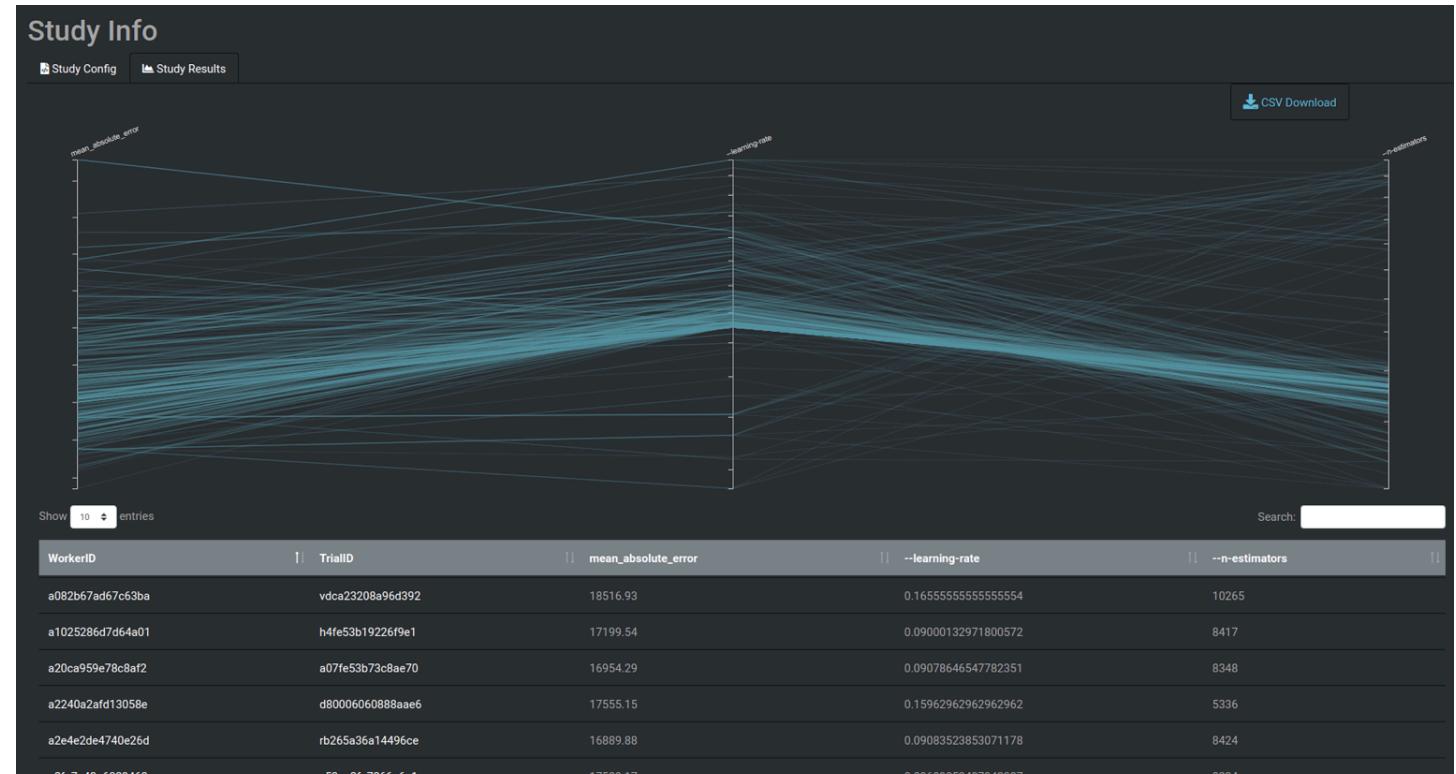


Figure 1: Summary of AutoML workflows

≡ Katib

Welcome to Katib

Choose type of experiment

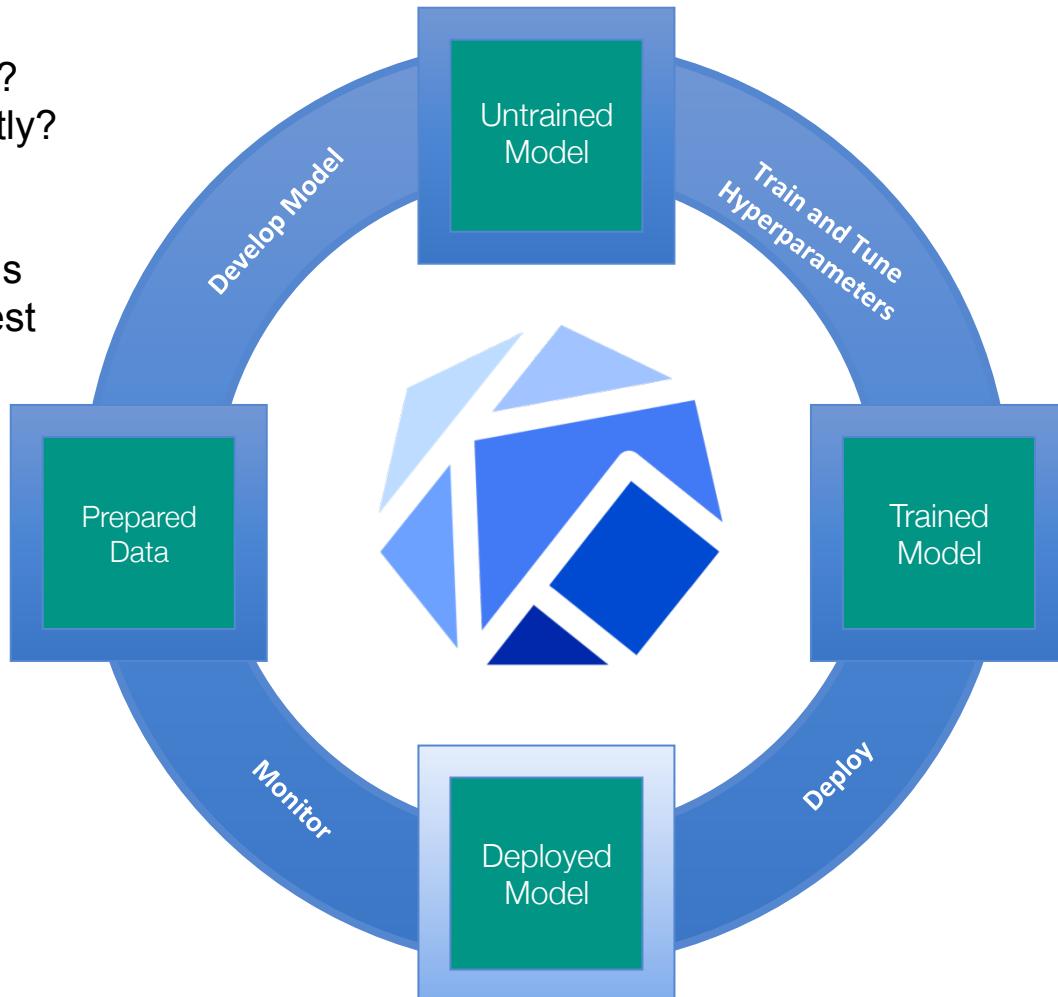
Hyperparameter
Tuning

Neural Architecture
Search

For usage instructions, see the [Kubeflow docs](#)

To contribute to Katib, visit [GitHub](#)

- 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?



- How do I handle batch predictions?
- How do I leverage standardized Data Plane protocol so that I can move my model across MLServing platforms?
- 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? Bias detection? Adversarial Detection?
- How do I wire up custom pre and post processing

- Seldon Core was pioneering Graph Inferencing.
- IBM and Bloomberg were exploring serverless ML lambdas. IBM gave a talk on the ML Serving with Knative at last KubeCon in Seattle
- Google had built a common Tensorflow HTTP API for models.
- Microsoft Kuberntizing their Azure ML Stack



Bloomberg





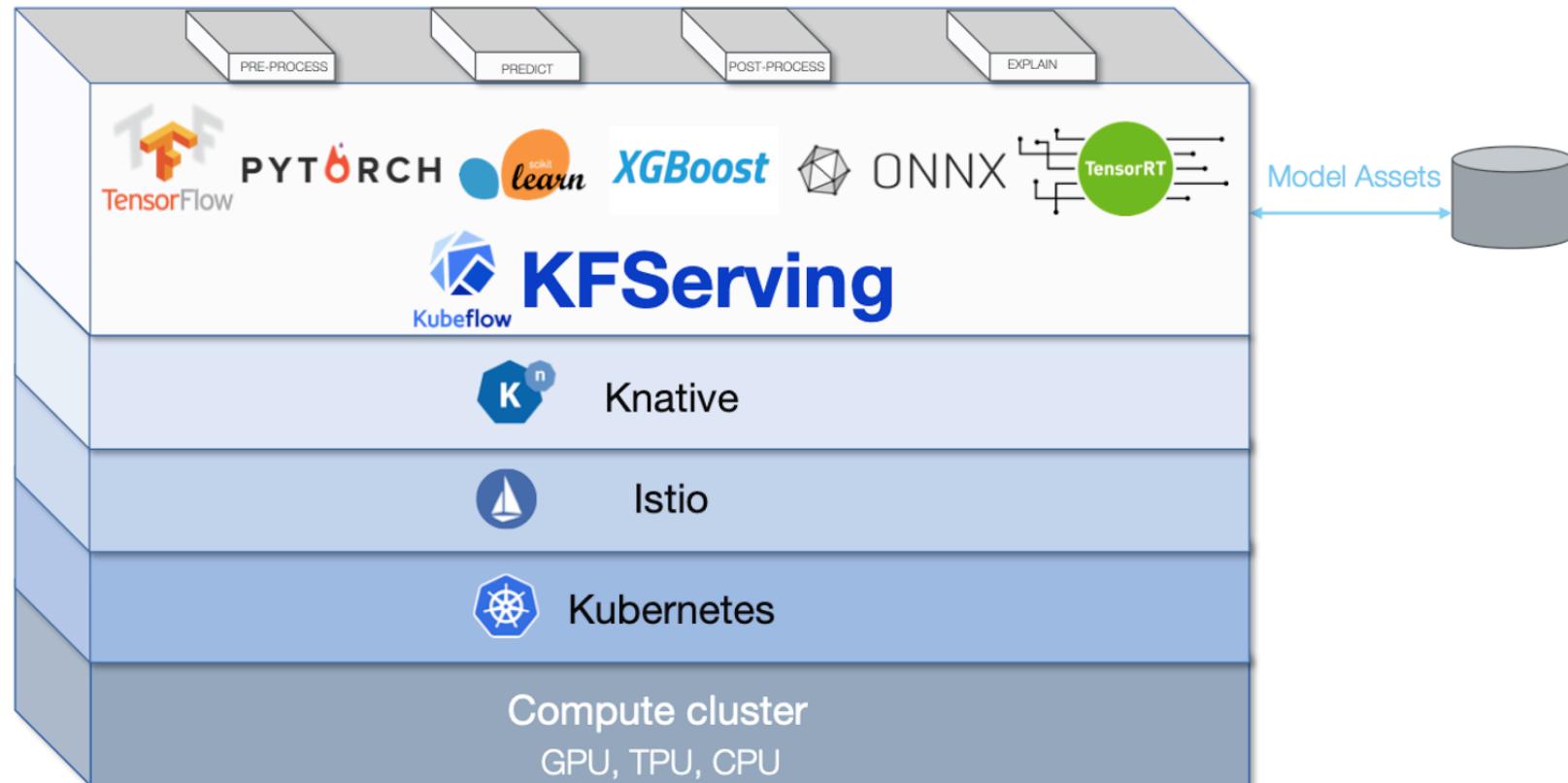
Putting the pieces together



- Kubeflow created the conditions for collaboration.
- A promise of open code and open community.
- Shared responsibilities and expertise across multiple companies.
- Diverse requirements from different customer segments



- Founded by Google, Seldon, IBM, Bloomberg and Microsoft
- Part of the Kubeflow project
- Focus on 80% use cases - single model rollout and update
- Kfserving 1.0 goals:
 - Serverless ML Inference
 - Canary rollouts
 - Model Explanations
 - Optional Pre/Post processing

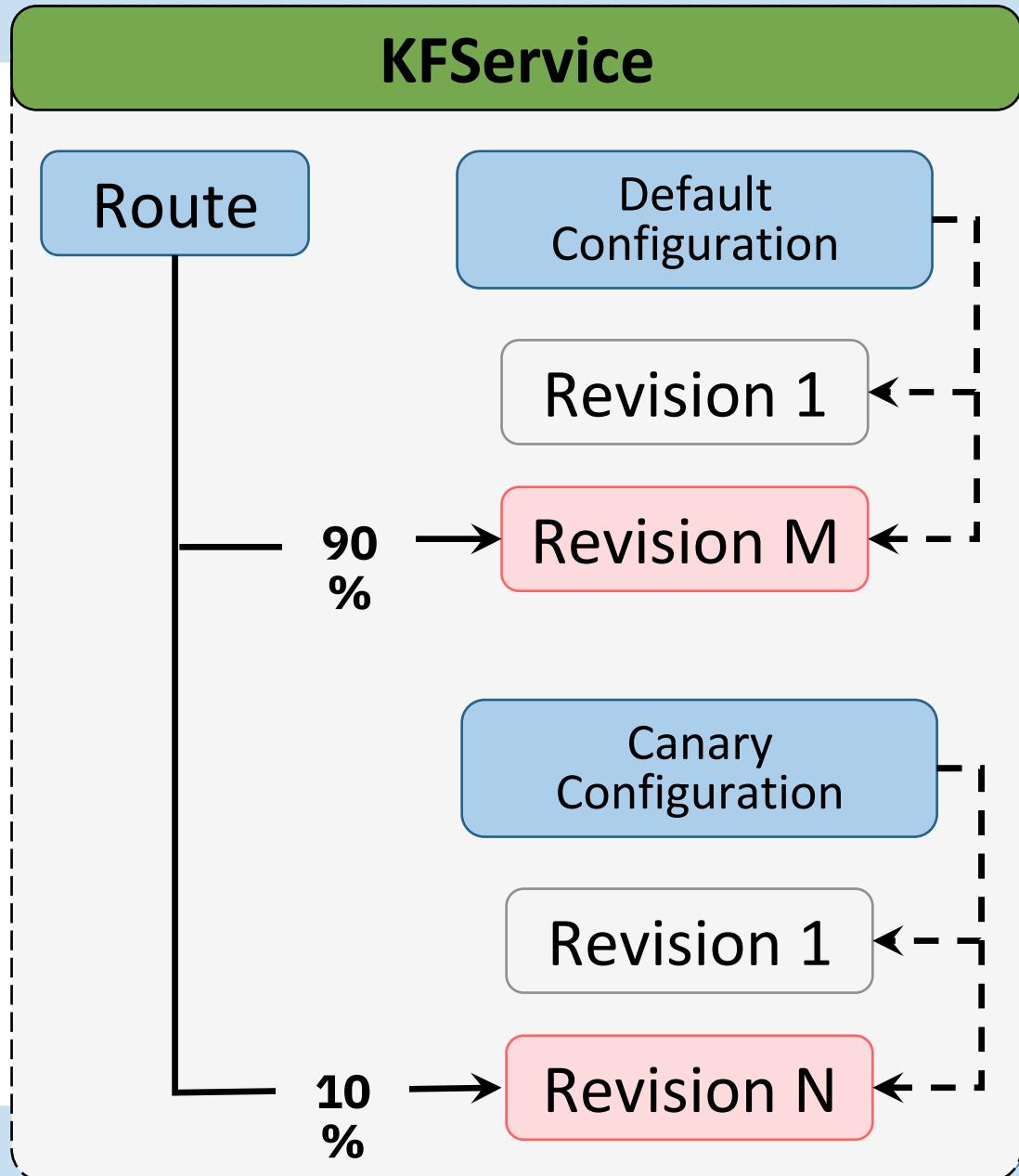




IBM KFServing: Default and Canary Configurations

Manages the hosting aspects of your models

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



Model Servers

- TensorFlow
- Nvidia TRTIS
- PyTorch
- XGBoost
- SKLearn
- ONNX

Components:

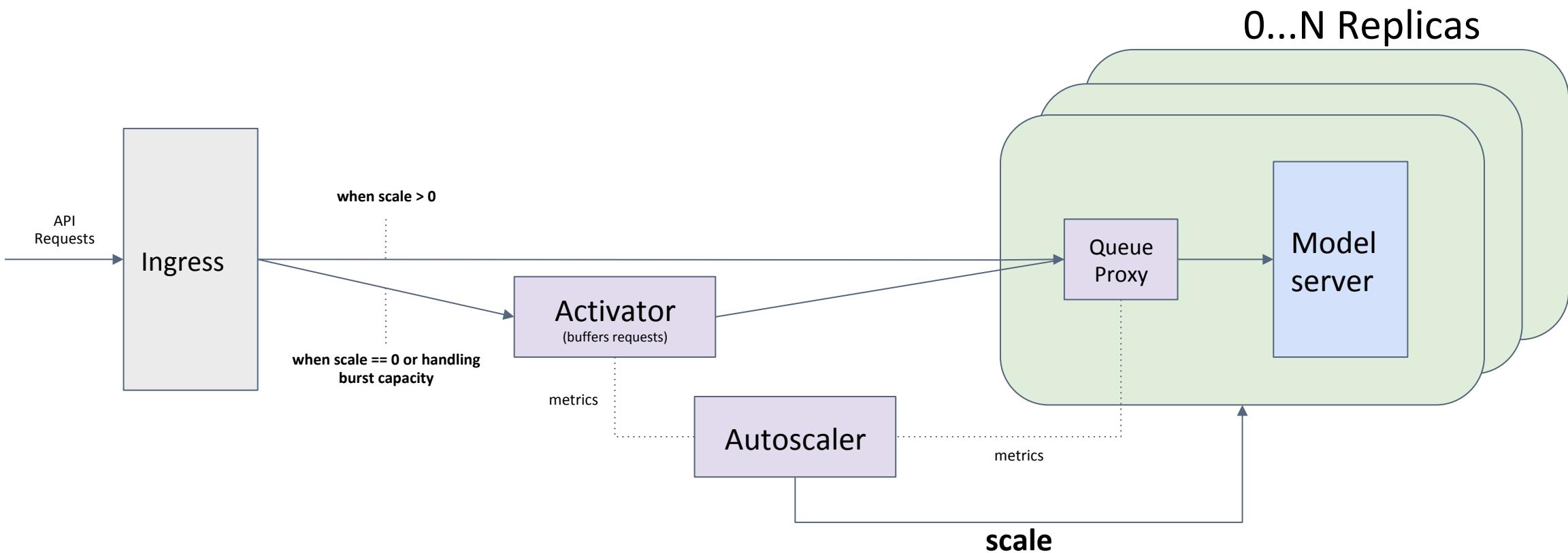
- - Predictor, Explainer, Transformer (pre-processor, post-processor)

Storage

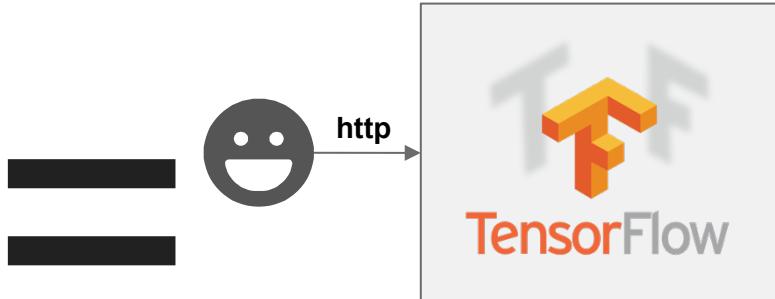
- AWS/S3
- GCS
- Azure Blob
- PVC



- Scale based on # in-flight requests against expected concurrency
- Simple solution for heterogeneous ML inference autoscaling



```
apiVersion: "serving.kubeflow.org/v1alpha2"
kind: "InferenceService"
metadata:
  name: "flowers-sample"
spec:
  default:
    predictor:
      tensorflow:
        storageUri: "gs://kfserving-samples/models/tensorflow/flowers"
```



- A pointer to a Serialized Model File
- 9 lines of YAML
- A live model at an HTTP endpoint

- Scale to Zero
- GPU Autoscaling
- Safe Rollouts
- Optimized Serving Containers
- Network Policy and Auth
- HTTP APIs (gRPC soon)
- Tracing
- Metrics

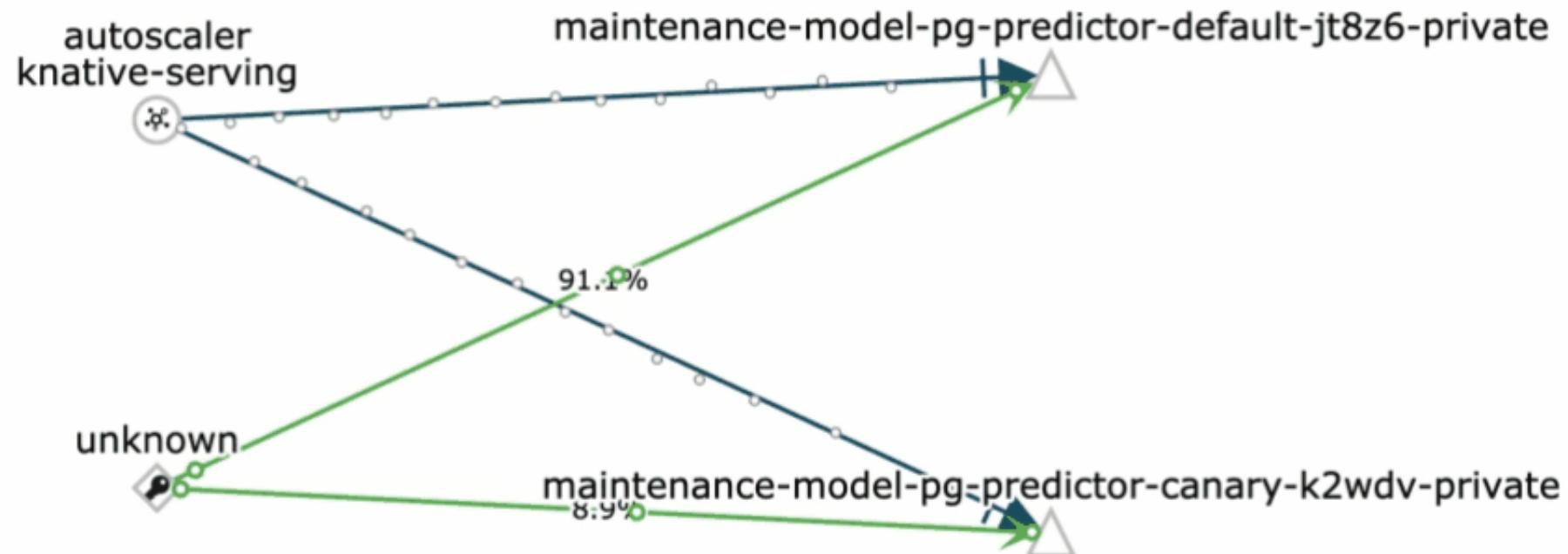
Production users include:

Bloomberg



gojek





KFServing – Existing Features

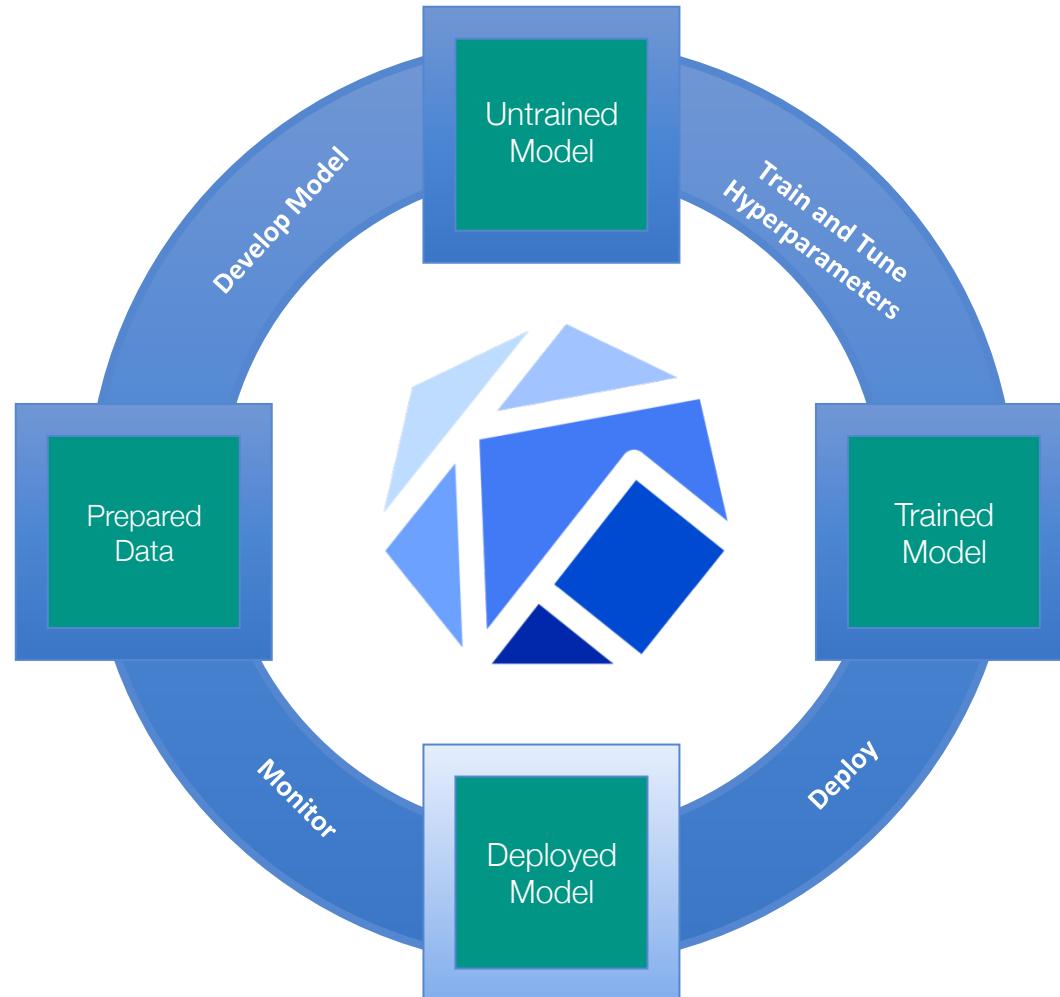
- ❑ Crowd sourced capabilities – Contributions by AWS, Bloomberg, Google, Seldon, IBM, NVidia and others.
- ❑ Support for multiple runtimes pre-integrated (TFServing, Nvidia Triton (GPU optimization), ONNX Runtime, SKLearn, PyTorch, XGBoost, Custom models).
- ❑ Serverless ML Inference and Autoscaling: Scale to zero (with no incoming traffic) and Request queue based autoscaling
- ❑ Canary and Pinned rollouts: Control traffic percentage and direction, pinned rollouts
- ❑ Pluggable pre-processor/post-processor via Transformer: Gives capabilities to plug in pre-processing/post-processing implementation, control routing and placement (e.g. pre-processor on CPU, predictor on GPU)
- ❑ Pluggable analysis algorithms: Explainability, Drift Detection, Anomaly Detection, Adversarial Detection (contributed by Seldon) enabled by Payload Logging (built using CloudEvents standardized eventing protocol)
- ❑ Batch Predictions: Batch prediction support for ML frameworks (TensorFlow, PyTorch, ...)
- ❑ Integration with existing monitoring stack around Knative/Istio ecosystem: Kiali (Service placements, traffic and graphs), Jaeger (request tracing), Grafana/Prometheus plug-ins for Knative)
- ❑ Multiple clients: kubectl, Python SDK, Kubeflow Pipelines SDK
- ❑ Standardized Data Plane V2 protocol for prediction/explainability et all: Already implemented by Nvidia Triton

KFServing – Upcoming Features

- ❑ MMS: Multi-Model-Serving for serving multiple models per custom KFService instance
- ❑ More Data Plane v2 API Compliant Servers: SKLearn, XGBoost, PyTorch...
- ❑ Multi-Model-Graphs and Pipelines: Support chaining multiple models together in a Pipelines
- ❑ PyTorch support via AWS TorchServe
- ❑ gRPC Support for all Model Servers
- ❑ Support for multi-armed-bandits
- ❑ Integration with IBM AIX360 for Explainability, AIF360 for Bias detection and ART for Adversarial detection

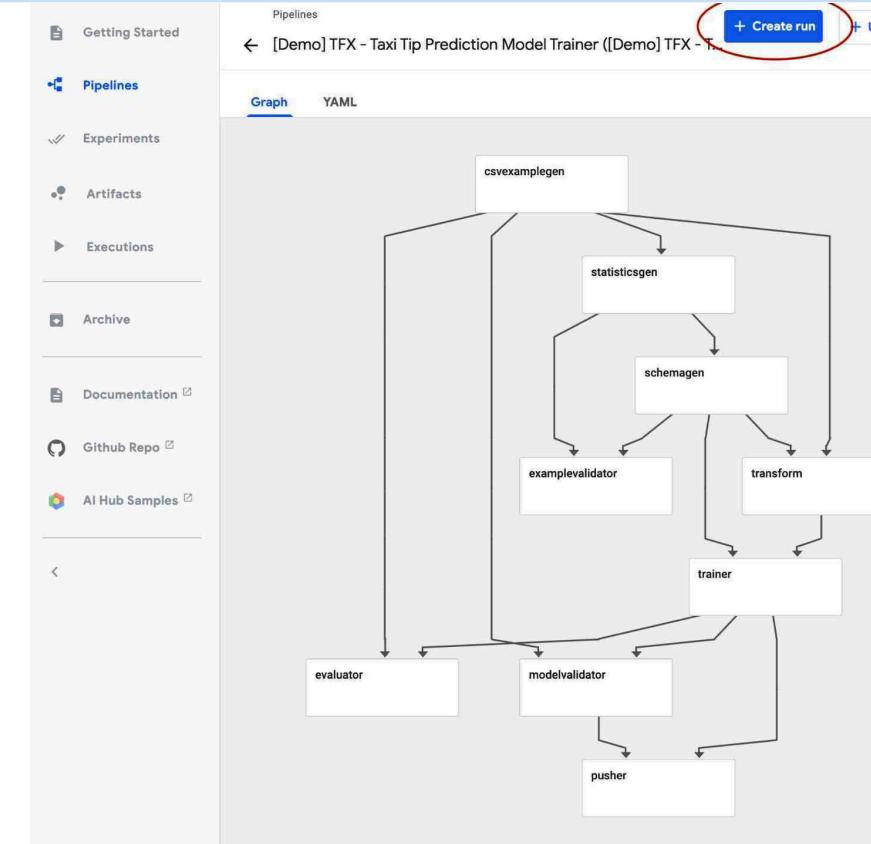


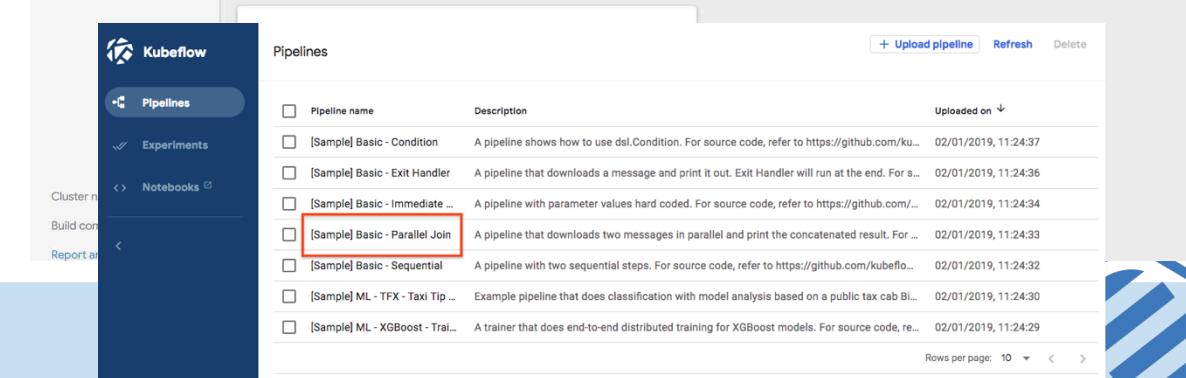
ML Lifecycle: Orchestrate Build, Train, Validate and Deploy



Kubeflow Pipelines

- 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





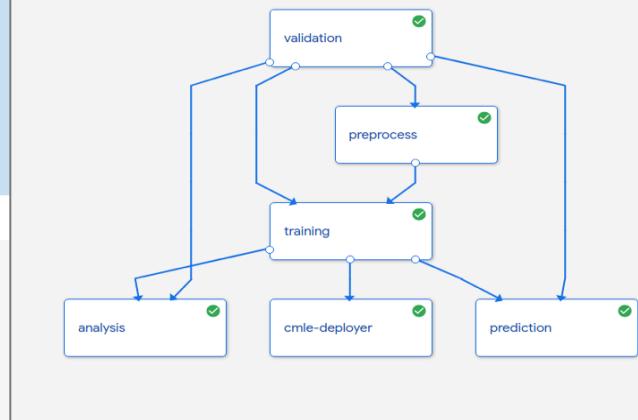
Pipeline name	Description	Uploaded on
[Sample] Basic - Condition	A pipeline shows how to use dsl.Condition. For source code, refer to https://github.com/ku...	02/01/2019, 11:24:37
[Sample] Basic - Exit Handler	A pipeline that downloads a message and print it out. Exit Handler will run at the end. For ...	02/01/2019, 11:24:36
[Sample] Basic - Immediate ...	A pipeline with parameter values hard coded. For source code, refer to https://github.com/...	02/01/2019, 11:24:34
[Sample] Basic - Parallel Join	A pipeline that downloads two messages in parallel and print the concatenated result. For ...	02/01/2019, 11:24:33
[Sample] Basic - Sequential	A pipeline with two sequential steps. For source code, refer to https://github.com/kubefo...	02/01/2019, 11:24:32
[Sample] ML - TFX - Taxi Tip ...	Example pipeline that does classification with model analysis based on a public tax cab Bl...	02/01/2019, 11:24:30
[Sample] ML - XGBoost - Trai...	A trainer that does end-to-end distributed training for XGBoost models. For source code, re...	02/01/2019, 11:24:29



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'})
```



Visualize the state of various components

Pipelines
Experiments **Artifacts**
Executions
Archive
Documentation
Github Repo
AI Hub Samples

Cluster name: cluster-4
Build commit: 743746b
Report an Issue

Graph Run output Config

csvexamplegen → statisticsgen → schemagen → examplevalidator → evaluator → pusher

resolvernode-lates... → evaluator

train → evaluator

Static HTML

Sort by Feature ▾ Reverse order Feature search (...)

Features: int(8) float(7) string(2)
 unknown(1)

Numeric Features (15)			
count	missing	mean	std dev
dropoff_census_tract 3,618	28.93%	17.0B	333k
dropoff_community_area 4,905	3.65%	21.2	17.85
dropoff_latitude 4,915	3.46%	41.9	0.04
dropoff_longitude 4,915	3.46%	-87.65	0.06

Runtime execution graph. Only steps that are currently running or have a

Pipelines

[+ Upload pipeline](#)[Refresh](#)[Delete](#)

Filter pipelines



<input type="checkbox"/>	Pipeline name	Description	Uploaded on
<input type="checkbox"/>	[Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstra...	2/20/2020, 3:28:12 PM
<input type="checkbox"/>	[Tutorial] Data passing in python com...	source code Shows how to pass data between python components.	2/20/2020, 3:28:11 PM
<input type="checkbox"/>	[Demo] TFX - Taxi Tip Prediction Mod...	source code GCP Permission requirements . Example pipeline that does classification with model analysis based on ...	2/20/2020, 3:28:10 PM
<input type="checkbox"/>	Version name		Uploaded on
<input type="checkbox"/>	TFX - Taxi Tip Prediction Model Trainer_version_at_2020-03-03T15:44:30.197Z		3/3/2020, 7:55:03 AM
<input type="checkbox"/>	[Demo] TFX - Taxi Tip Prediction Model Trainer		2/20/2020, 3:28:10 PM

Rows per page: 10 ▾ < >

Rows per page: 10 ▾ < >

Pipelines lets you group and manage multiple versions of a pipeline.



Artifacts

Pipeline/Workspace ↑	Name	ID	Type	URI	Created at
		1	ExternalArtifact	gs://ml-pipeline-playground/tfx_t...	2020-02-20T15:10:00Z
taxi_pipeline_with_parameters	examples	2	Examples	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	statistics	3	ExampleStatistics	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	schema	4	Schema	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	anomalies	5	ExampleAnomalies	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	transform_graph	6	TransformGraph	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	transformed_e...	7	Examples	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	model	8	Model	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	evaluation	9	ModelEvaluation	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	blessing	10	ModelBlessing	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	pushed_model	11	PushedModel	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z
	evaluation	12	ModelEvaluation	gs://aju-pipelines/tfx_taxi_simpl...	2020-02-20T15:10:00Z

model

Overview Lineage Explorer

Type: Model

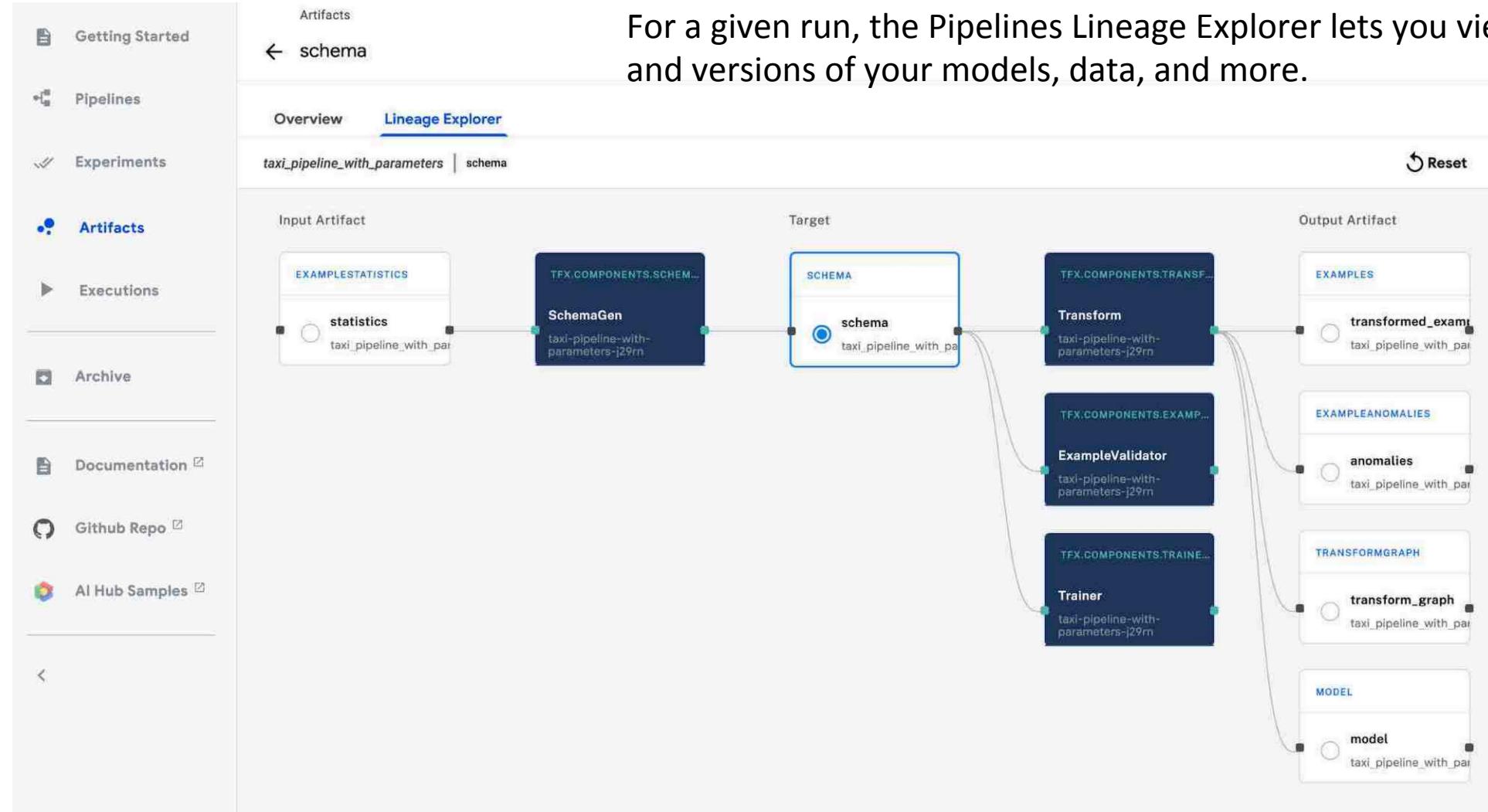
URI: gs://aju-pipelines/tfx_taxi_simple/85265540-6a06-4969-a49e-1f65741878be/Trainer/model/7

Properties

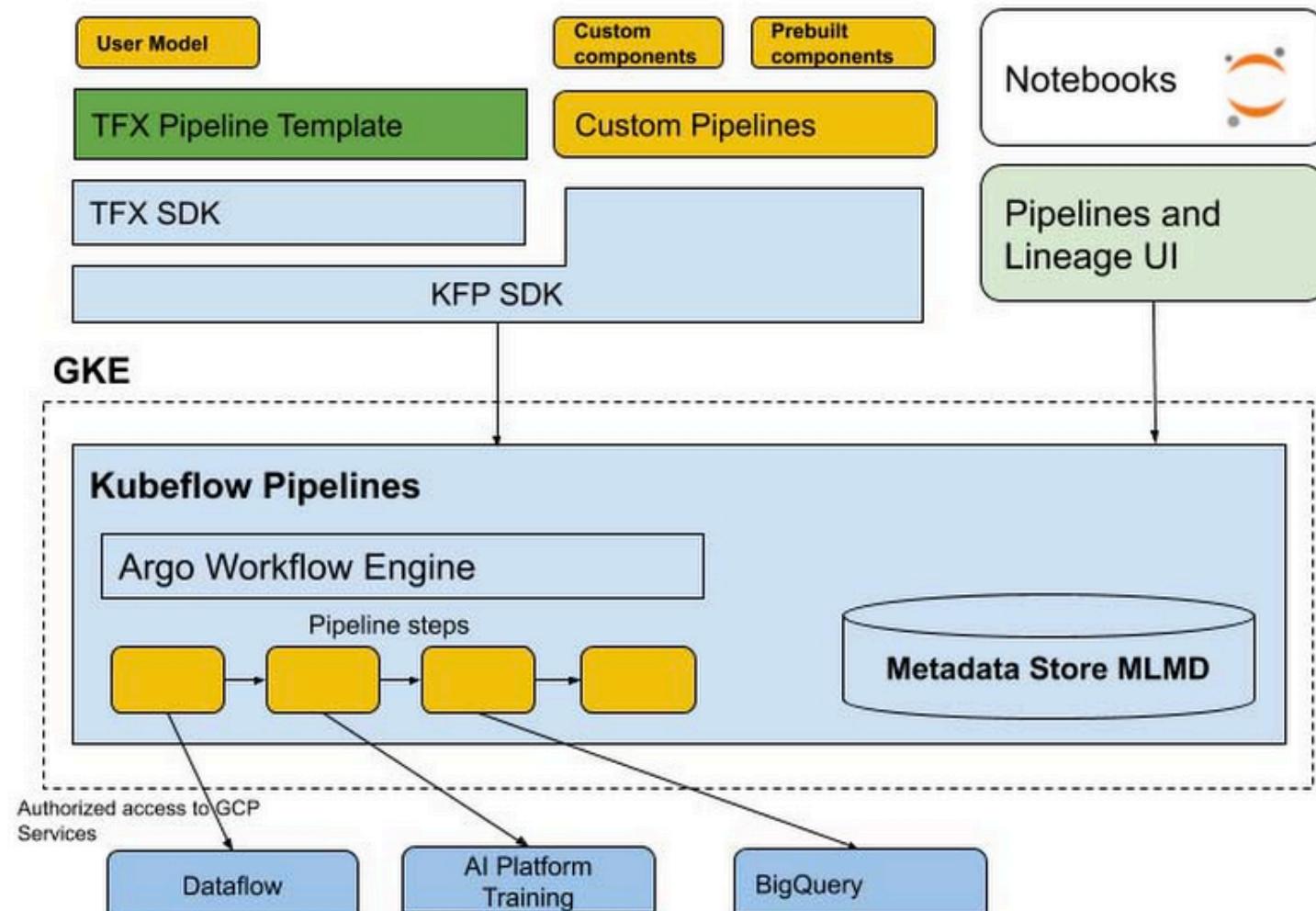
Custom Properties

name	pipeline_name	producer_component	state
model	taxi_pipeline_with_parameters	Trainer	published

Artifacts for a run of the “TFX Taxi Trip” example pipeline. For each artifact, you can view details and get the artifact URL—in this case, for the model.



Kubeflow Pipeline Architecture



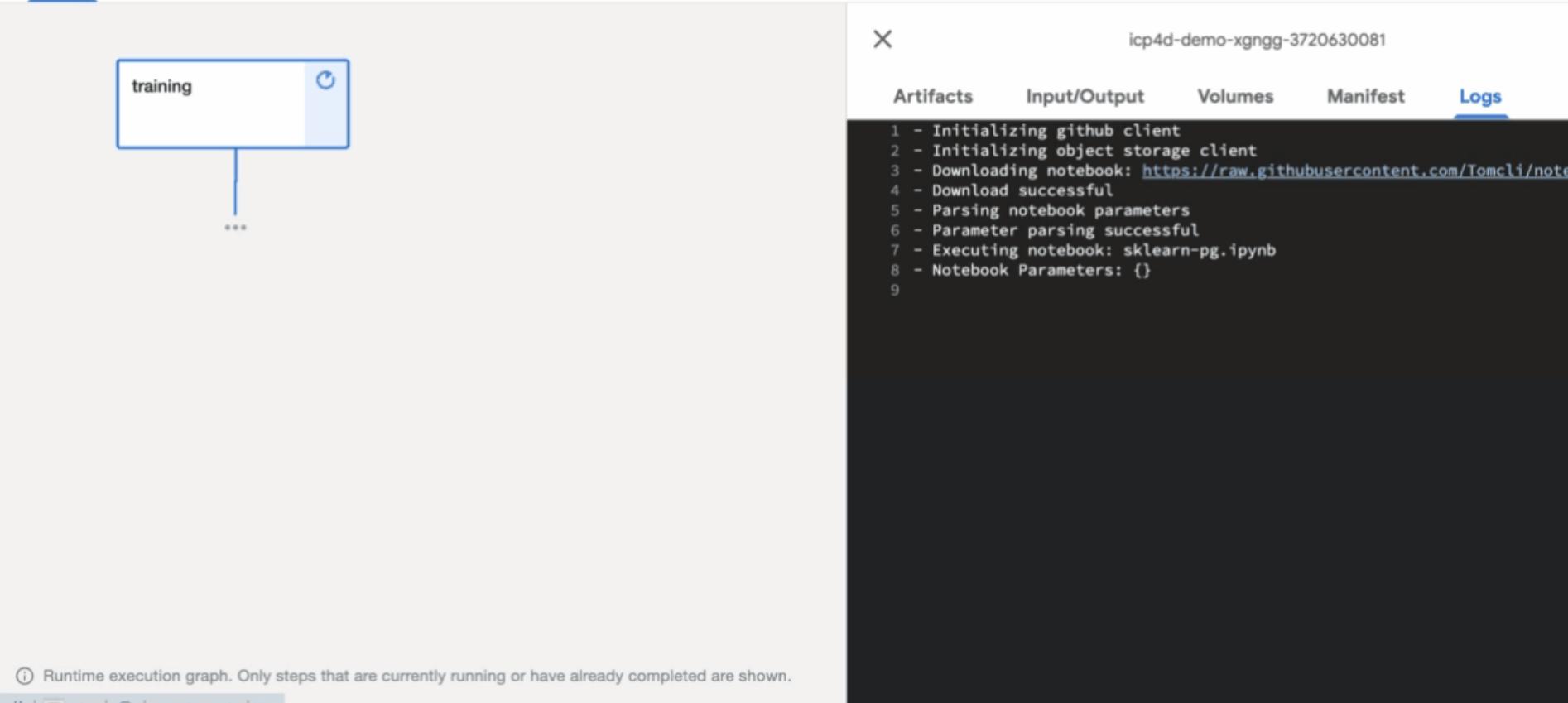
Kubeflow Pipelines can train, deploy and serve

Experiments > KFServing Experiments

← ⏪ animesh-refarch-reefer-ml (f6766)

Retry Clone run Terminate Archive

Graph Run output Config



The screenshot shows the Kubeflow Pipelines interface. On the left, there's a 'Graph' view displaying a single step named 'training' with a progress bar. Below it, there's a note: 'Runtime execution graph. Only steps that are currently running or have already completed are shown.' On the right, there's a detailed log pane for a specific run labeled 'icp4d-demo-xnggg-3720630081'. The log pane has tabs for Artifacts, Input/Output, Volumes, Manifest, and Logs, with 'Logs' being the active tab. The log output is as follows:

```
1 - Initializing github client
2 - Initializing object storage client
3 - Downloading notebook: https://raw.githubusercontent.com/Tomcli/notebook
4 - Download successful
5 - Parsing notebook parameters
6 - Parameter parsing successful
7 - Executing notebook: sklearn-pg.ipynb
8 - Notebook Parameters: {}
9
```



seldon

Spark

jupyter

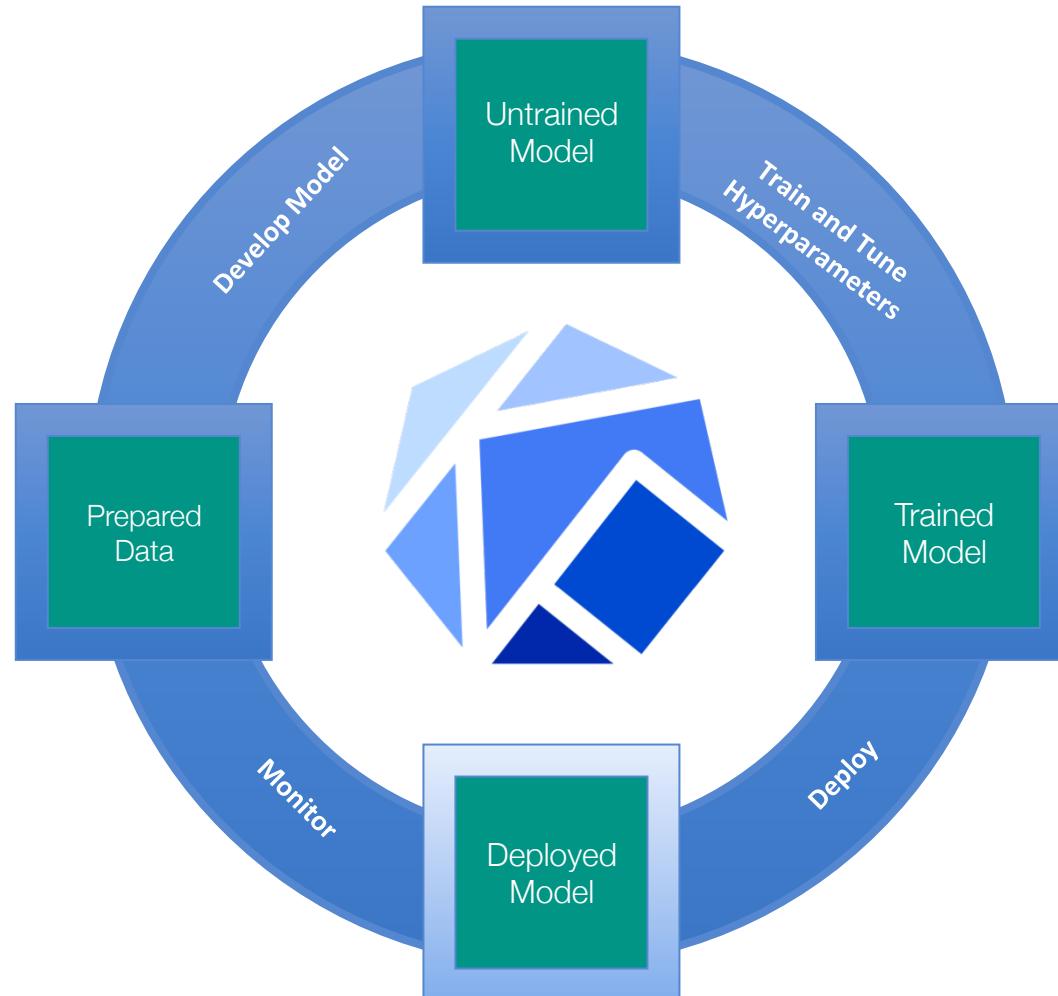
Kubernetes
Ready



Kubeflow

ML and AI Platform

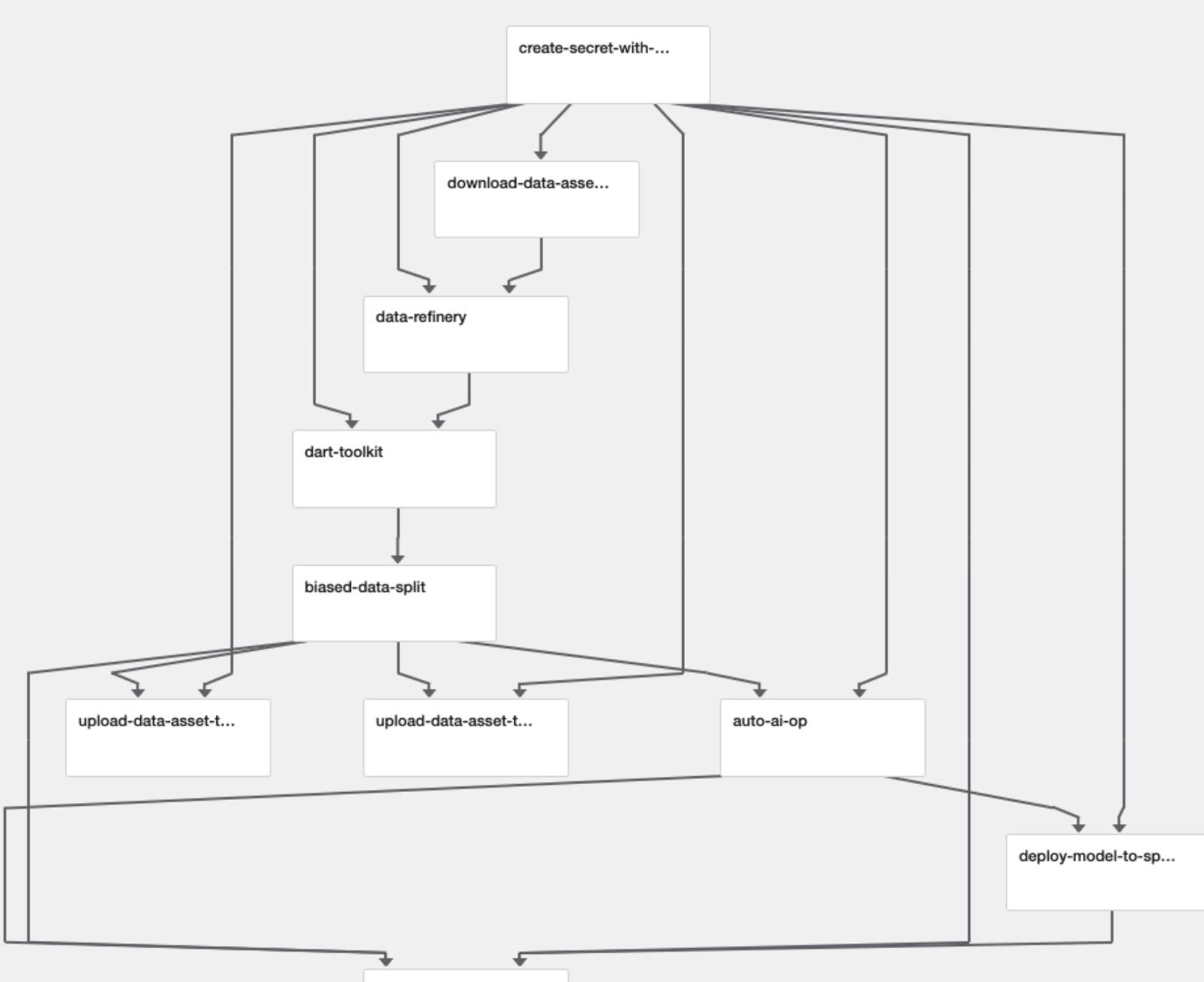
Watson Productization of Kubeflow Pipelines



Watson AI Pipelines

- Demonstrate that Watson can be used for end-end AI lifecycledata prep/model training/model risk validation/model deployment/monitoring/updating models
 - Demonstrate that the full lifecycle can be operated programmatically, and have **Tekton** as a backend instead of Argo

The screenshot displays the Kubeflow UI interface for managing machine learning experiments. On the left, a sidebar provides navigation links for Pipelines, Experiments, Artifacts, Executions, Archive, Documentation, GitHub Repo, and AI Hub Samples. The main area shows the 'Experiments' view for 'GCR-AutoAI-Experiment-1'. A top banner indicates the current run is 'Run of Train the model and monitor with OpenScale (a2ba6)'. Below this, tabs for Graph, Run output, and Config are available. The Graph tab shows a runtime execution graph with nodes like 'create-secret-wit...', 'download-data-a...', 'data-refinery', 'dart-toolkit', 'biased-data-split', 'upload-data-asses...', 'auto-ai-op', 'deploy-model-to...', 'mm-check-in-openscale', and 'mm-check-in-op...'. Most nodes are marked as completed with green checkmarks. A modal window titled 'train-the-model-and-monitor-with-openscale-pjgr-2081484978' is open, showing the Logs tab with a detailed log of the pipeline's execution steps. To the right, a large circular 'Progress map' visualization tracks the status of various pipelines and algorithms. The map is divided into segments labeled 'FEATURE TRANSFORMERS', 'PIPELINES', 'TOP ALGORITHMS', and 'german_credit_dat...'. Colored dots represent different stages or metrics. Below the map, a 'Pipeline leaderboard' table lists four entries: Pipeline 4 (Gradient Boosting Classifier, Accuracy 0.807, Enhancements HPO-1, FE, HPO-2), Pipeline 3 (Gradient Boosting Classifier, Accuracy 0.804, Enhancements HPO-1, FE), Pipeline 2 (Gradient Boosting Classifier, Accuracy 0.804, Enhancements HPO-1), and Pipeline 1 (Gradient Boosting Classifier, Accuracy 0.802, Enhancements None). The bottom of the page includes a note about the build commit (ca58b22) and a runtime message about completed steps.

**Run details**

Pipeline*

Train the model and monitor with OpenScale

Choose

Pipeline Version*

Train the model and monitor with OpenScale

Choose

Run name*

Run of Train the model and monitor with OpenScale (a28a6)

Description (optional)

This run will be associated with the following experiment

Experiment*

GCR-AutoAI-Experiment-1

Choose**Run Type** One-off Recurring**Run parameters**

Specify parameters required by the pipeline

github_token

6fd86cff0394892e772cd84d43a9e2d7546b1576

ai_config_url

https://raw.githubusercontent.com/IBM-Lifecycle-Poland/kubeflow-pipelines-credentials/master/config_cpd

catalog_name

DataCatalog

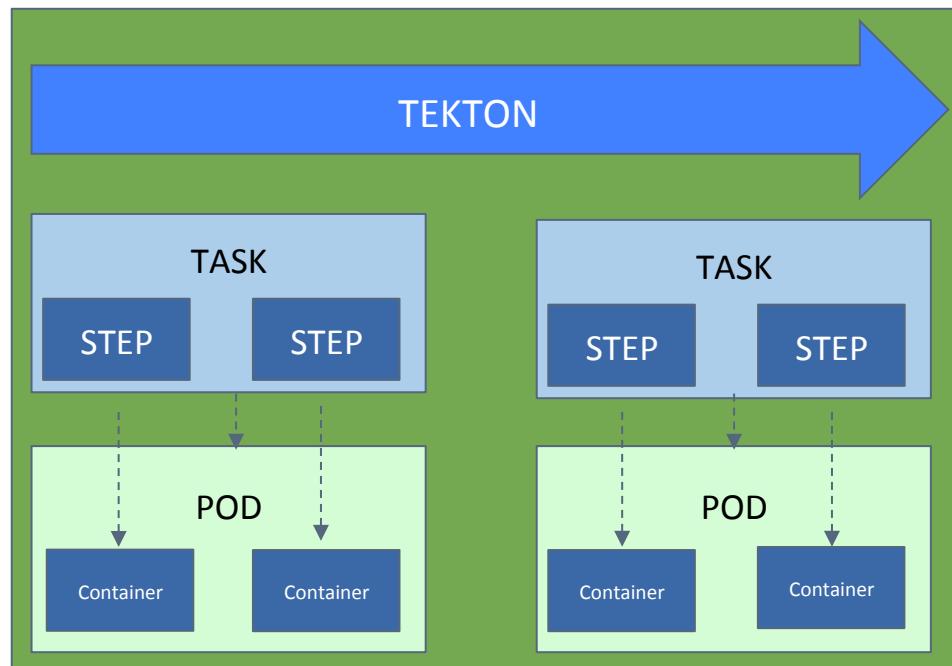
asset_id

2737bafc-3f78-4e2d-850a-e7f352b3d6b8

pre_production_space_uid

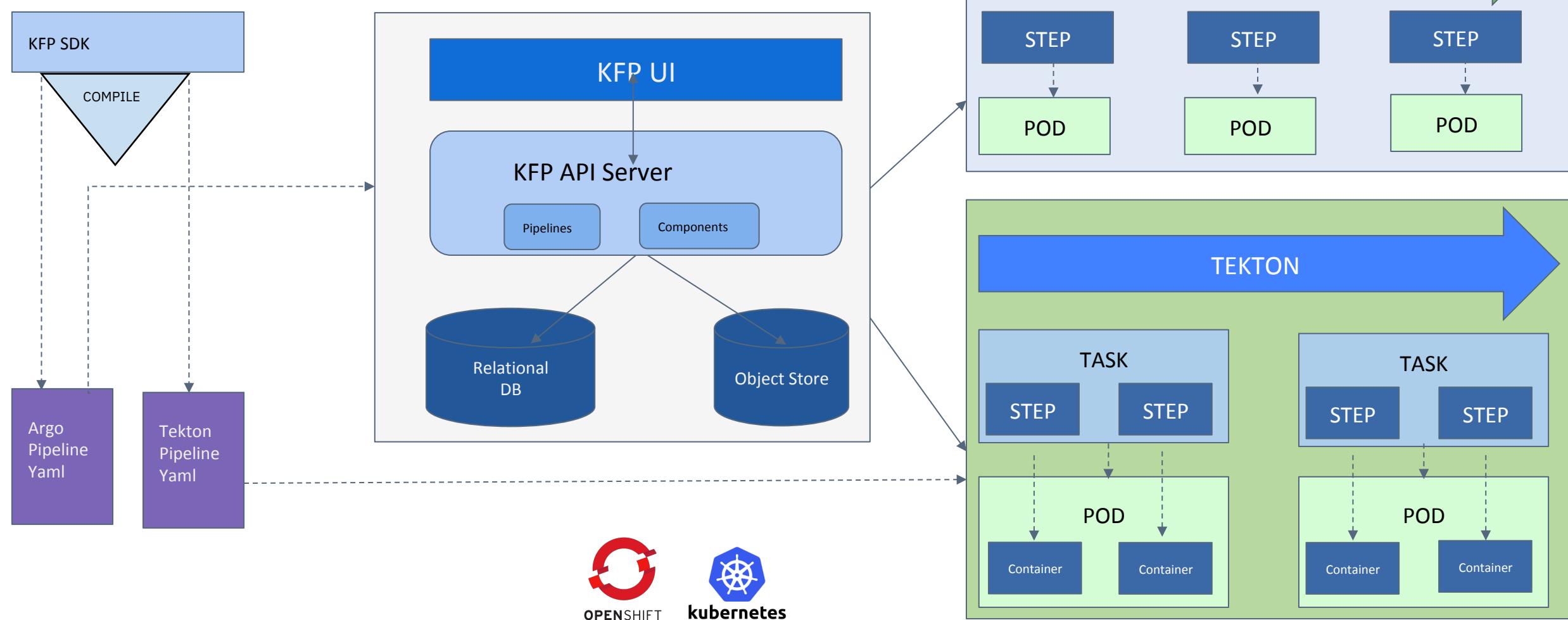
1dd2aaec-781a-4712-a7ff-ae1862cf7a84

- The Tekton Pipelines project provides Kubernetes-style resources for declaring CI/CD-style pipelines.
- Tekton introduces several new CRDs including Task, Pipeline, TaskRun, and PipelineRun.
- A PipelineRun represents a single running instance of a Pipeline and is responsible for creating a Pod for each of its Tasks and as many containers within each Pod as it has Steps.



- A **PipelineResource** defines an object that is an input (such as a git repository) or an output (such as a docker image) of the pipeline.
- A **PipelineRun** defines an execution of a pipeline. It references the Pipeline to run and the PipelineResources to use as inputs and outputs.
- A **Pipeline** defines the set of Tasks that compose a pipeline.
- A **Task** defines a set of build Steps such as compiling code, running tests, and building and deploying images.

KFP – Tekton Phase One



Pluggable Components

Spark

Watson Studio

WML

Open Scale

Kubeflow Training

Seldon

AIF360

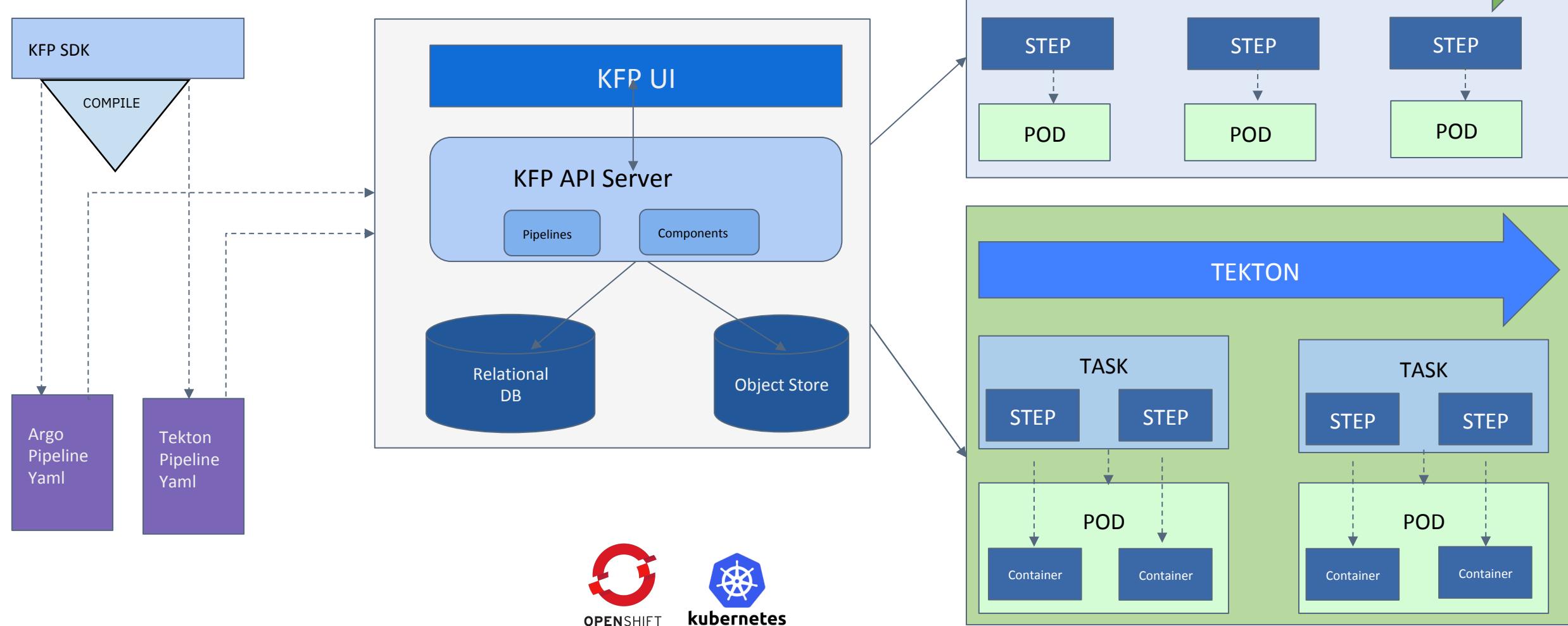
ART

KATIB

KFSERVING



KFP – Tekton Phase Two



Pluggable Components

Spark

Watson Studio

WML

Open Scale

Kubeflow Training

Seldon

AIF360

ART

KATIB

KFSERVING



Multiple Moving parts, with different stakeholders

Tekton Community: Argo with version 2.6 much more mature than Tekton v0.11 (alpha) when the work started around 5 months ago

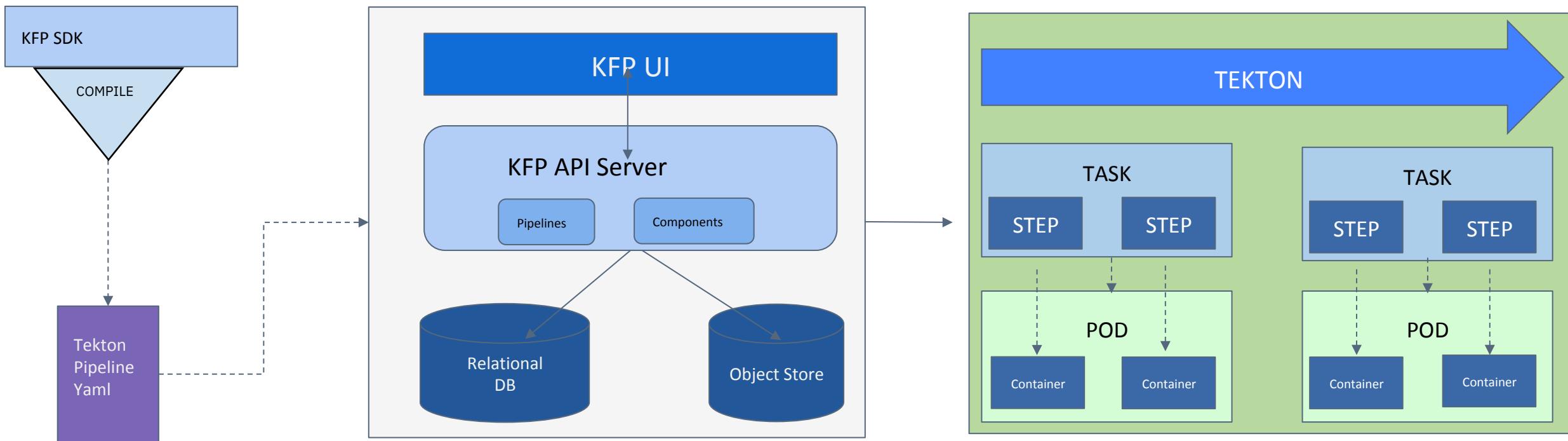
- Multiple features and capabilities lacking in Tekton when we kick started
 - The team had to default to a spreadsheet to start tracking and mapping KFP DSL features, and areas where Tekton needed to bring features and functions.
- Overall 50 DSL capabilities identified and corresponding Tekton features started getting mapped.
- Multiple features like Kubernetes resources support to create/patch/update/delete them, image pull secrets, loops, conditionals, support for system params didn't exist. Or existed partially
 - Tekton started moving from alpha to beta as the work progressed, and few features left behind in alpha mode
 - Multiple issues opened on Tekton. Required ramping up the team of Tekton contributors to help drive these issues . Formed a virtual team of IBM Open tech developers (Andrea Frittoli, Priti Desai), IBM Systems team (Vincent Pli) DevOps team (Simon Kaegi), RedHat (Vincent Demeester etc.) to drive Tekton requirements

Kubeflow Pipeline and TFX Community: Open source team needed to be formed for the specific mission. And trained. Additionally Google needed to be brought up on the same page, and convinced the validity of integration.

- Multiple design reviews established with Google, and jointly agreed on a direction after they were convinced why we were doing it, and why it makes sense.
- Convincing to accelerate the IR (Intermediate Representation) strategy with TFX, so as to be able to drive this the right way
- Huge dependency in Kubeflow Pipeline code on Argo, including the API backend and UI all written with Argo dependency
- Internal IBM team divided to attack different areas: Compiler (Christian Kadner), API (Tommy Li), UI (Andrew), Feng Li (IBM Systems, China)
- Inability of Kubeflow Pipeline backend to take multiple CRDs, which is the default model Tekton follows. So everything needed to be bundled in one Pipeline Spec
- Type check, workflow utils, and parameter replacement are heavily tied with Argo API. In addition, the persistent agent is watching the resources using the Argo API type.
- MLOps Sig in CD Foundation leveraged to bring Kubeflow Pipelines and Tekton team together



KFP – Tekton: Delivered



OPENSHIFT



kubernetes



Pluggable Components

Spark

Watson Studio

WML

Open Scale

Kubeflow Training

Seldon

AIF360

ART

KATIB

KFSERVING

...



Same KFP Experience: DAG, backed by Tekton YAML

Pipelines

← default-watson-ml (default-watson-ml)

+ Create run + Upload version + Create experiment Delete

[Graph](#) [YAML](#)

```
graph TD; A[train-model-watson-ml] --> B[store-model-watson-ml]; B --> C[deploy-model-watson-ml]
```

x train-model-watson-machine-learning

Input parameters

- compute_name
- compute_nodes
- execution_command
- framework
- framework_version
- run_definition
- run_name
- runtime
- runtime_version
- train_code

Output parameters

- run-uid /tmp/outputs/run_uid/data
- training-uid /tmp/outputs/training_uid/data

Arguments

Show summary ⓘ Static pipeline graph

0s | 1194 x 660 | x

Same KFP Exp: Logs, Lineage Tracking and Artifact Tracking

Experiments > tekton-experiments

← ✓ Run of watson-ml-pipeline-with-artifacts (d6bd5)

Retry Clone run Terminate Archive

Graph Run output Config

create-secret-k... ✓

train-model-wat... ✓

store-model-wat... ✓

deploy-model-wa... ✓

X kfp-on-wml-training-run-1dd60-train-model-watson-machine--xt4gc

Input/Output Visualizations ML Metadata Volumes Logs Pod Events

```
9
10
11
12
13
14 -----
15 Log monitor done.
16 -----
17
18
19
20
21 #####
22
23 Metric monitor started for training run: af80b10e-12f3-4053-a71c-31ff4ea8df56
24
25 #####
26
27
28
29
30 -----
31 Metric monitor done.
32 -----
33
34
35 status: {'state': 'pending'}
36 {'completed_at': '2020-07-06T21:15:15.208Z', 'message': {'text': 'Training job af80b10e-12f3-4053-a71c-31ff4ea8df56 started.'}}
37 training_details {'metadata': {'created_at': '2020-07-06T21:11:38.049Z', 'guid': 'af80b10e-12f3-4053-a71c-31ff4ea8df56'}}
38
```

① Runtime execution graph. Only steps that are currently running are shown.

0s | 1214 x 669 | X

End to end Kubeflow Components : With KFP-Tekton

Recurring run configs Experiment description 

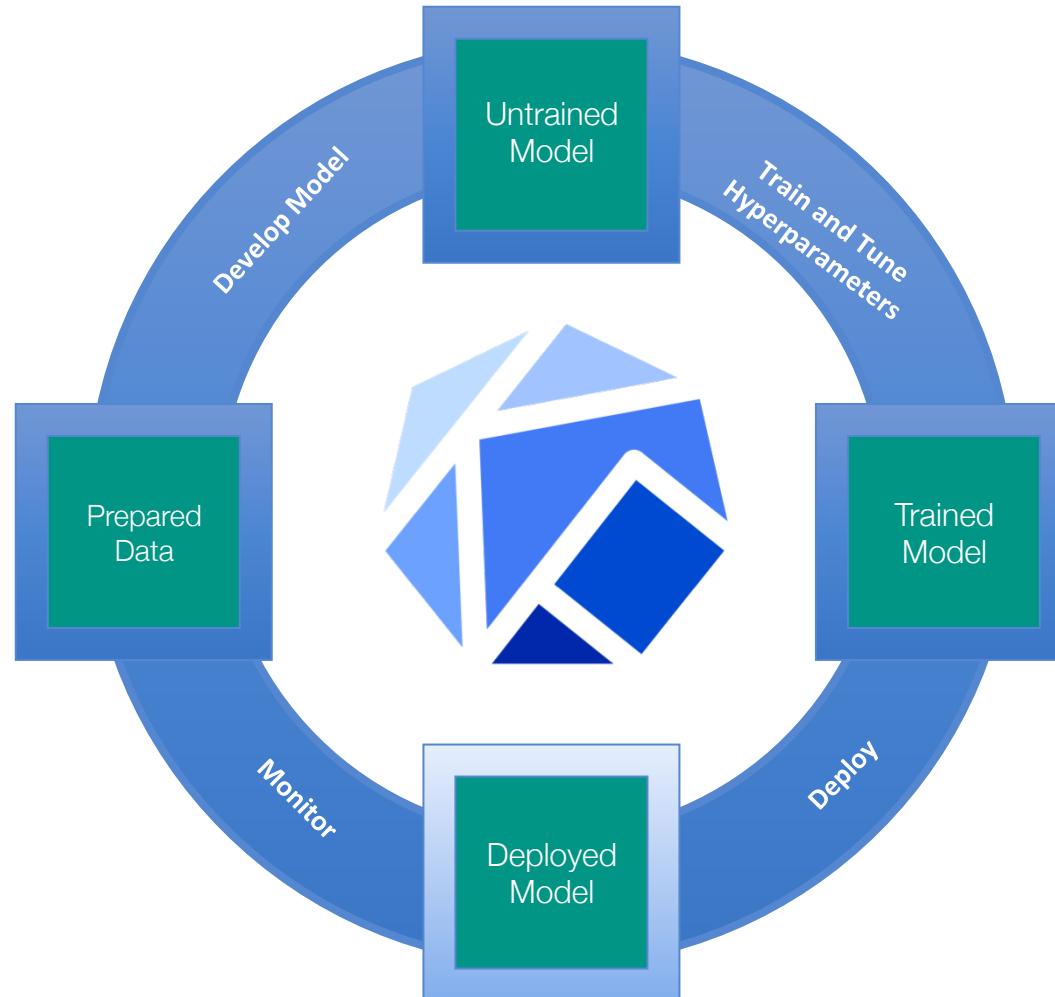
0 active Manage

Runs [+ Create run](#) [+ Create recurring run](#) Compare runs Clone run Archive

Filter runs 

<input type="checkbox"/> Run name	Status	Duration	Pipeline Version	Recurring Run...	Start time ↓
Run of mnist-e2e-pipeline (7d2c8)		-	mnist-e2e-pipeline	-	7/7/2020, 12:28:38 AM
Run of mnist-model-cleanup (91455)		-	mnist-model-cleanup	-	7/6/2020, 5:27:54 PM
mnist-e2e-pipeline-animesh (bf69b)		-	mnist-e2e-pipeline	-	7/6/2020, 4:48:15 PM
Run of watson-ml-pipeline-with-artifacts (d...)		-	watson-ml-pipeline-with-arti...	-	7/6/2020, 2:11:07 PM
Run of watson-ml-pipeline-with-artifacts (d...)		-	watson-ml-pipeline-with-arti...	-	6/22/2020, 6:21:28 PM
Watson-ml-pipeline-with-artifacts		-	watson-ml-pipeline-with-arti...	-	6/14/2020, 7:15:30 PM
Run of watson-ml-pipeline (f5876)	 	-	-	-	6/11/2020, 4:23:45 PM
0s 1195 x 632 (b2541)		-	-	-	6/2/2020, 5:19:25 PM

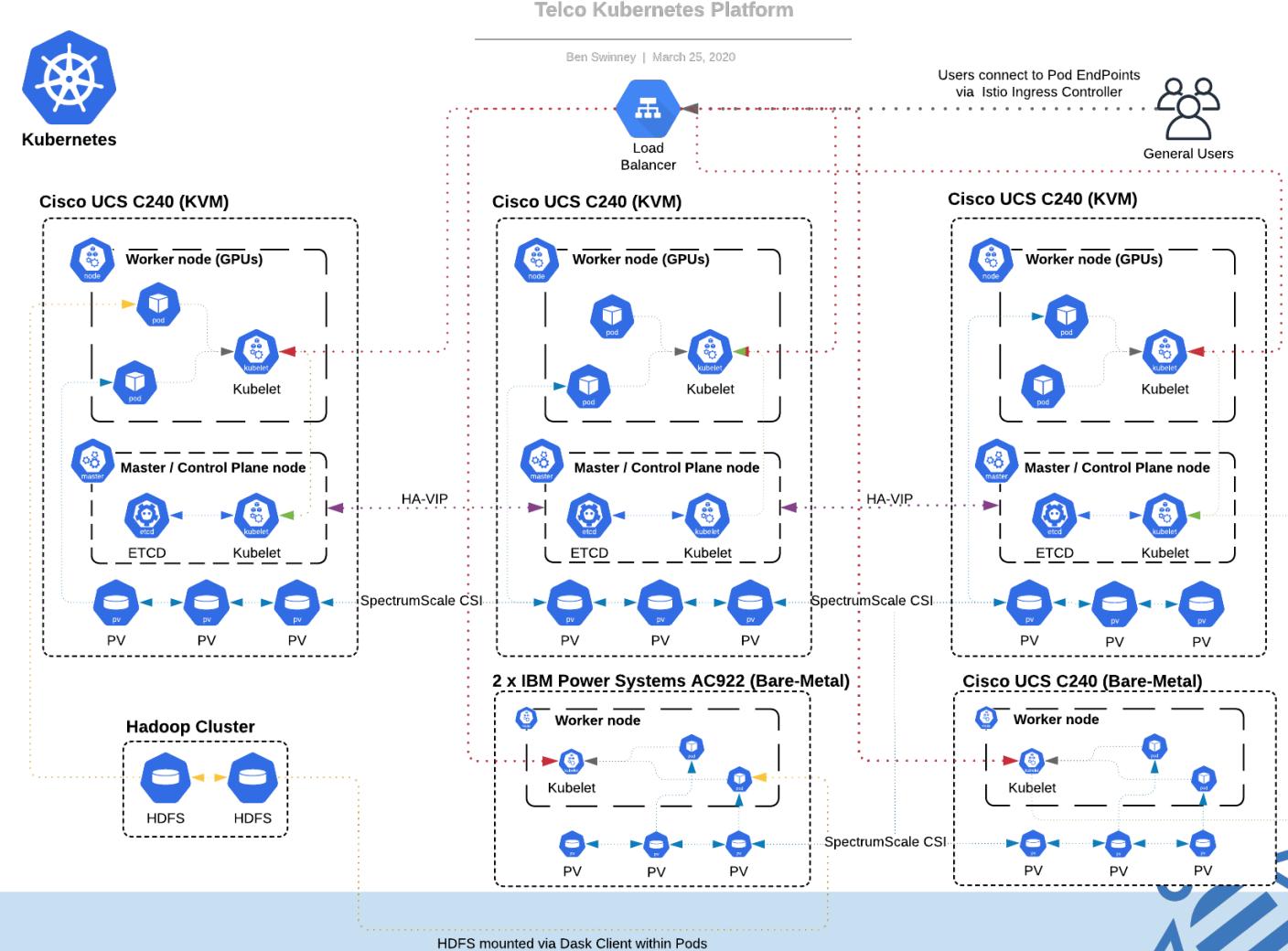
Kubeflow Adoption: External and Internal



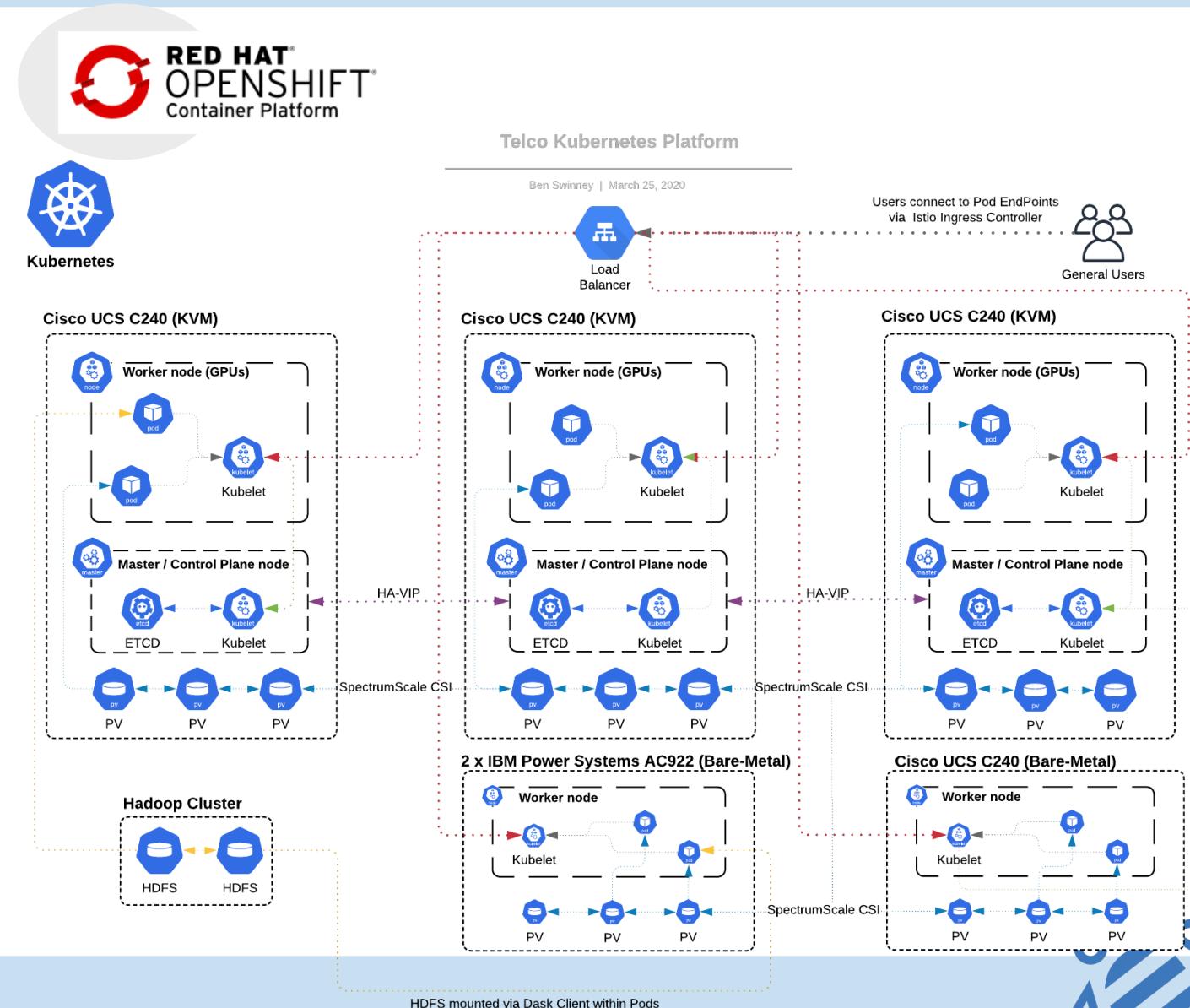
THINK 2020 Session: End-to-End Data Science and Machine Learning for Telcos: Telstra's Use Case
<https://www.ibm.com/events/think/watch/replay/126561688>

Telstra AI Lab - (TAIL) - Configuration

- Kubernetes – 1.15
- Spectrum Scale CSI Driver
- MetallLB for Load Balancing
- Istio 1.3.1 for ingress
- Kubeflow – 1.0.1
- Jupyter Notebook images are IBM's multiarchitecture powerai images (<https://hub.docker.com/r/ibmcom/powerai/tags>)



- RedHat OpenShift – 4.3
- GPU Operator
- Kubeflow Operator
- Extending the compute
- Integrate feature stores and streaming technologies
- Integrate with CI/CD tools (Tekton Pipelines)



IBM Yara – Working with IBM to build a Data Science Platform for Digital Farming

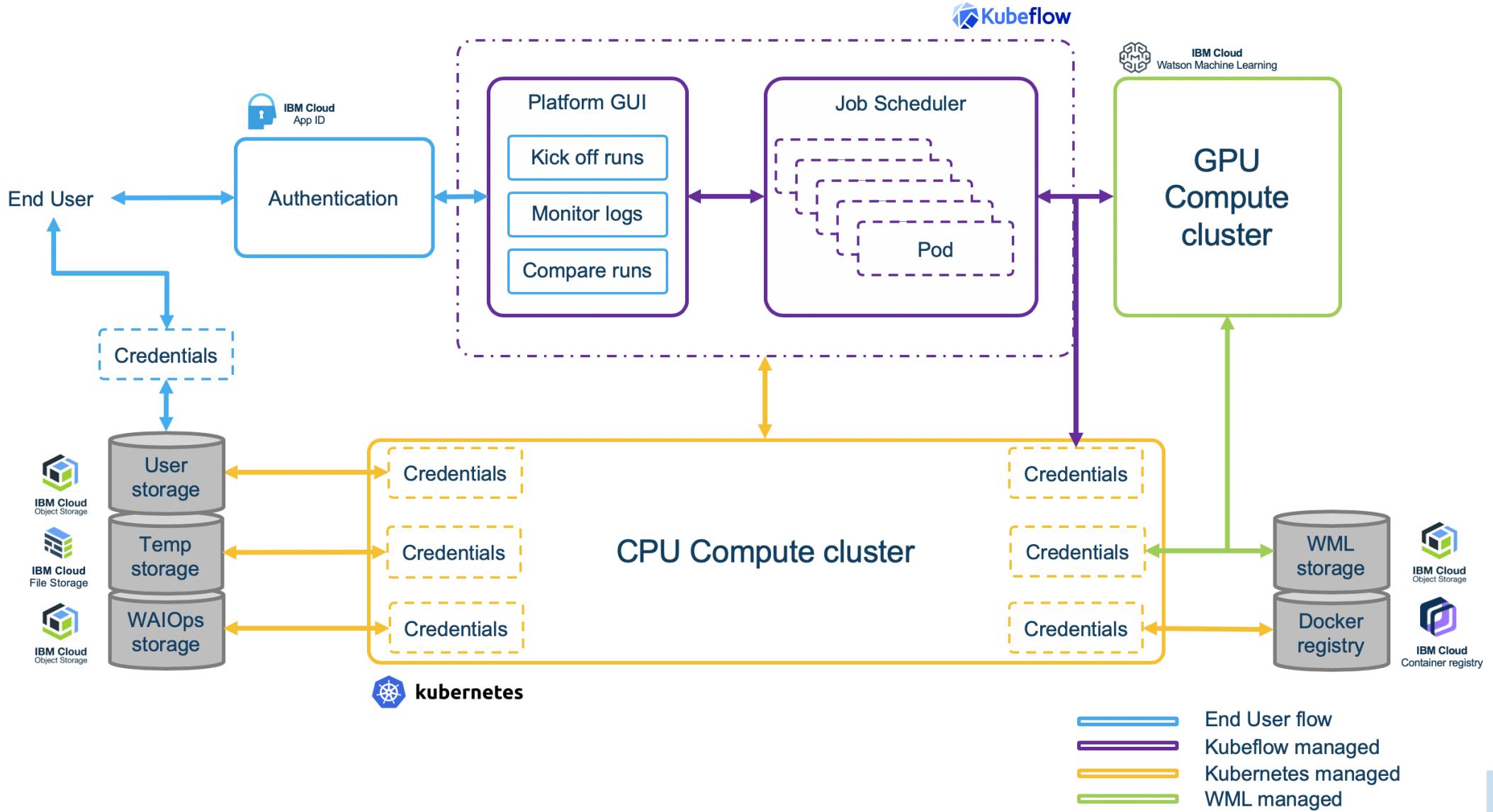


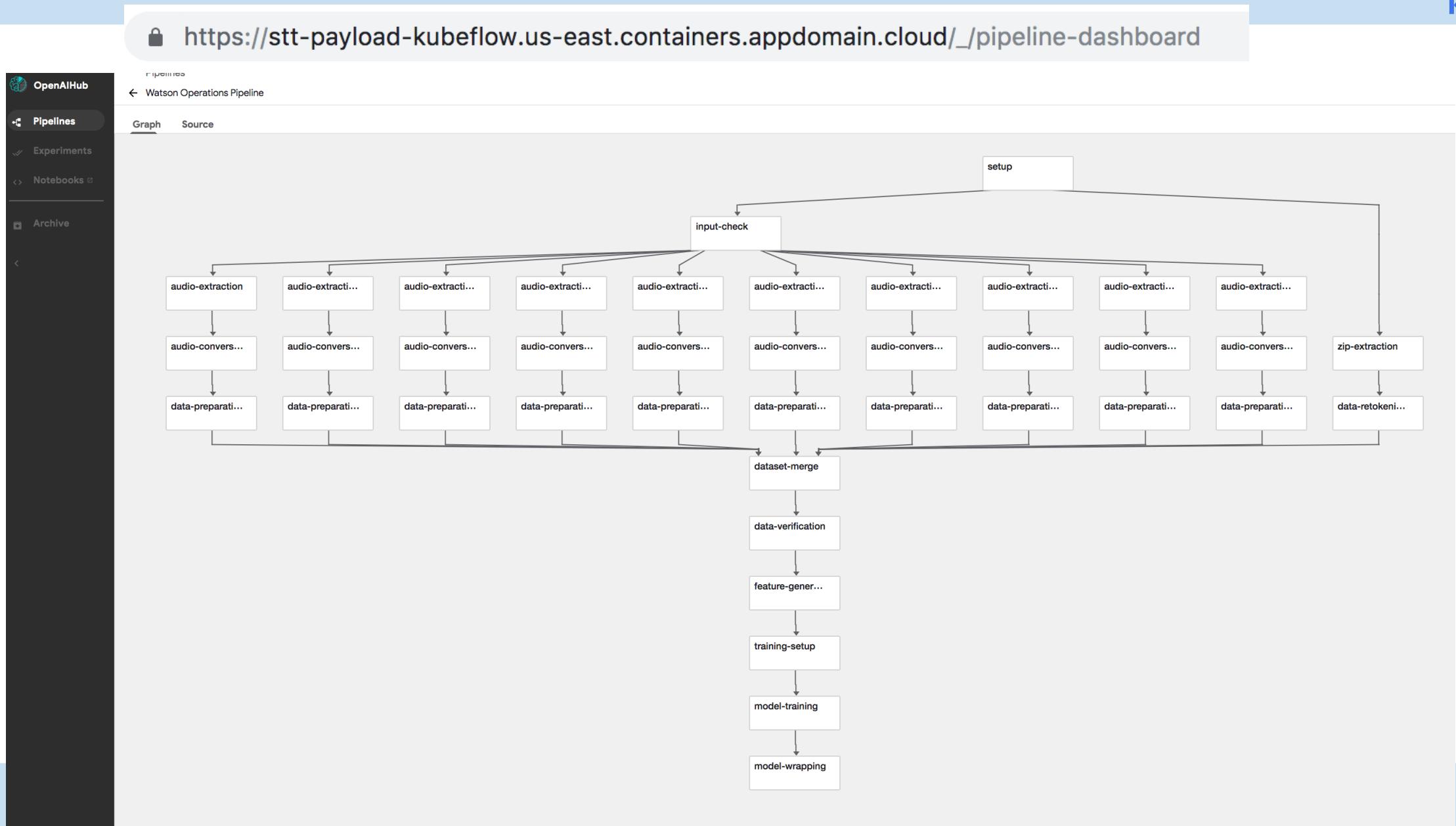
THINK 2020 Session: Enable Smart Farming using Kubeflow

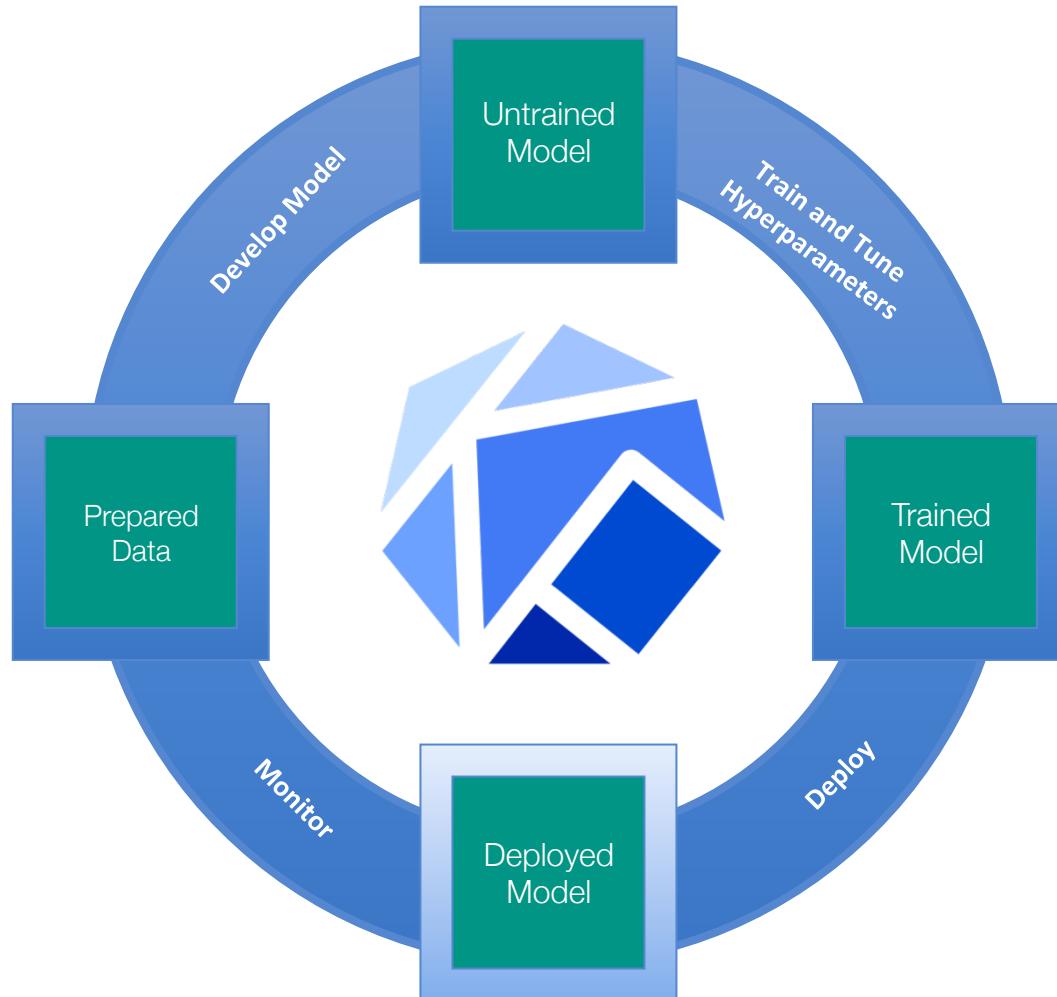
<https://www.ibm.com/events/think/watch/replay/126494864>



Watson STT: Kubeflow Pipelines running Operations







'Upstream' is about extracting oil and natural gas from the ground; 'midstream' is about safely moving them thousands of miles; and 'downstream' is converting these resources into the fuels and finished products we all depend on.

Upstream



Upstream has many phases, beginning with the exploratory process. Geologists search on dry land or in oceans for signs of gas or oil.

Midstream



When a well is producing, oil or gas enters the midstream juncture. The middle part of the process requires multiple cooperation.

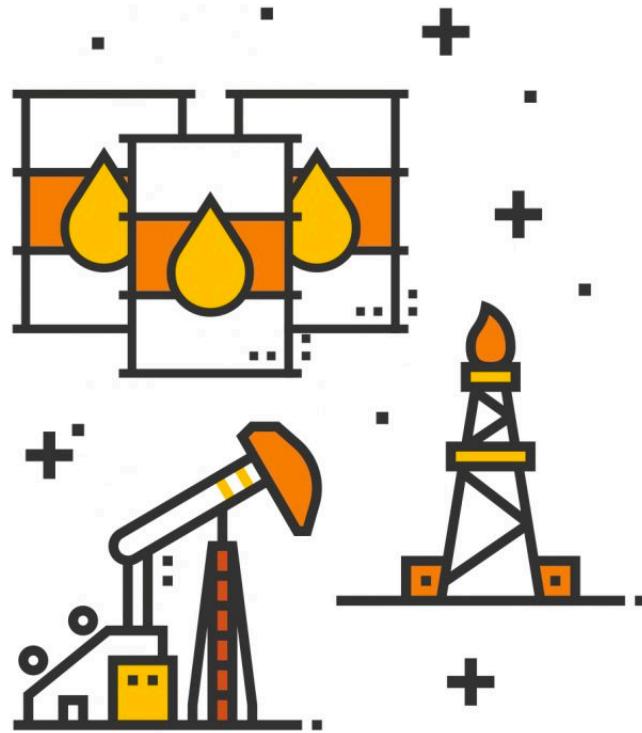
Downstream



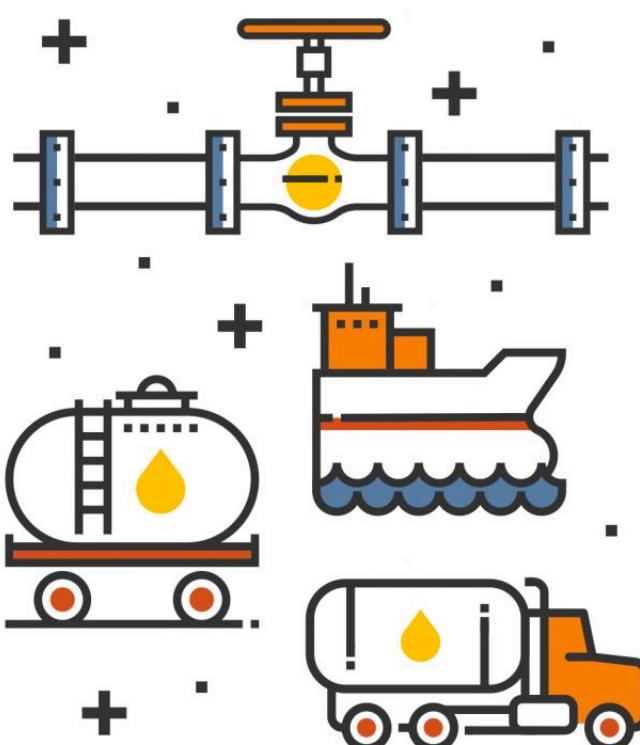
The downstream stage handles processing, selling, marketing and distributing gas or oil. Final products depend upon the initial resource.

'Upstream' is about extracting oil and natural gas from the ground; 'midstream' is about safely moving them thousands of miles; and 'downstream' is converting these resources into the fuels and finished products we all depend on.

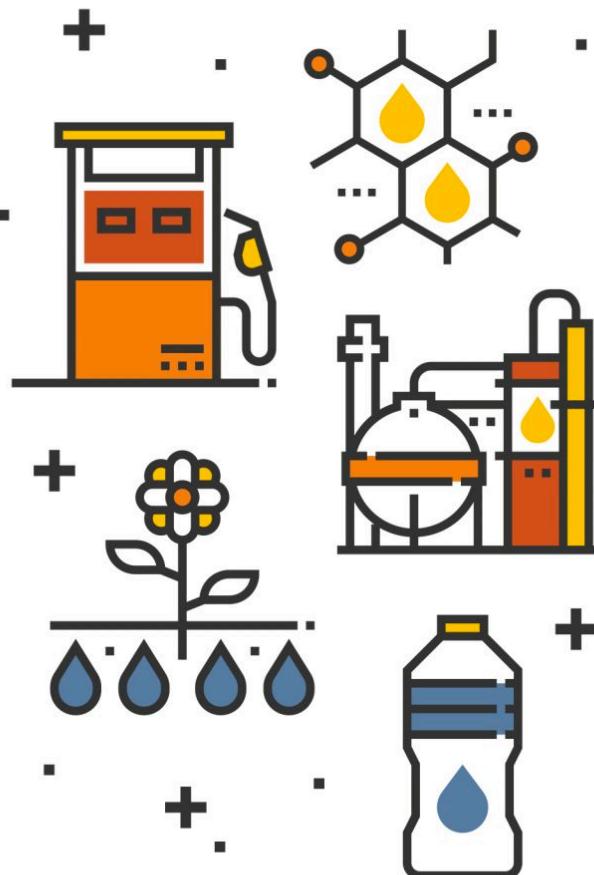
UPSTREAM

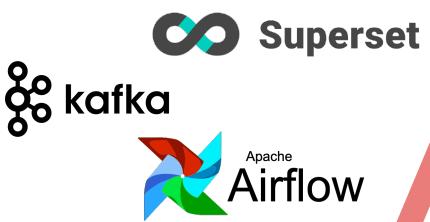


MIDSTREAM



DOWNSTREAM



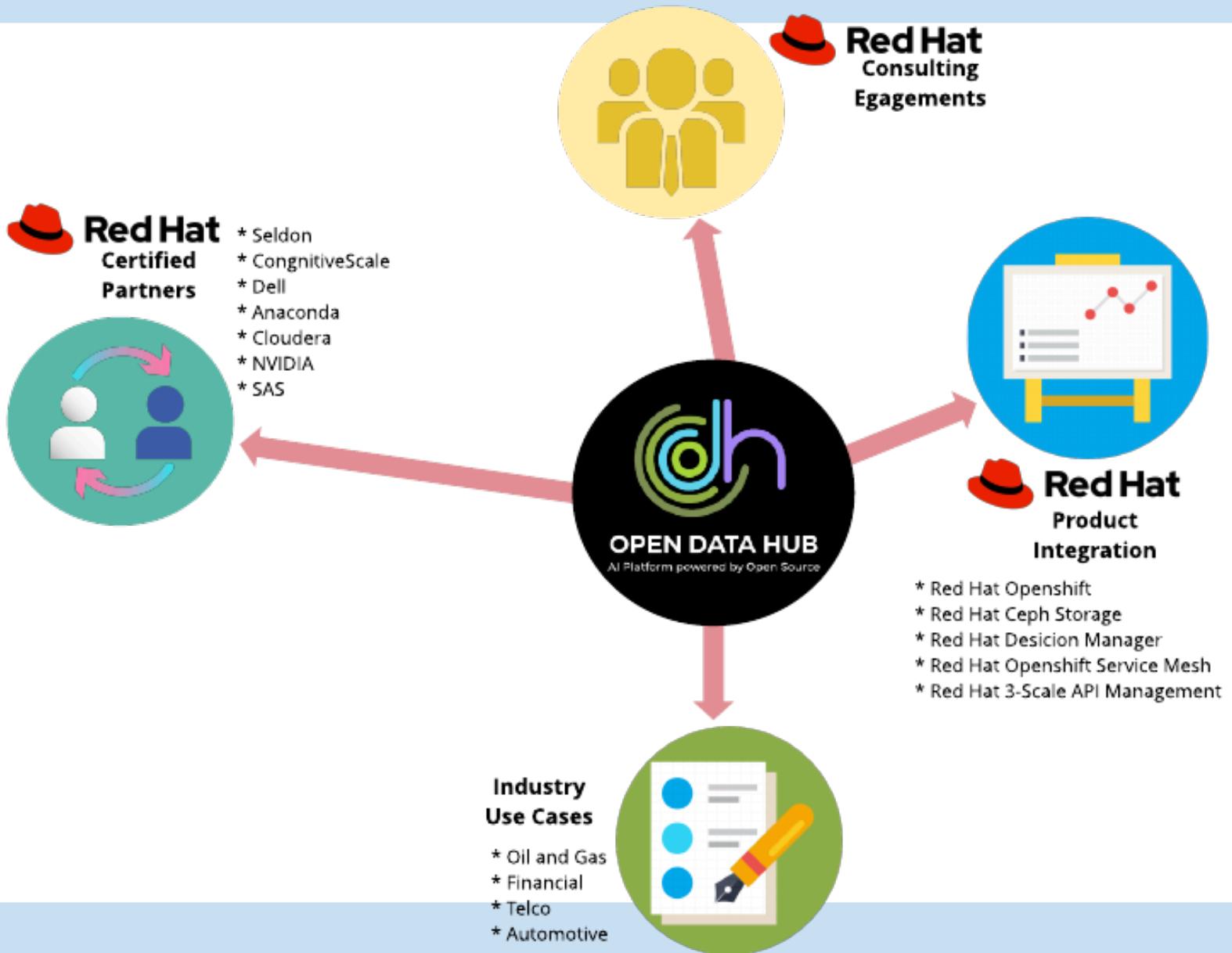


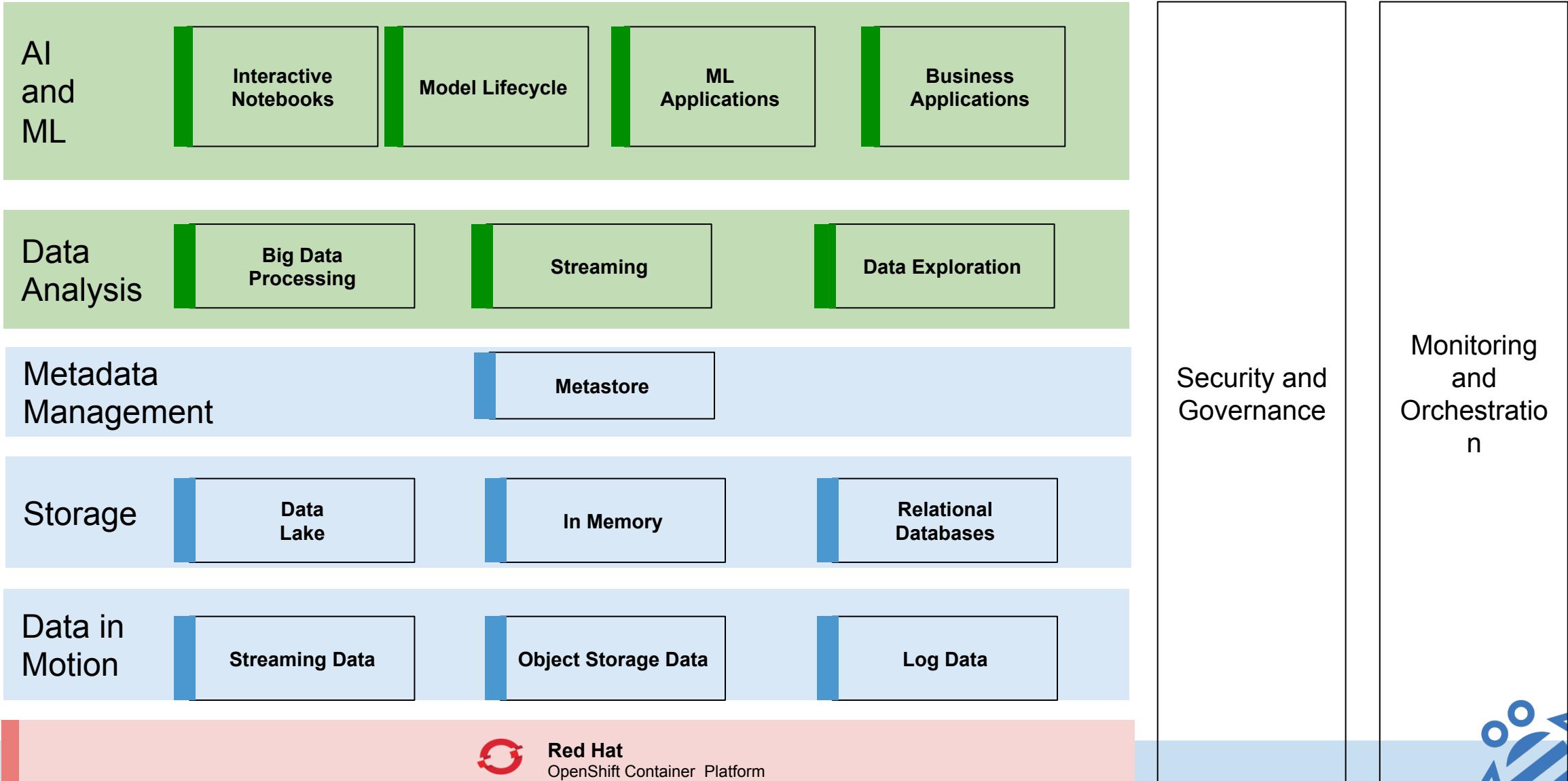
jupyterhub

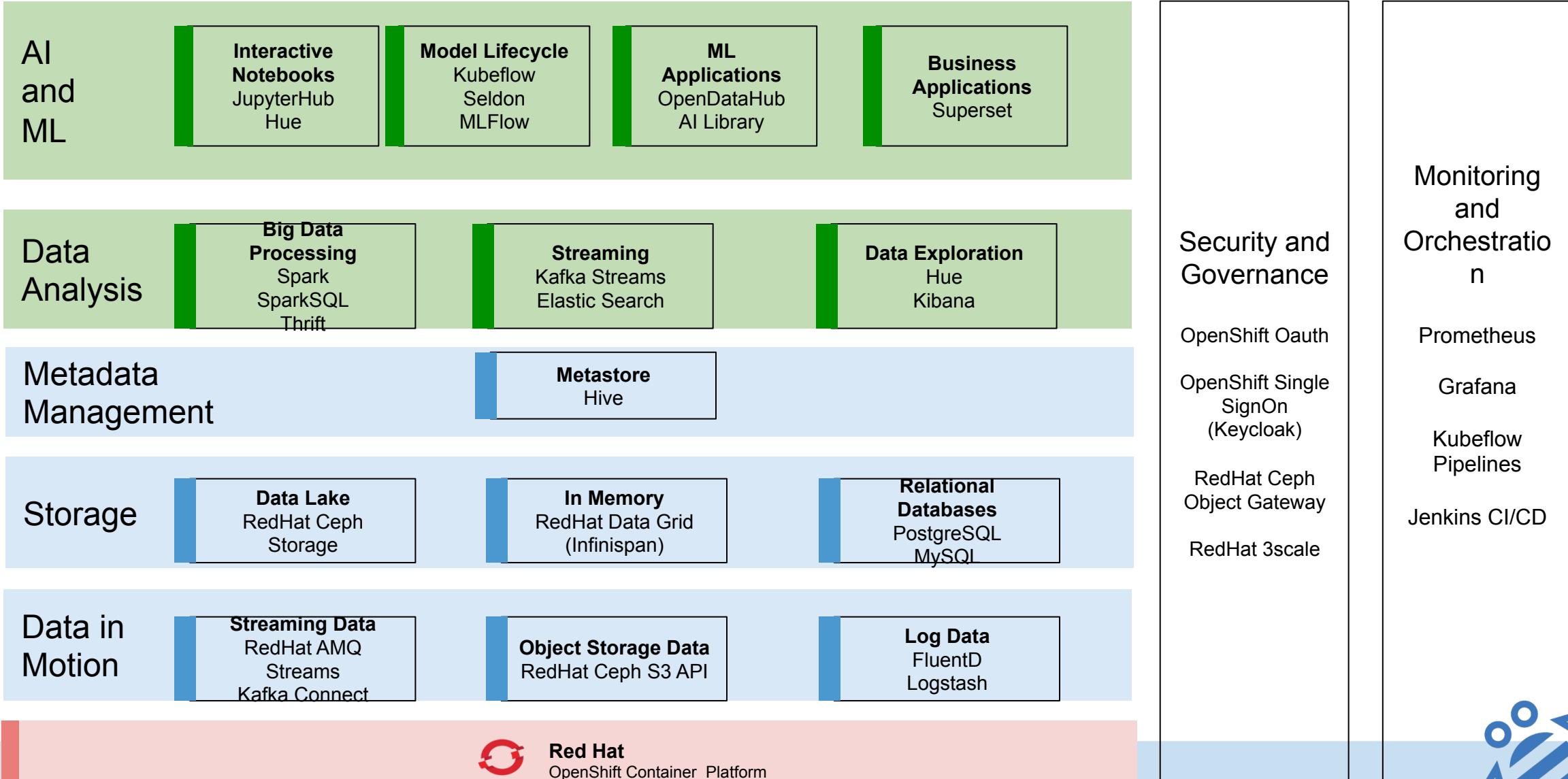


Data Platform

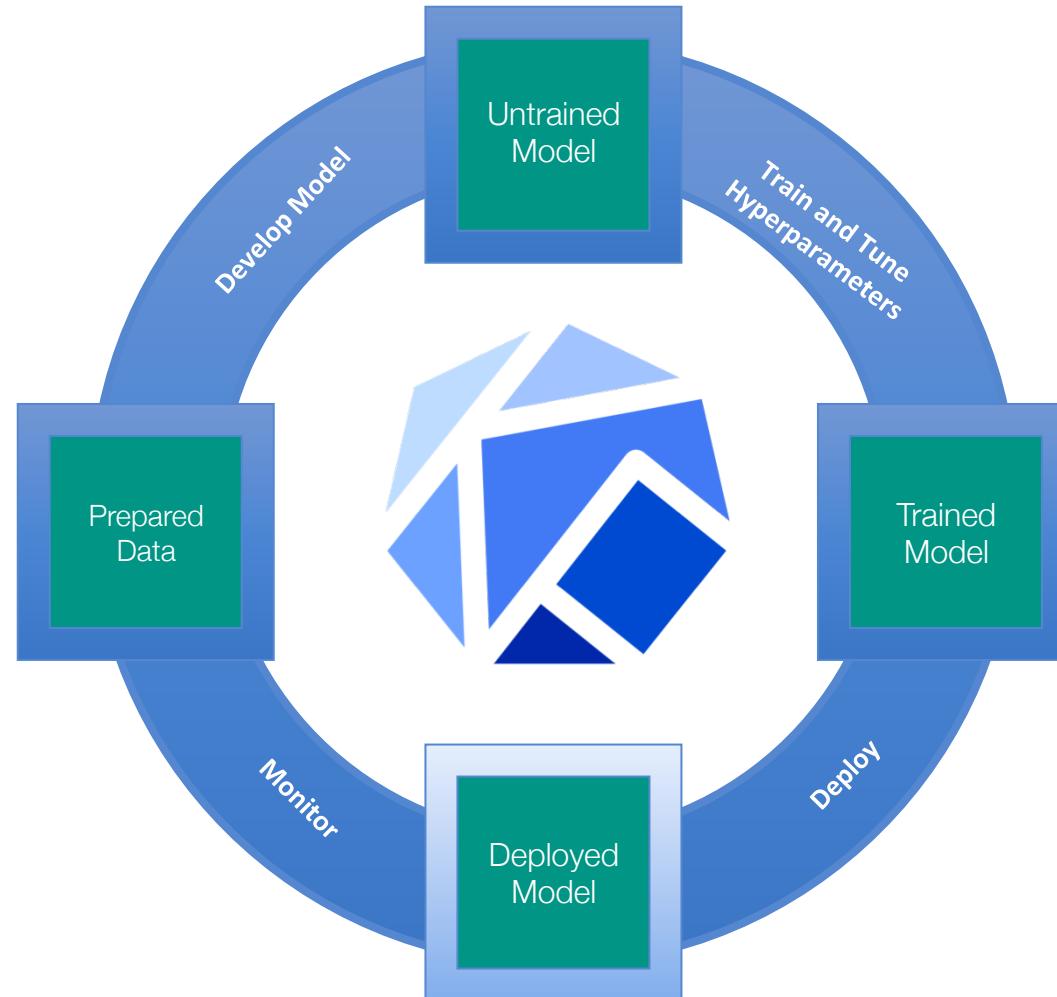
Operator Hub - operatorhub.io







OpenDataHub and Kubeflow: Relationship



Initial Goals:

- Kubeflow has a great traction, Make it available for OpenShift users
 - Done in <https://github.com/opendatahub-io/manifests>
- Offer ODH users components installed by KF
- And offer components from ODH (Kafka, Apache SuperSet, Hive...) to KF community
- Decide if we can leverage KF project and community as upstream for ODH
- Think Kubernetes -> OpenShift
- Frees up ODH maintainers time to make sure KF keeps running well on OpenShift

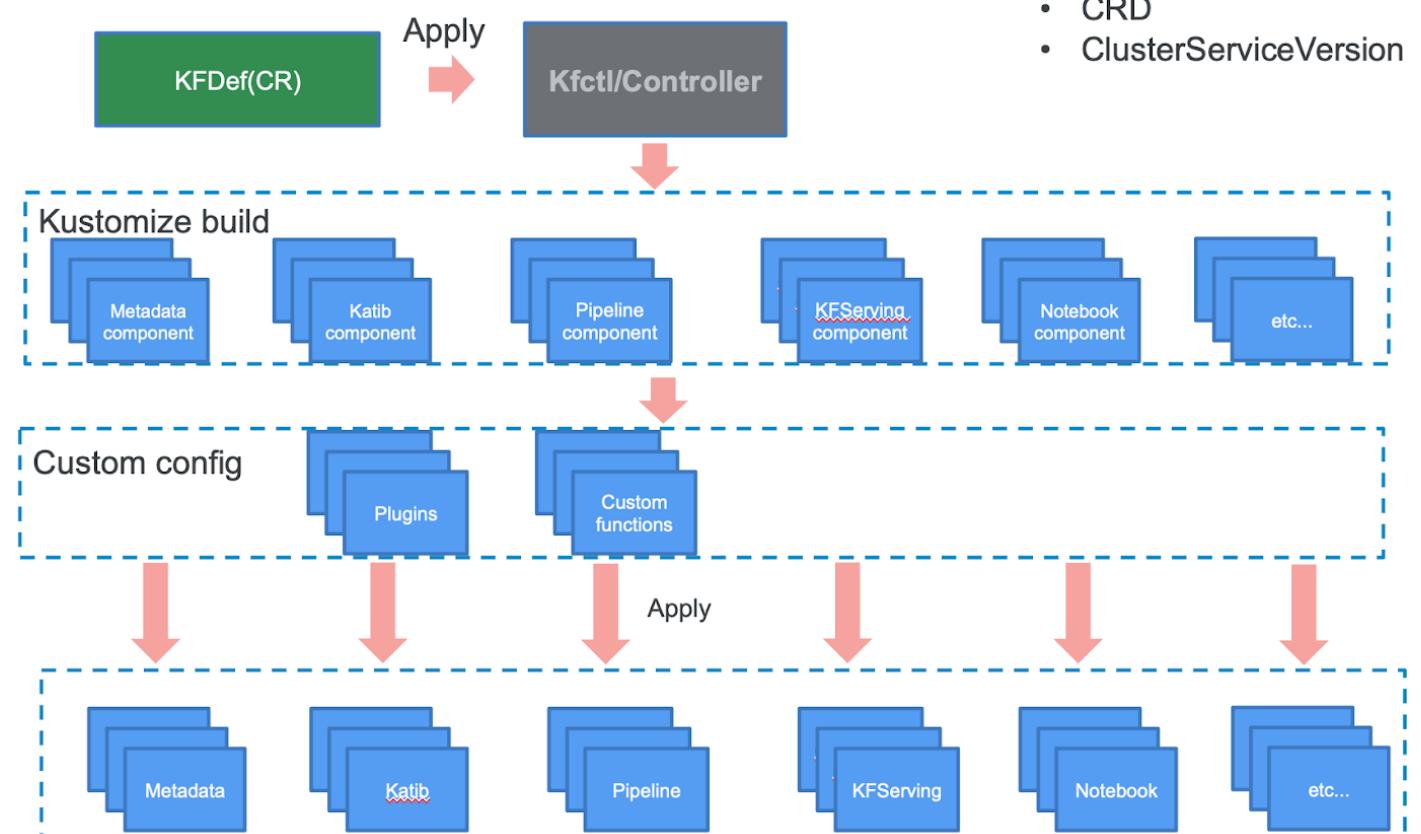


IBM Kubeflow Operator – Contributed by IBM to Kubeflow community to help enable OpenDataHub



- <https://operatorhub.io/operator/kubeflow>
- Deploy, manage and monitor Kubeflow
- On various environments
 - ❑ IBM Cloud
 - ❑ GCP
 - ❑ AWS
 - ❑ Azure
 - ❑ OpenShift
 - ❑ Other K8S

KFCTL CONTROLLER - Initial deployment



Kubeflow
provided by IBM

Kubeflow Operator for
deployment and management
of Kubeflow



Controller deployment files

- CRD
- ClusterServiceVersion

- A version of the Operator based on Kubeflow Architecture released:
https://developers.redhat.com/blog/2020/05/07/open-data-hub-0-6-brings-component-updates-and-kubeflow-architecture/?sc_cid=7013a000002DTqEAAW
- Most of the components converted:
<https://github.com/opendatahub-io/odh-manifests>
- Still a separate deployment – needs to do both ODH and Kubeflow in one go.

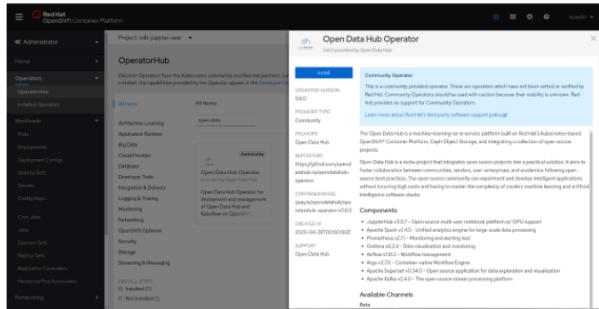
Future

- KF 1.0 on OpenShift
- Disconnected deployment
- Open Data Hub CI/CD
- Kubeflow on OpenShift CI
- UBI based ODH & KF
- Multitenancy model
- Mixing KF & ODH



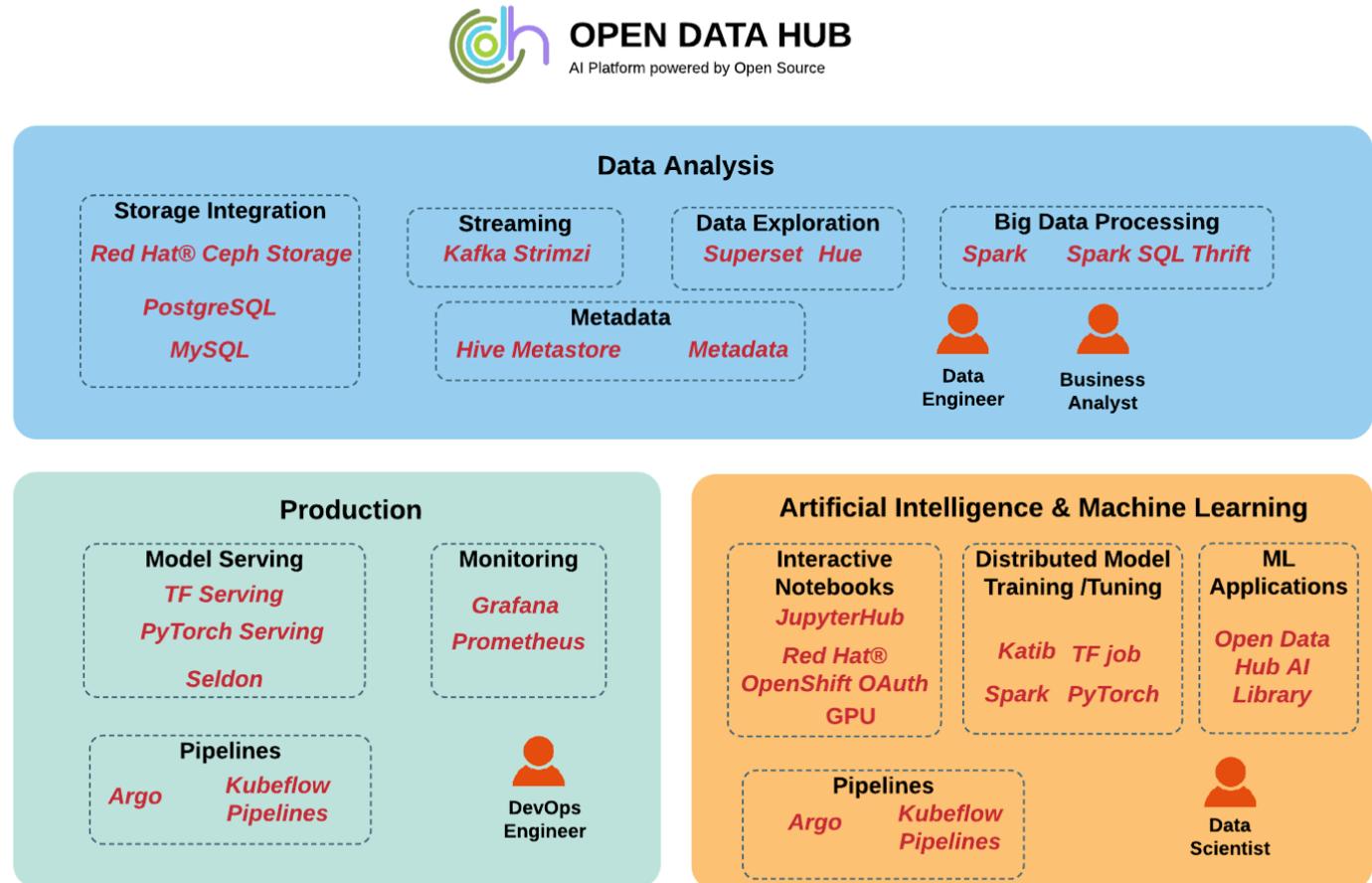
Open Data Hub 0.6 brings component updates and Kubeflow architecture

 By Václav Pavlín May 7, 2020



Open Data Hub (ODH) is a blueprint for building an AI-as-a-service platform on Red Hat's **Kubernetes**-based **OpenShift 4.x**. Version 0.6 of Open Data Hub comes with significant changes to the overall architecture as well as component updates and additions. In this article, we explore these changes.





Open Data Hub in OpenShift

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar has a dark theme with white text. The 'Projects' option is currently selected, highlighted in blue. The main content area displays a table of projects:

NAME ↑	STATUS
PR airflow-on-k8s-operator-system	Active
PR anonymous	Active
PR default	Active
PR kube-public	Active
PR kube-system	Active
PR opendatahub	Active

The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes the IBM logo, the Red Hat logo, and the text "Red Hat OpenShift Container Platform". A dropdown menu on the top right is open, showing options like Home, Projects, Status, Search, Events, Catalog, Workloads, Networking, Storage, Builds, Monitoring, Compute, Administration, and other resources. The "Projects" option is also listed here. The main content area shows a list of projects under the heading "Project: opendatahub".

- P jupyterhub-nb-kube-3aadmin
- spark-operator
- D spark-operator, #1
- strimzi
- D odh-message-bus-entity-operator, #1
- SS odh-message-bus-kafka
- SS odh-message-bus-zookeeper
- superset
- DC superset, #1
- other resources
- D ailibrary-operator, #1
- D airflow-on-k8s-operator-controller-manager, #1
- D argo-server, #1



Apache Superset

Superset Security Manage Sources Charts Dashboards SQL Lab

Growth Analysis Scratchpad

Database: main Schema: superset Add a table (43)

slices

- created_on
- changed_on
- id**
- slice_name
- datasource_type
- datasource_name
- viz_type
- params
- created_by_fk**
- changed_by_fk**
- description
- cache_timeout
- perm
- datasource_id

dashboards

- created_on
- changed_on
- id**
- dashboard_title
- position_json
- created_by_fk**
- changed_by_fk**
- CSS

Database: main Schema: superset Add a table (43)

SELECT b.dashboard_id, a.dashboard_title, b.slice_id, c
JOIN dashboards a
JOIN dashboard_slices b ON a.id = b.dashboard_id
JOIN slices c ON c.id = b.slice_id

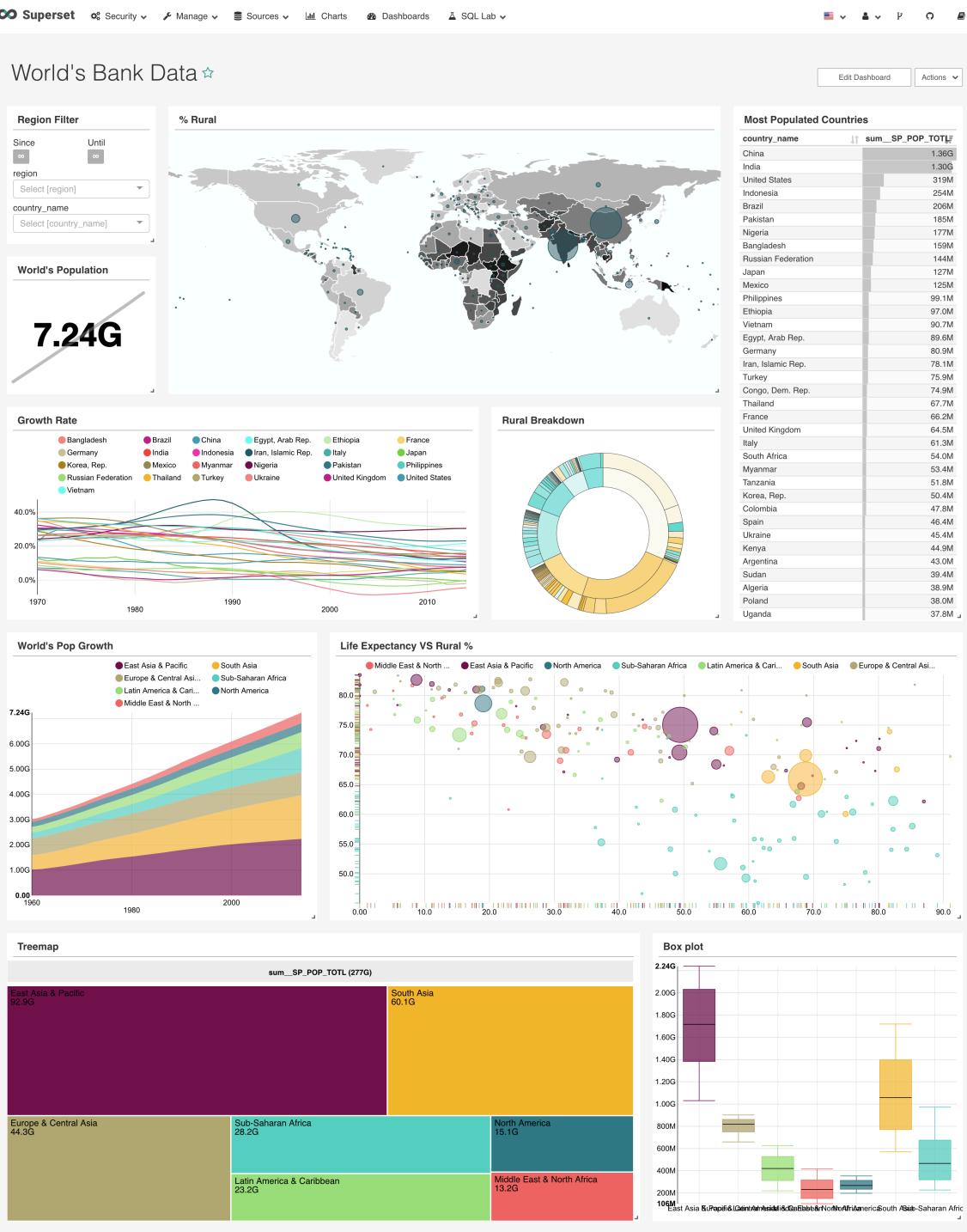
dashboards
dashboard_title
DATABASE
datasource_type
datasource_name
datasource_id

Run Query Save Query Share Query

Results Query History Preview for slices Preview for d

Visualize .CSV

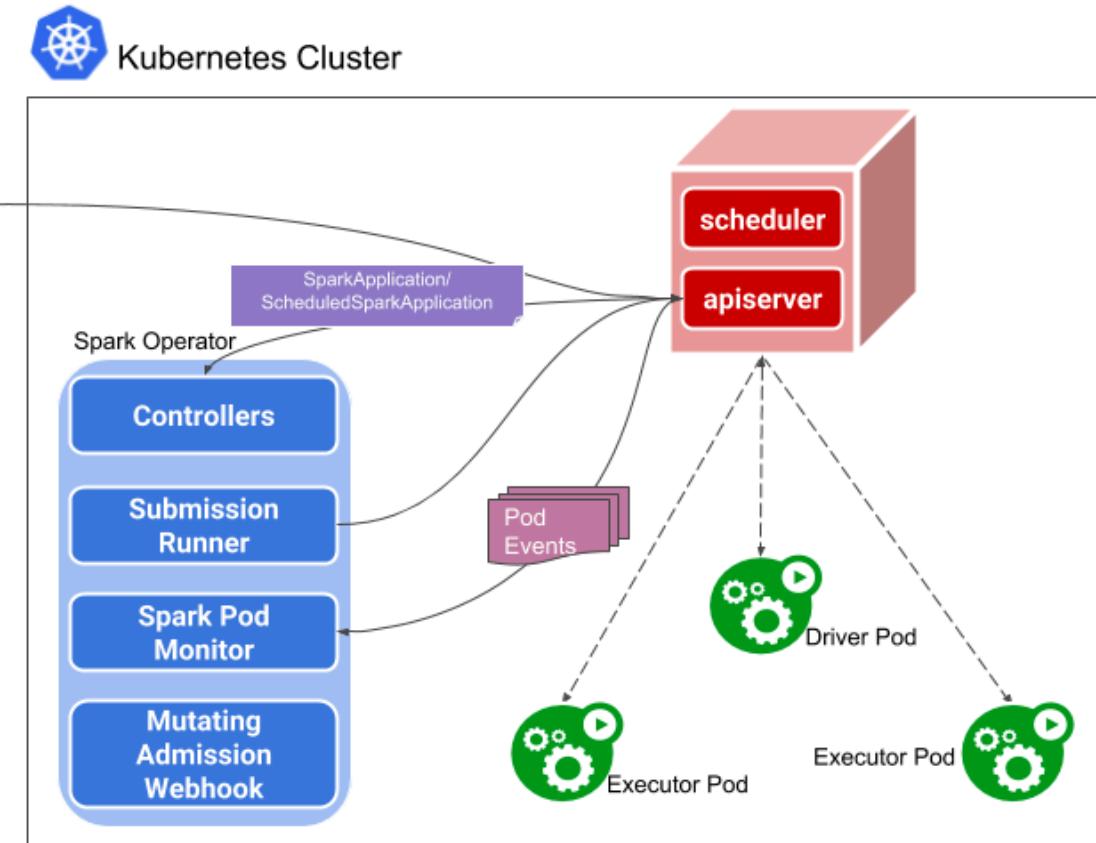
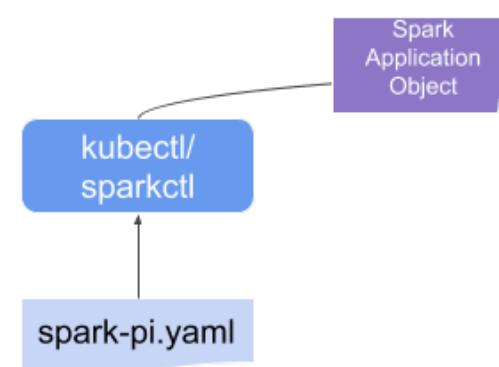
dashboard_id	dashboard_title	slice_id	slice_name
2	Births	882	Girls
2	Births	883	Boys
2	Births	884	Participants
2	Births	885	Genders
2	Births	886	Genders by State
2	Births	887	Trends
2	Births	888	Average and Sum Treemap
2	Births	889	Title
2	Births	890	Name Cloud



IBM Spark with Open Data Hub

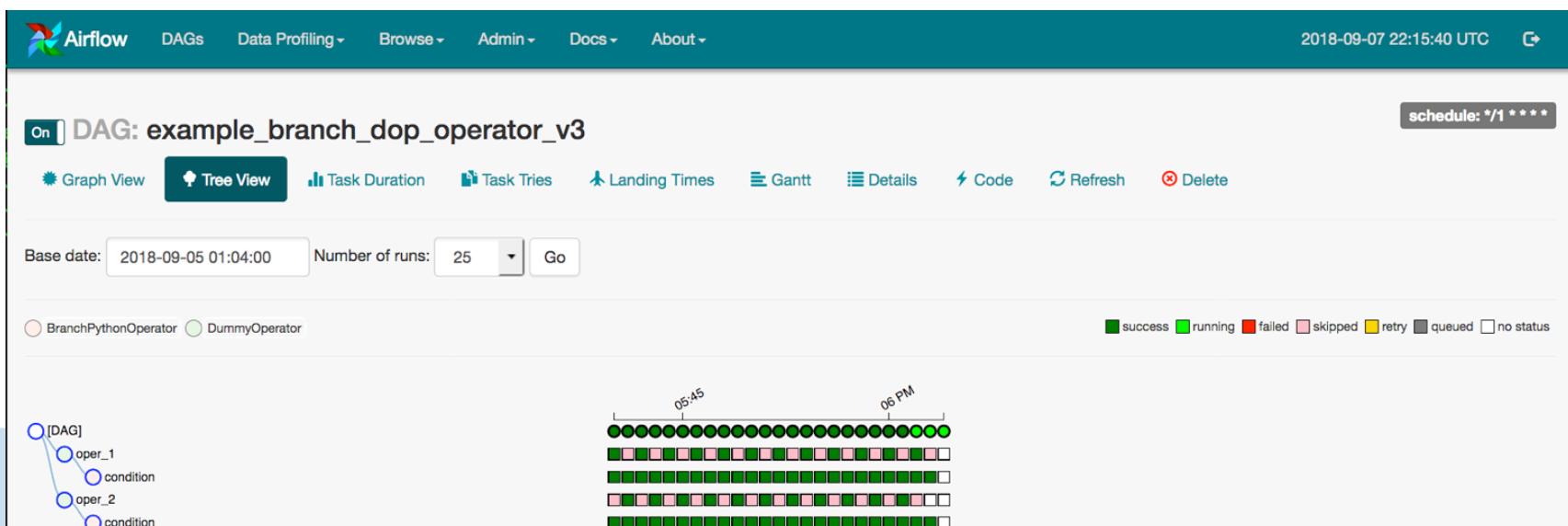


- Open Data Hub will also deploy the Spark Operator to manage Spark as an application.
- Two versions of Spark – Spark in dedicated mode and Spark on K8s
- Currently moving towards Spark on K8s Operator from Google for serverless Spark. IBM Hummingbird team investigating this

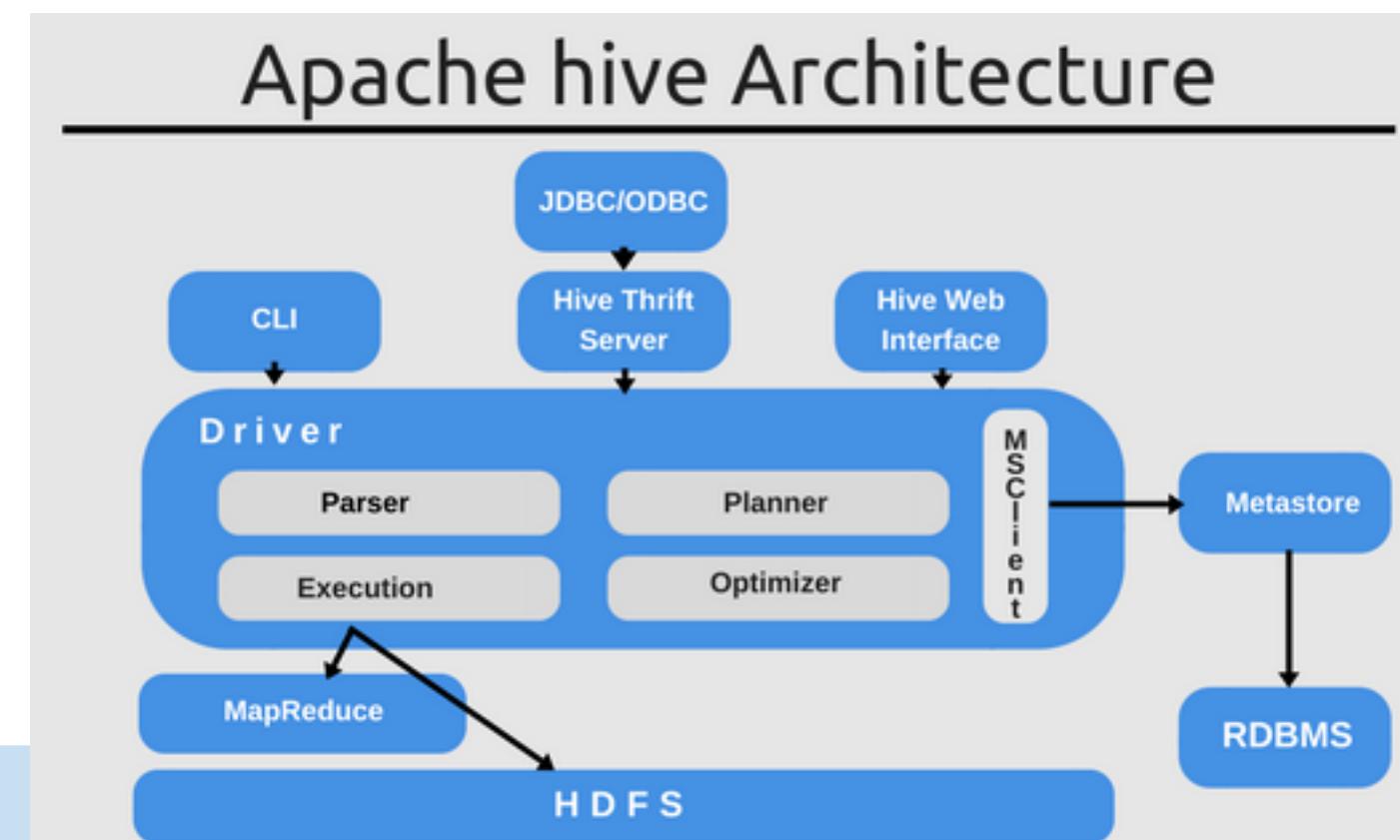


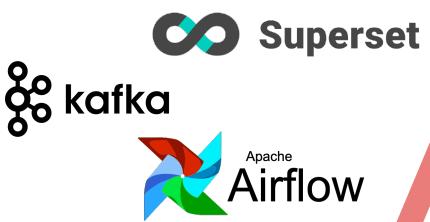
Google Cloud Platform

- Open Data Hub will also deploy the Airflow Operator to manage Airflow as an application.
- Using the Airflow Operator originally developed in the GoogleCloudPlatform repository and later donated to Apache.
- The Operator creates a controller-manager pod which will be created as a part of the Open Data Hub deployment.
- Users can then install the Airflow components they need from the available options (eg: CeleryExecutor or KubernetesExecutor, Postgres deployment or MySQL deployment etc.)



- Hive was one of the first abstraction engines to be built on top of MapReduce.
- Started at Facebook to enable data analysts to analyse data in Hadoop by using familiar SQL syntax without having to learn how to write MapReduce.
- Hive an essential tool in the Hadoop ecosystem that provides an SQL dialect for querying data stored in HDFS, other file systems that integrate with Hadoop such as MapR-FS and Amazon's S3 and databases like HBase(the Hadoop database) and Cassandra.
- Hive is a Hadoop based system for querying and analysing large volumes of structured data which is stored on HDFS.
- Hive is a query engine built to work on top of Hadoop that can compile queries into MapReduce jobs and run them on the cluster.





jupyterhub



Data Platform

Operator Hub - operatorhub.io



PYTORCH



XGBoost



seldon

Spark

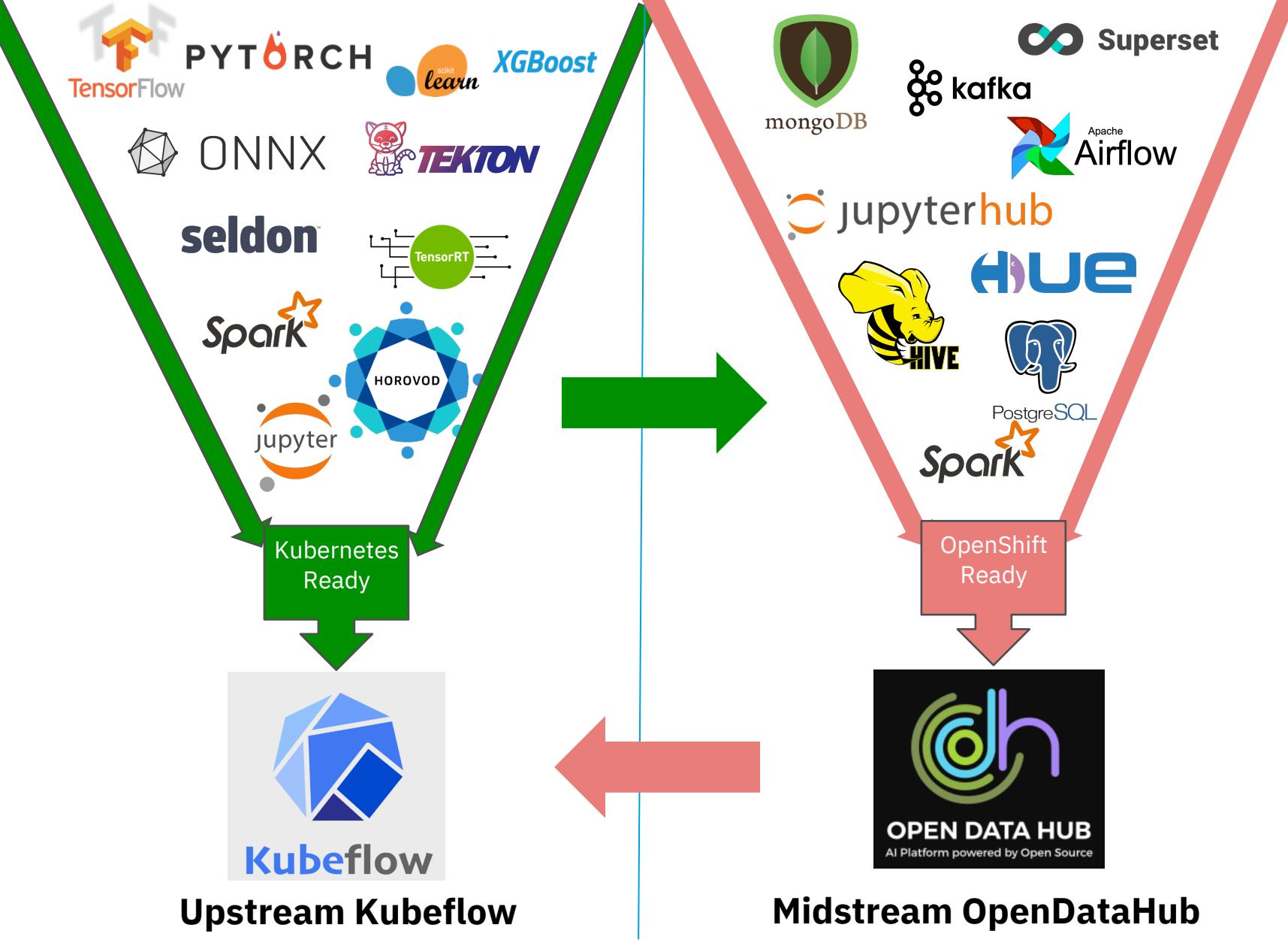
jupyter

Kubernetes
Ready



ML and AI Platform

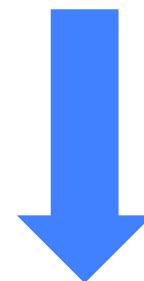
Operator Hub - operatorhub.io



OpenDataHub



Kubeflow



**Open Source End To End
Data and AI Platform**

Upstream Kubeflow

Midstream OpenDataHub

Operator Hub - operatorhub.io

RedHat MarketPlace <https://marketplace.redhat.com/en-us>

Coming Next: Kubeflow Dojo

Date: Wed July 15, 2020

Time	Topic	Presenter
8:00am - 8:30 am	Data and AI Open Source at CODAIT	Animesh
8:30am - 9:30 am	Kubeflow Overview - End to end ML on Kubernetes	Animesh
9:30am - 9:45am	Break	
9:45am - 10:45am	Git and Github	Tom & Morgan
10:45am - 11:00am	Break	
11:00am - 11:30am	Kubeflow development environment	Weiqiang
11:30am - 12:00 pm	Control plane deep dive	Weiqiang
12:00pm - 1:00pm	Lunch break	
1:00pm - 2:00pm	Kubeflow deployment handson	
2:00pm - 3:00pm	Tryout Kubeflow Components	Tommy
3:00pm - 4:00pm	Q&A	

[https://github.com/IBM/
KubeflowDojo](https://github.com/IBM/KubeflowDojo)



<https://github.com/kubeflow>

<https://github.com/opendatahub-io>

Date: Thu July 16, 2020

Time	Topic	Presenter
8:00am - 8:30am	Overview of Kubeflow repos	Tommy
8:30 am - 9:30am	Kubeflow Pipelines deep dive	Animesh, Tommy, Christian
9:30am - 9:45am	Break	
9:45 am - 10:45am	Kubeflow Pipelines-Tekton hands on	Christian Kadner, Tommy Li
10:45am - 11 am	Break	
11:00am - 12 am	KFServing deep dive	Animesh, Tommy
12:00pm - 1:00pm	Lunch break	
1:00pm - 2:00pm	Distributed Training and HPO Deep Dive	Andrew, Kevin, Animesh
2:00pm - 2:15pm	Break	
2:15pm - 2:30pm	Kubeflow PR workflow	Weiqiang
2:30pm - 3:30pm	PR workflow handson	
3:30pm - 4:00pm	Wrap up and final Q&A	Animesh

Kubeflow Dojo: Prerequisites

- Knowledge of Kubernetes, watch the dojo for Kubernetes project with the [IBM internal link](#) or [external link](#)
- Access to a Kubernetes cluster, either minikube or remote hosted
- Source code control and development with git and github, watch the presentation with the [IBM internal link](#) or [external link for git](#) and [external link for pull requests](#)
- Get familiar with golang language, watch the introduction dojo with the [IBM internal link](#) or [external link](#)
- (optional) Knowledge of Istio and knative
- If you have more time,
 - Read [Kubeflow document](#) to learn more about Kubeflow project
 - Browse through Kubeflow [community](#) github



Kubeflow Dojo: Tips for success

- Access to a Kubernetes cluster
 - minimal spec: 8vcpu, 16gb ram and at least 50gb disk for docker registry
- On IBM Kubernetes Service, provision the cluster with machine type b2c.4x16 and 2 worker nodes
- Follow Kubeflow [document](#) to have your cluster prepared
- On IKS cluster, follow this [link](#) to install the IBM Cloud CLI and helm followed by setting up IBM Cloud Block Storage as the default storage class



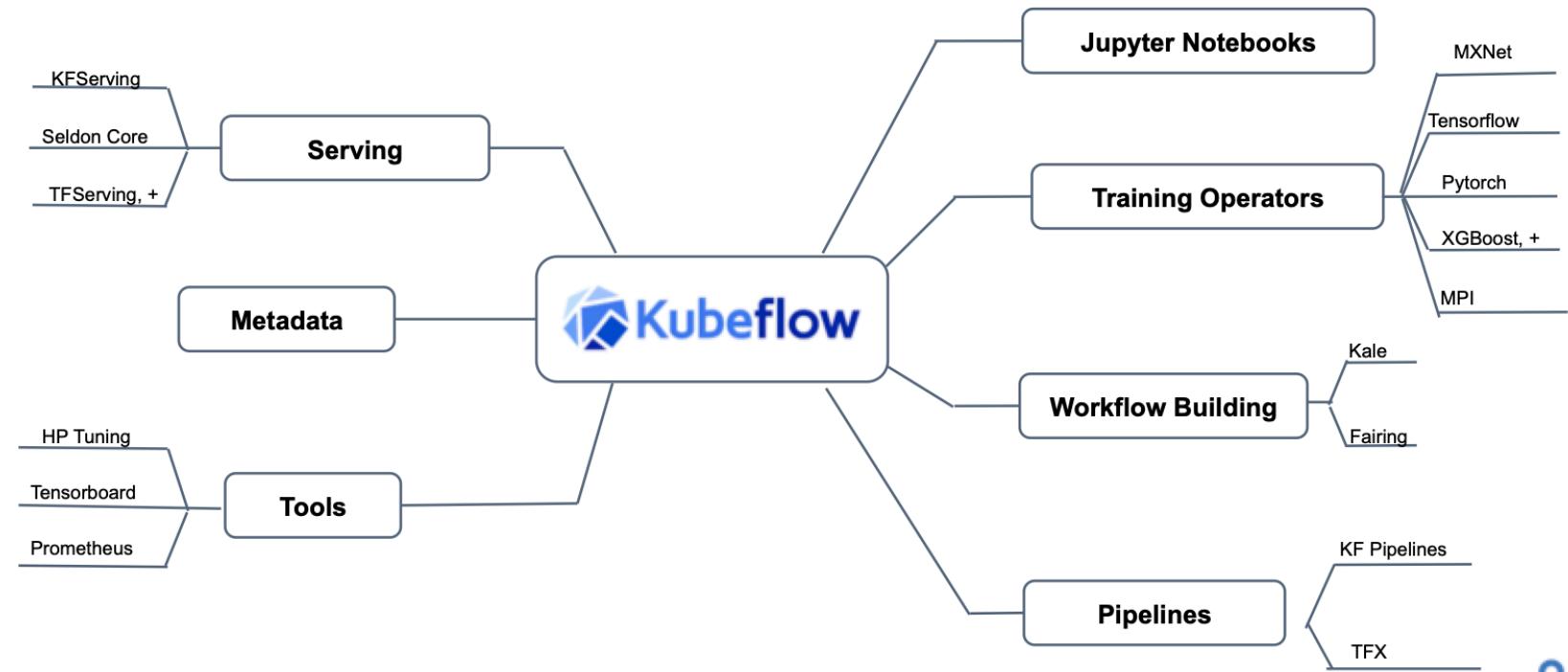
Reach Out!

Animesh Singh

singhan@us.ibm.com

twitter.com/AnimeshSingh

github.com/AnimeshSingh



Kubeflow Dojo: Live
Dates: 15th and 16th July

<https://ec.yourlearning.ibm.com/w3/event/10082348>

Kubeflow Dojo: Virtual
github.com/ibm/KubeflowDojo

