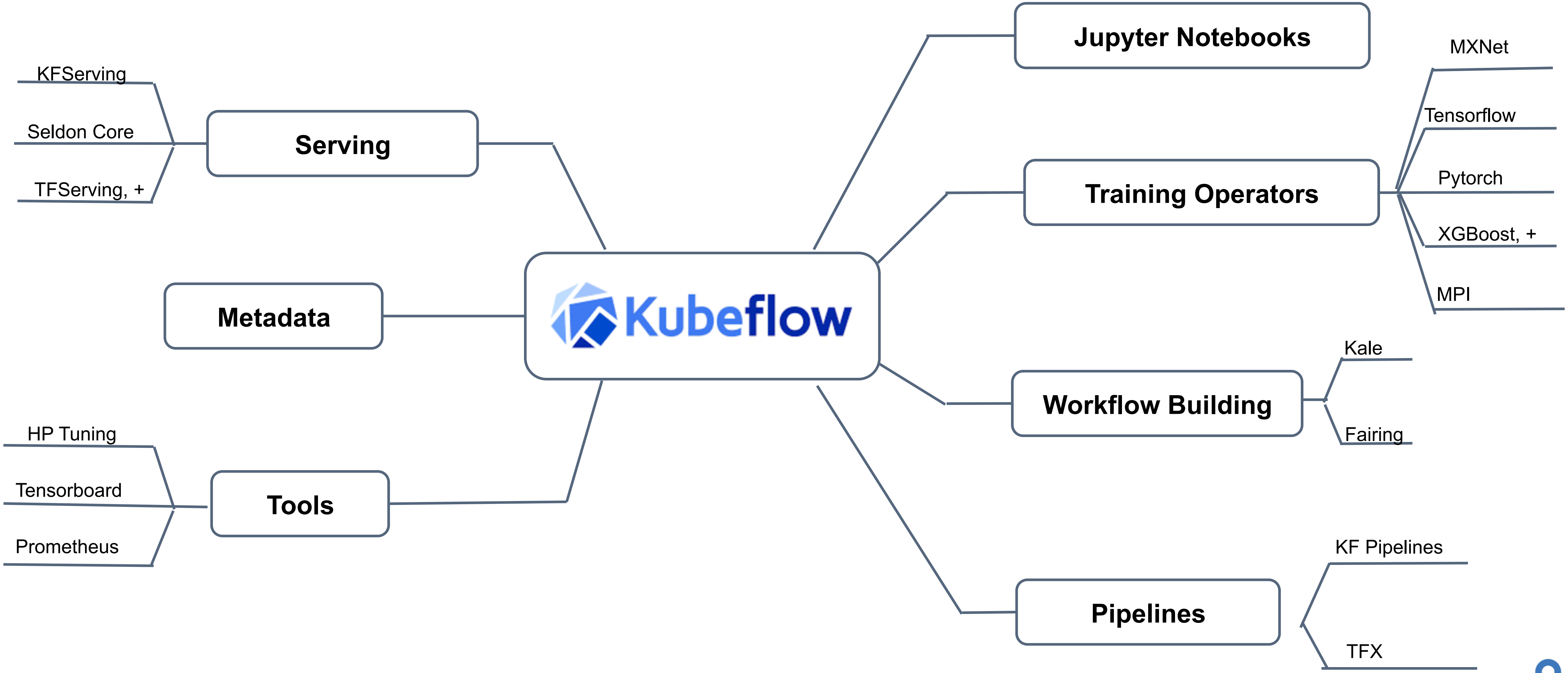
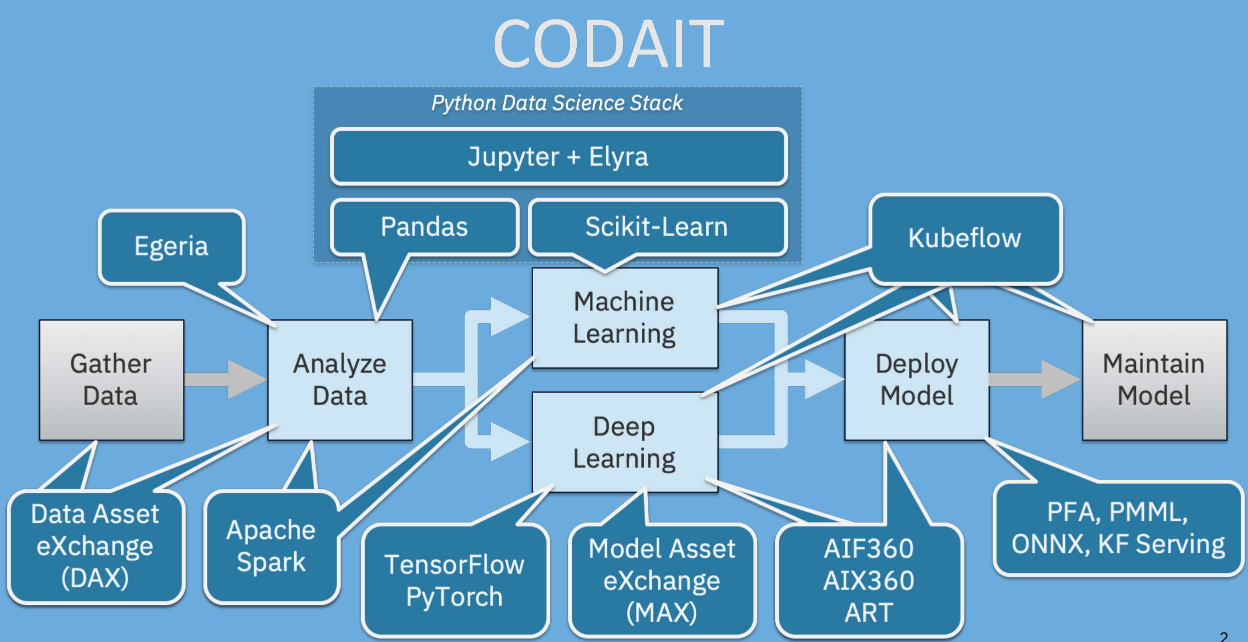
Kubeflow: End to End ML Platform

Animesh Singh



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)

Kubeflow

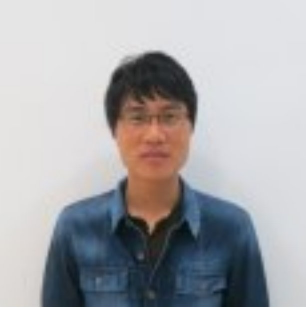
github.com/kubeflow

Kubeflow: Current IBM Contributors

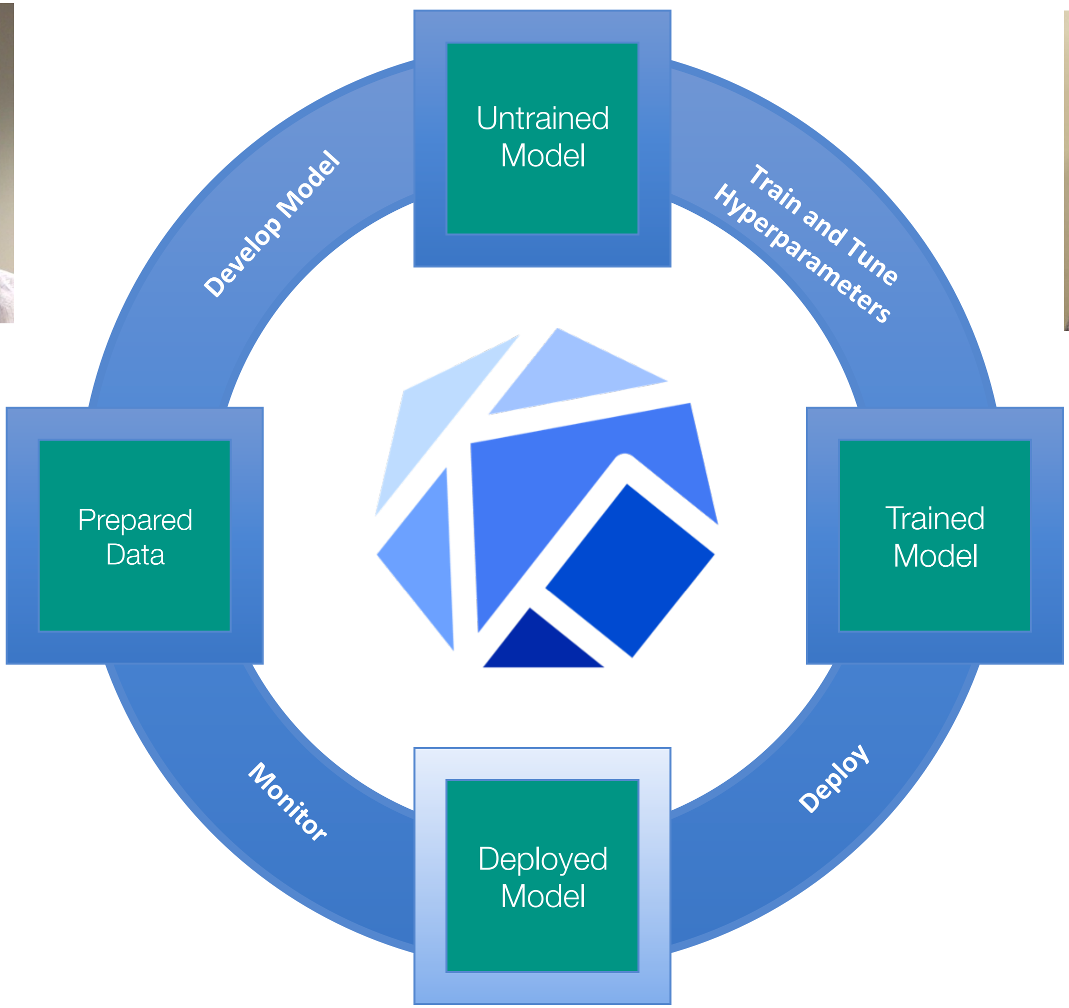
Christian Kadner

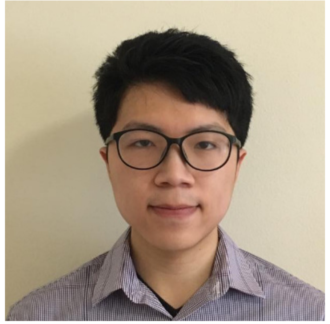
Weiqiang Zhuang

Jin Chi He



Feng Li

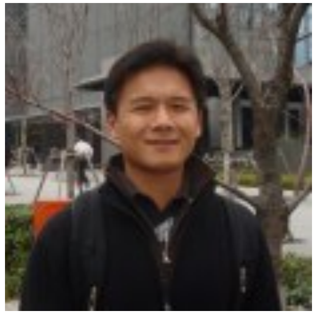




Tommy Li

Andrew Butler

Ke Zhu



Kevin Yu

IBM is the 2nd Largest Contributor

Commits by Company

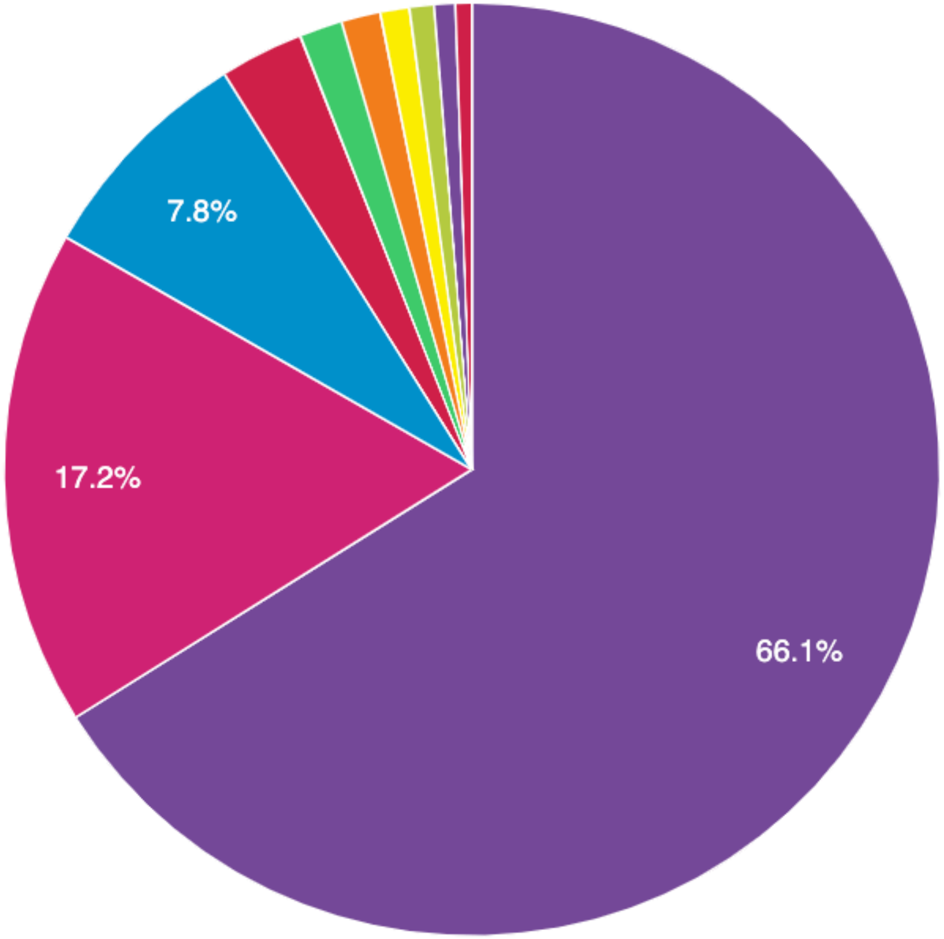
Show 10 entries

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

Google	22064
IBM	4727
Cisco	4009
Caicloud	1865
Amazon	1425
Microsoft	553
Seldon	449
Net EASE	266
NetEase	260
Arrikto	213
DaoCloud	143
Huawei	139
NVidia	80
Oracle	78
Alibaba	70
Dell	63
Red Hat	52
Intel	50

IBMers contributing across projects in Kubeflow

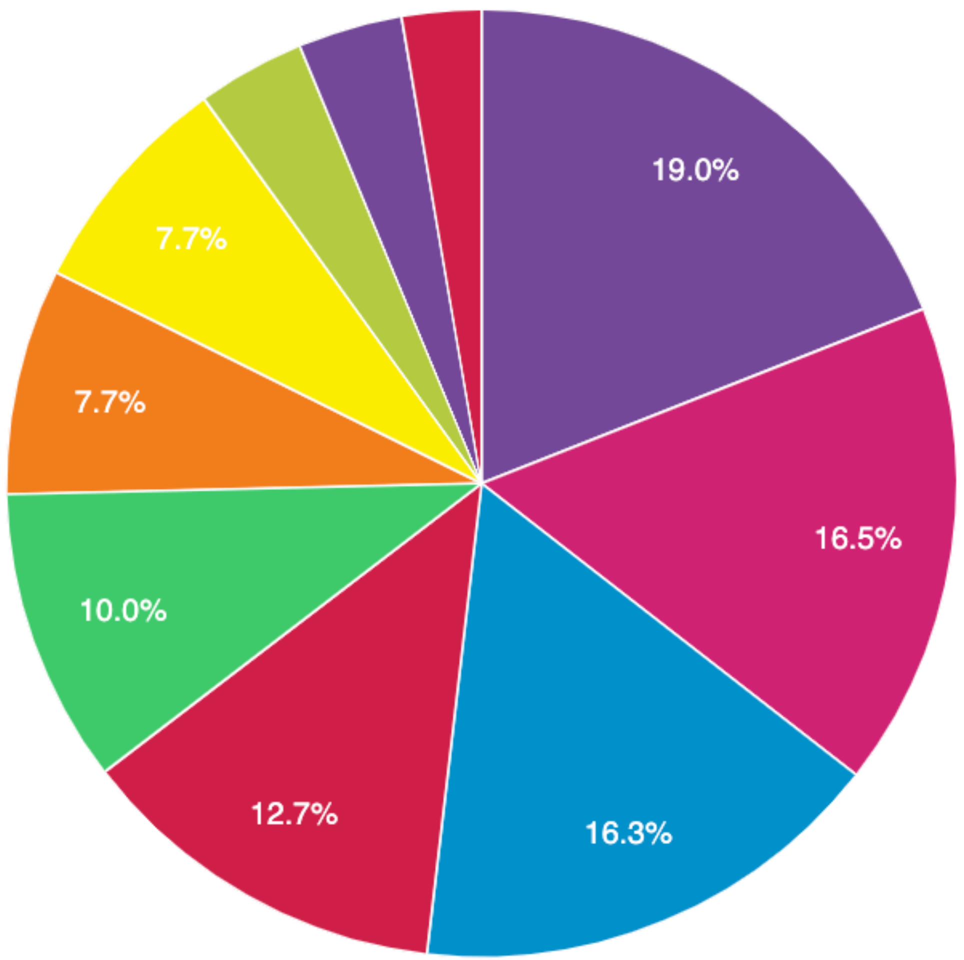
Show 10 entries

Search

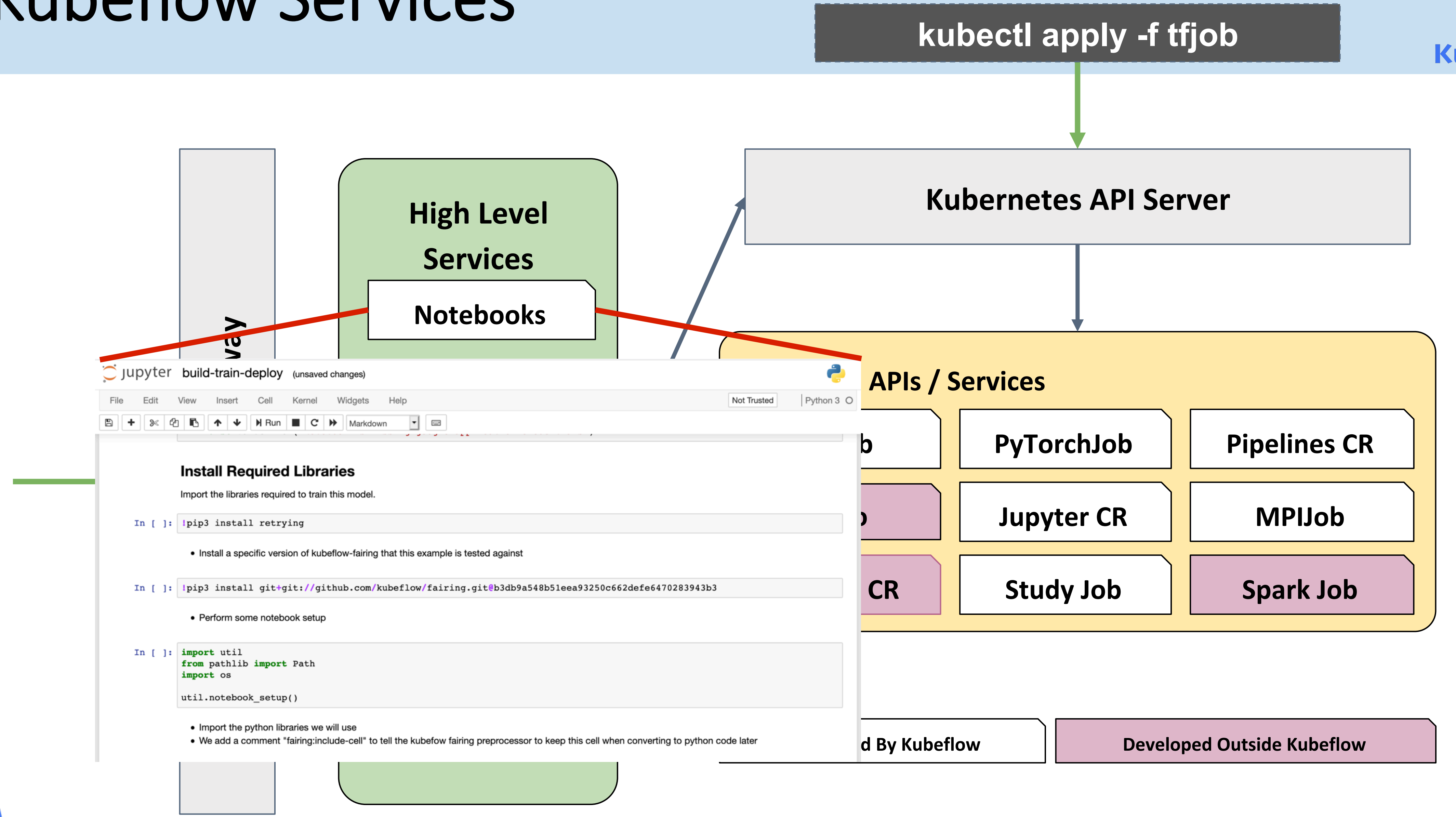
#	Module	Commits
1	kfserving@kubeflow	155
2	kfp-tekton@kubeflow	135
3	katib@kubeflow	133
4	website@kubeflow	104
5	fairing@kubeflow	63
6	pipelines@kubeflow	63
7	examples@kubeflow	30
8	kfctl@kubeflow	29
9	kubeflow	22
10	manifests@kubeflow	21

Showing 1 to 10 of 18 entries

Previous Next



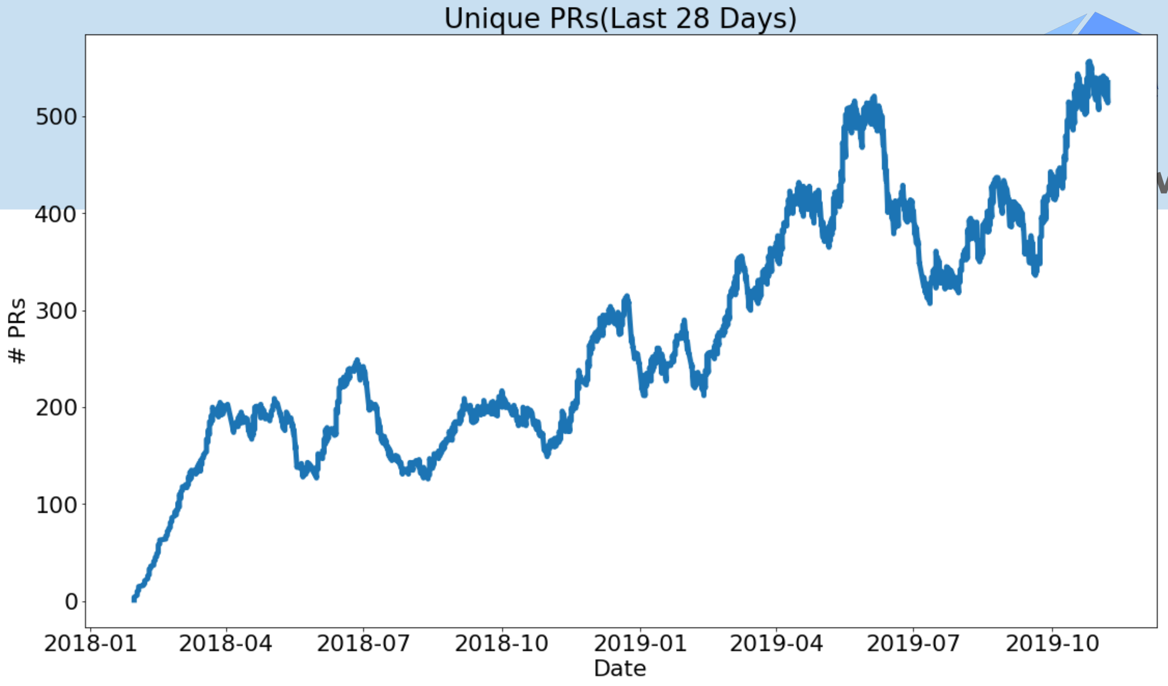
Kubeflow Services

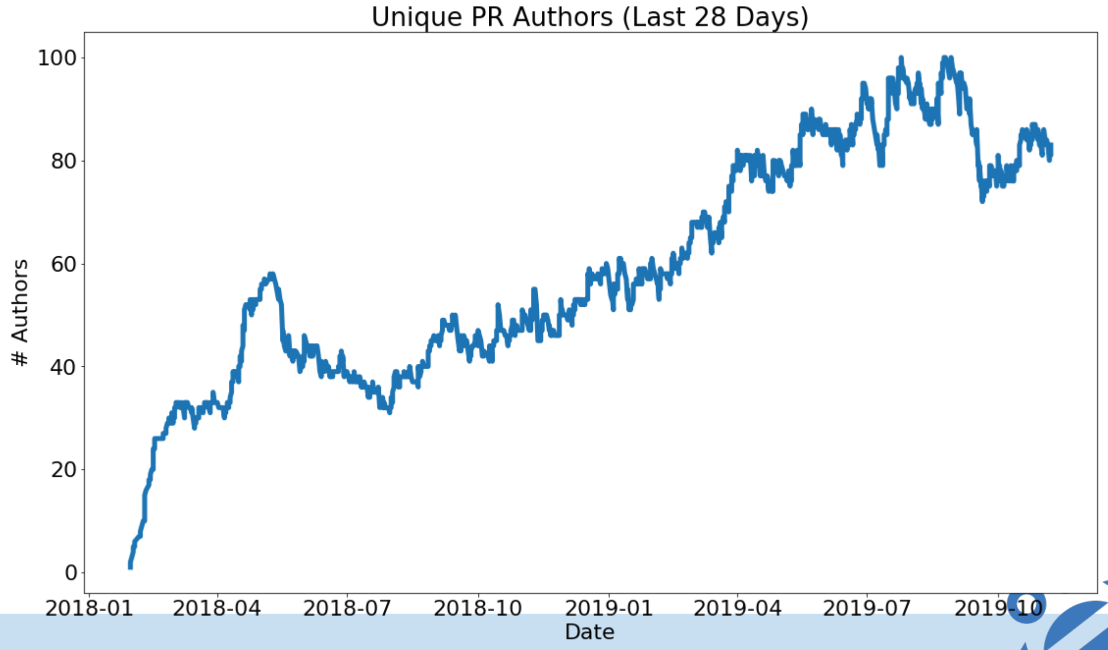


Adapted from Kubeflow Contributor Summit 2019 talk: Kubeflow and ML Landscape (Not all components are shown)

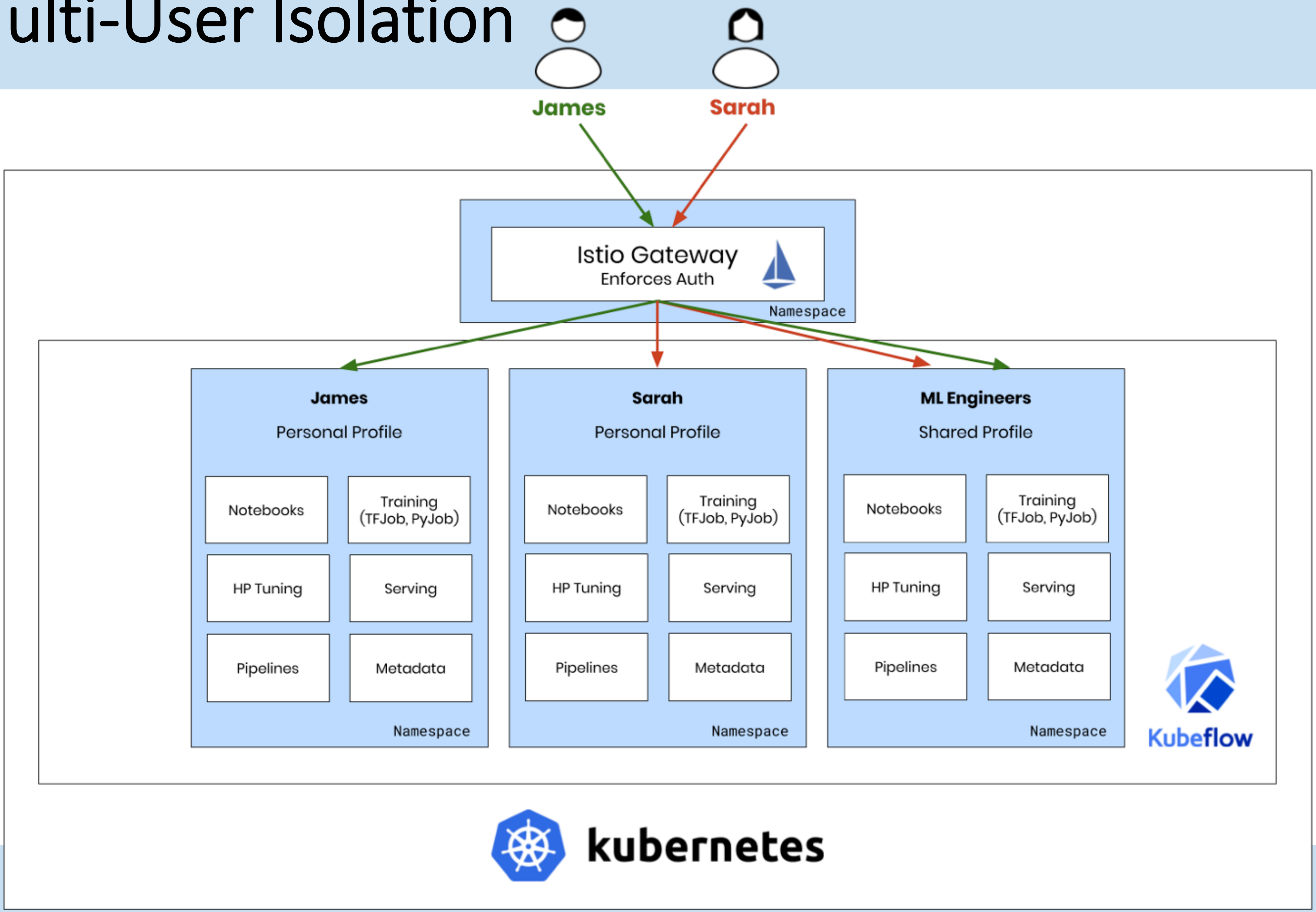
Community is growing!

Alibaba Cloud
aws
蚂蚁金服 ANT FINANCIAL
argo
Arrikto
Bloomberg
Caicloud
CANONICAL
cisco
DATAWIRE
DELL
GitHub
Google
GO-JEK
heptio
IBM
iguazio
intel
in
Microsoft
MOMENTA
One Convergence
Pachyderm
PRIMER
jupyter
redhat
SELDON
shopify
tusimple
UBER
云知声 Unisound
weaveworks

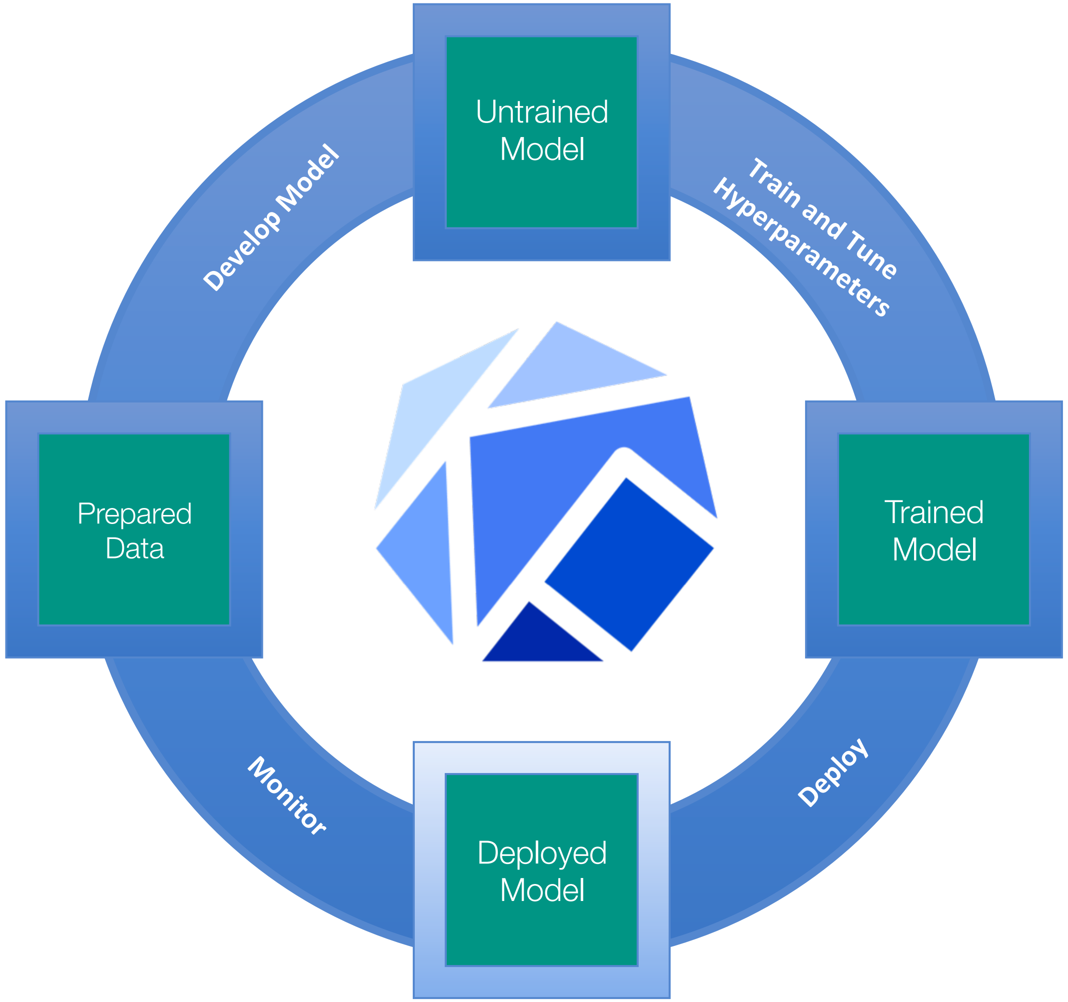




Multi-User Isolation



ML Lifecycle: Build: Development, Training and HPO

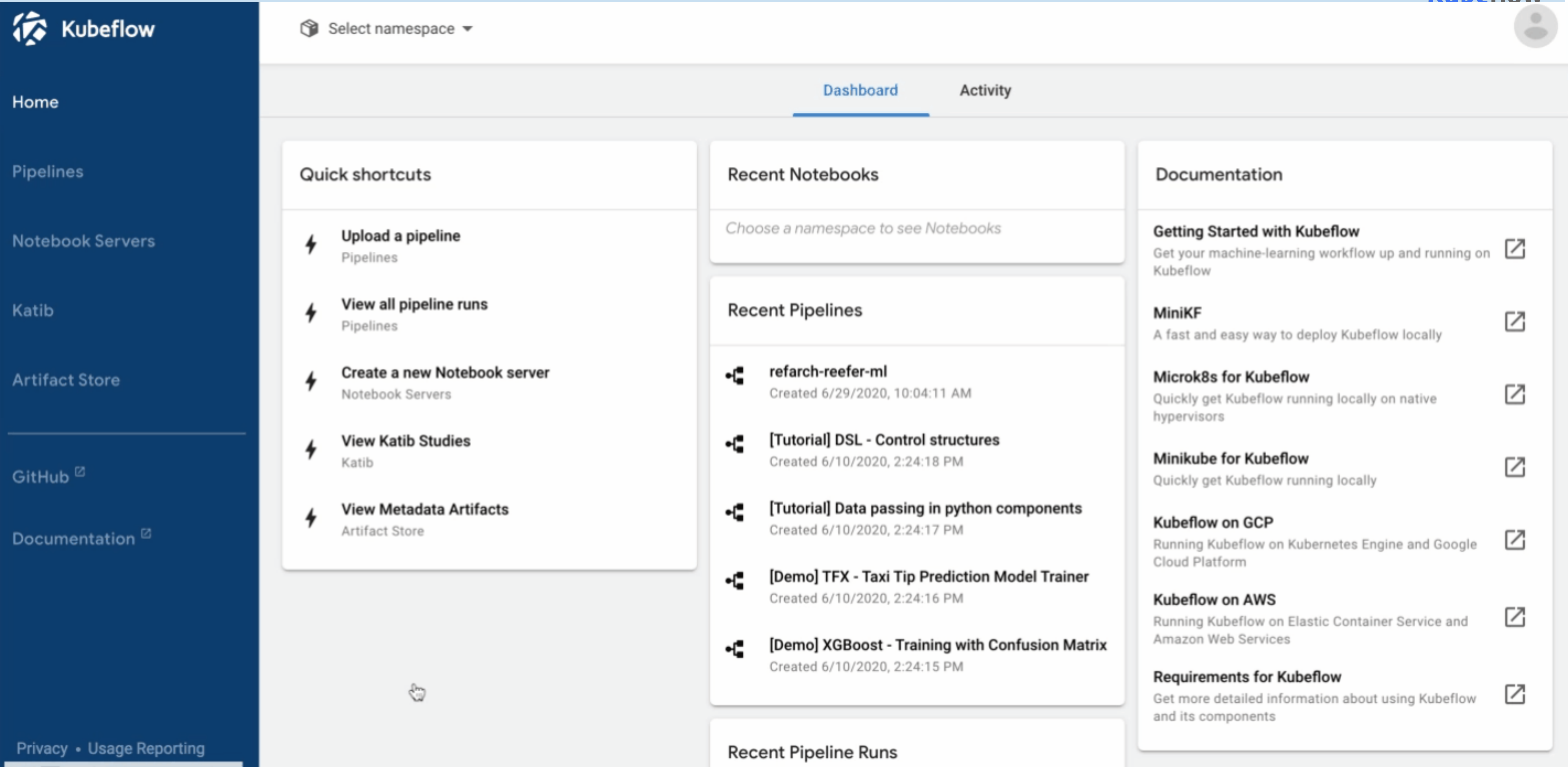


Develop (Kubeflow Jupyter Notebooks)

- 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

Develop (Kubeflow Jupyter Notebooks)



Distributed Training Operators

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

Distributed Training Operators

mxnet

tf-operator
Tools for ML/Tensorflow on Kubernetes.
Jsonnet · Apache-2.0 · 323 · ★

pytorch-operator
PyTorch on Kubernetes
Jsonnet · Apache-2.0 · 87 · ★

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

mxnet-operator
A Kubernetes operator for mxnet jobs
Go · Apache-2.0 · 20 · ★ 50

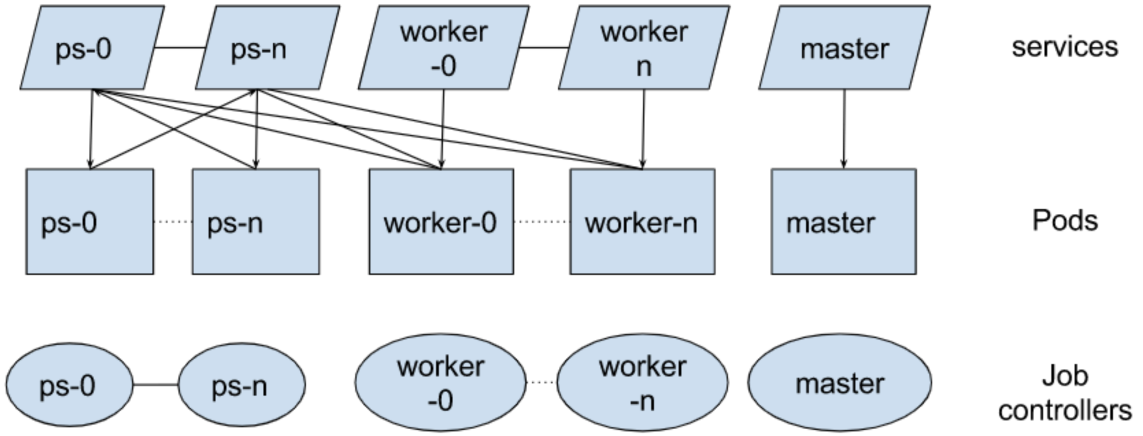
chainer-operator
Repository for chainer operator
Go · Apache-2.0 · 9 · ★ 12

XGBoost

Chainer

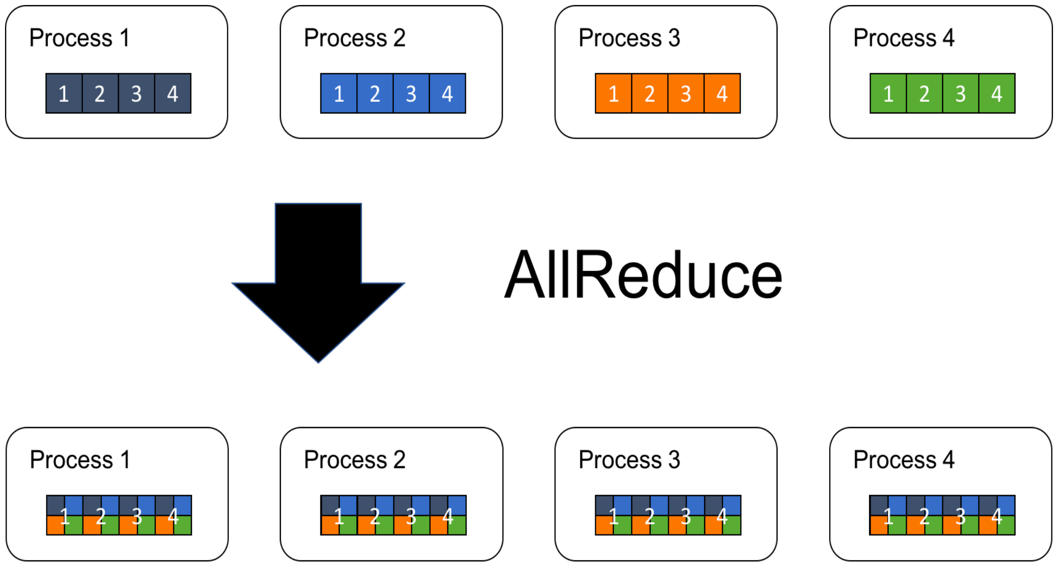
Distributed Tensorflow Operator

- 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

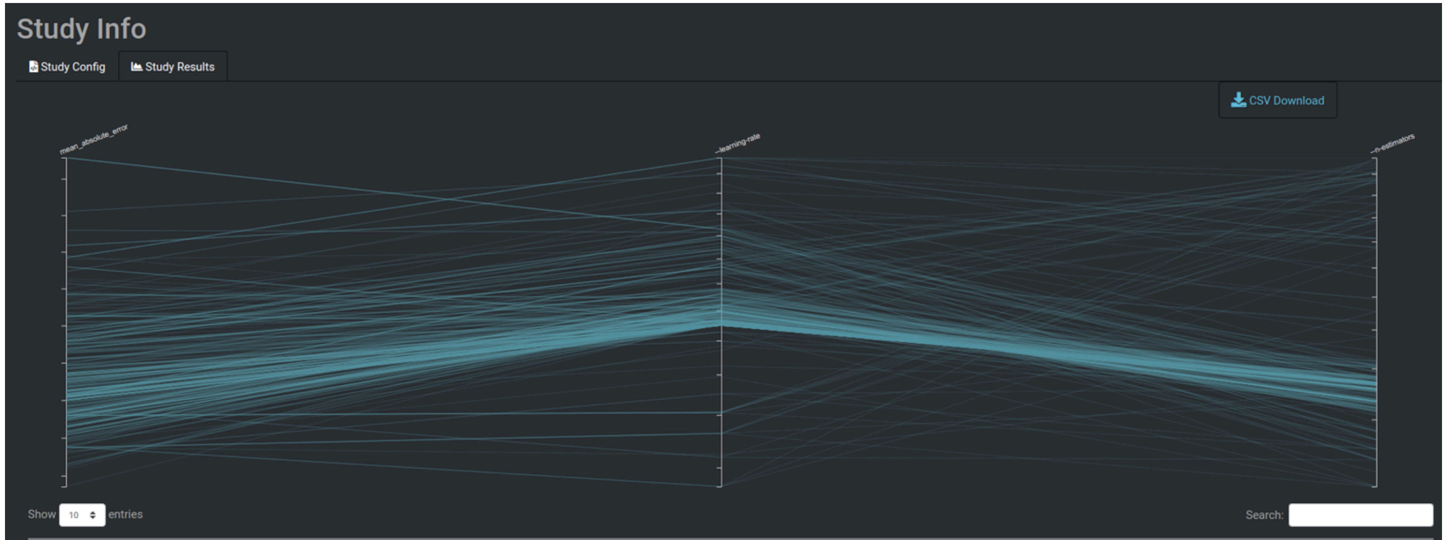
- AllReduce is an operation that reduces many arrays spread across multiple processes into a single array which can be returned to all the processes
- This ensures consistency between distributed processes while allowing all of them to take on different workloads
- The operation used to reduce the multiple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



Hyper Parameter Optimization and Neural Architecture Search - Katib

- Katib: Kubernetes Native System for Automated tuning of machine learning model's Hyperparameter Turning 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)



WorkerID	TrialID	mean_absolute_error	--learning-rate	--n-estimators
a082b67ad67c63ba	vdca23208a96d392	18516.93	0.16555555555555554	10265
a1025286d7d64a01	h4fe53b19226f9e1	17199.54	0.09000132971800572	8417
a20ca959e78c8af2	a07fe53b73c8ae70	16954.29	0.09078646547782351	8348
a2240a2afd13058e	d80006060888aae6	17555.15	0.15962962962962962	5336
a2e4e2de4740e26d	rb265a36a14496ce	16889.88	0.09083523853071178	8424
a2fa7a43a6029460	z53ac2fa7266a6e1	17589.17	0.09609053497942387	8234

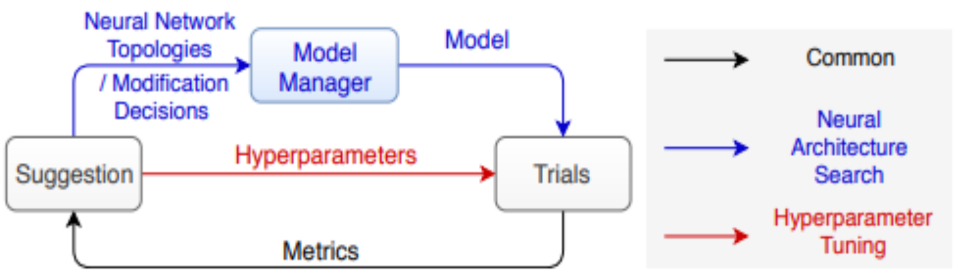
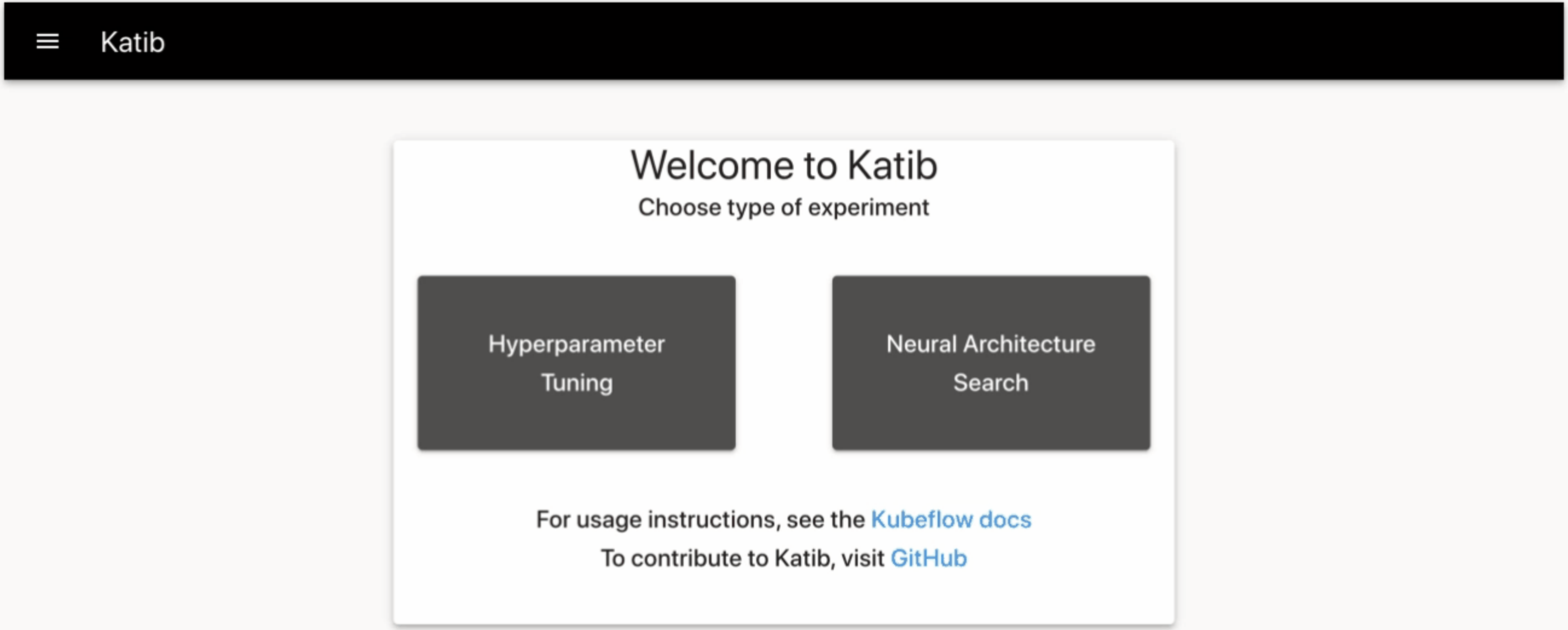


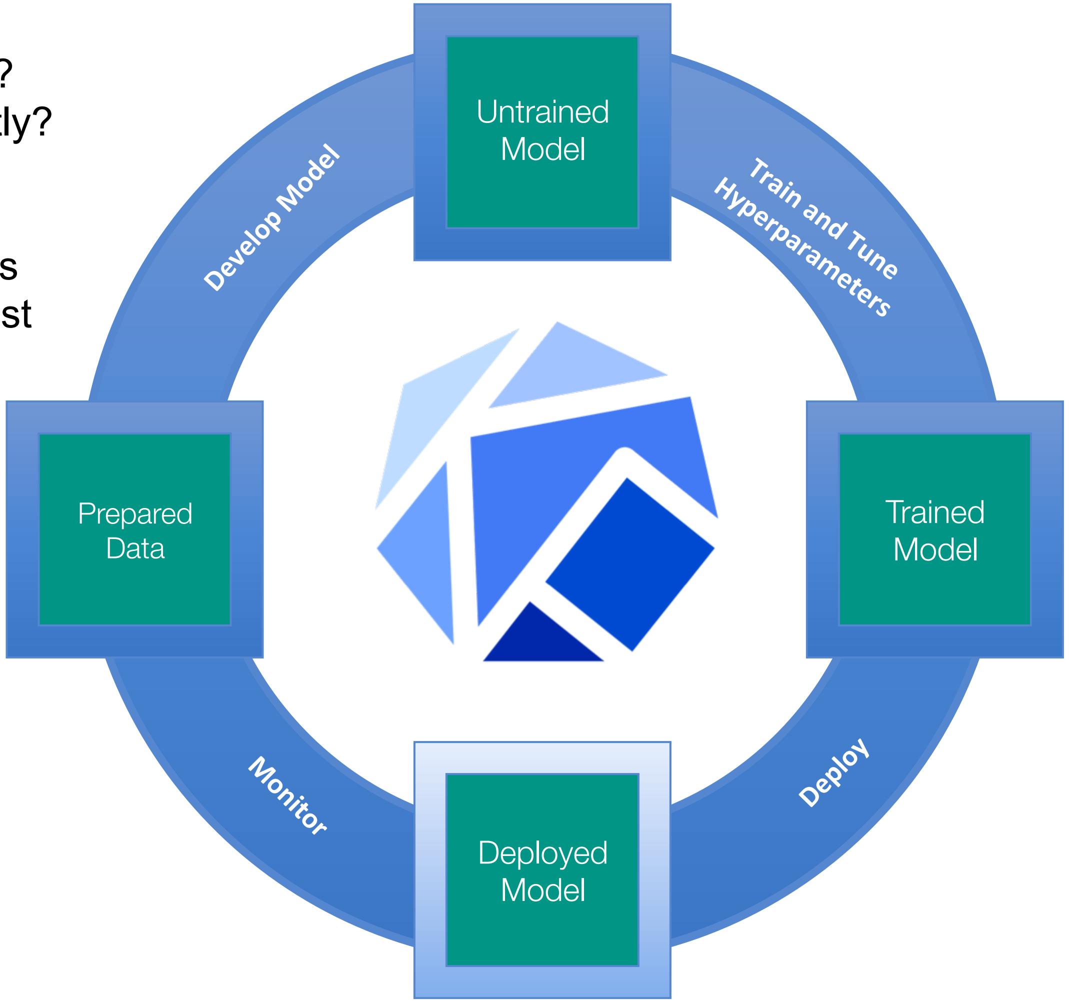
Figure 1: Summary of AutoML workflows

Katib


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

ML Lifecycle: Production Model Serving

- 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

Experts fragmented across industry

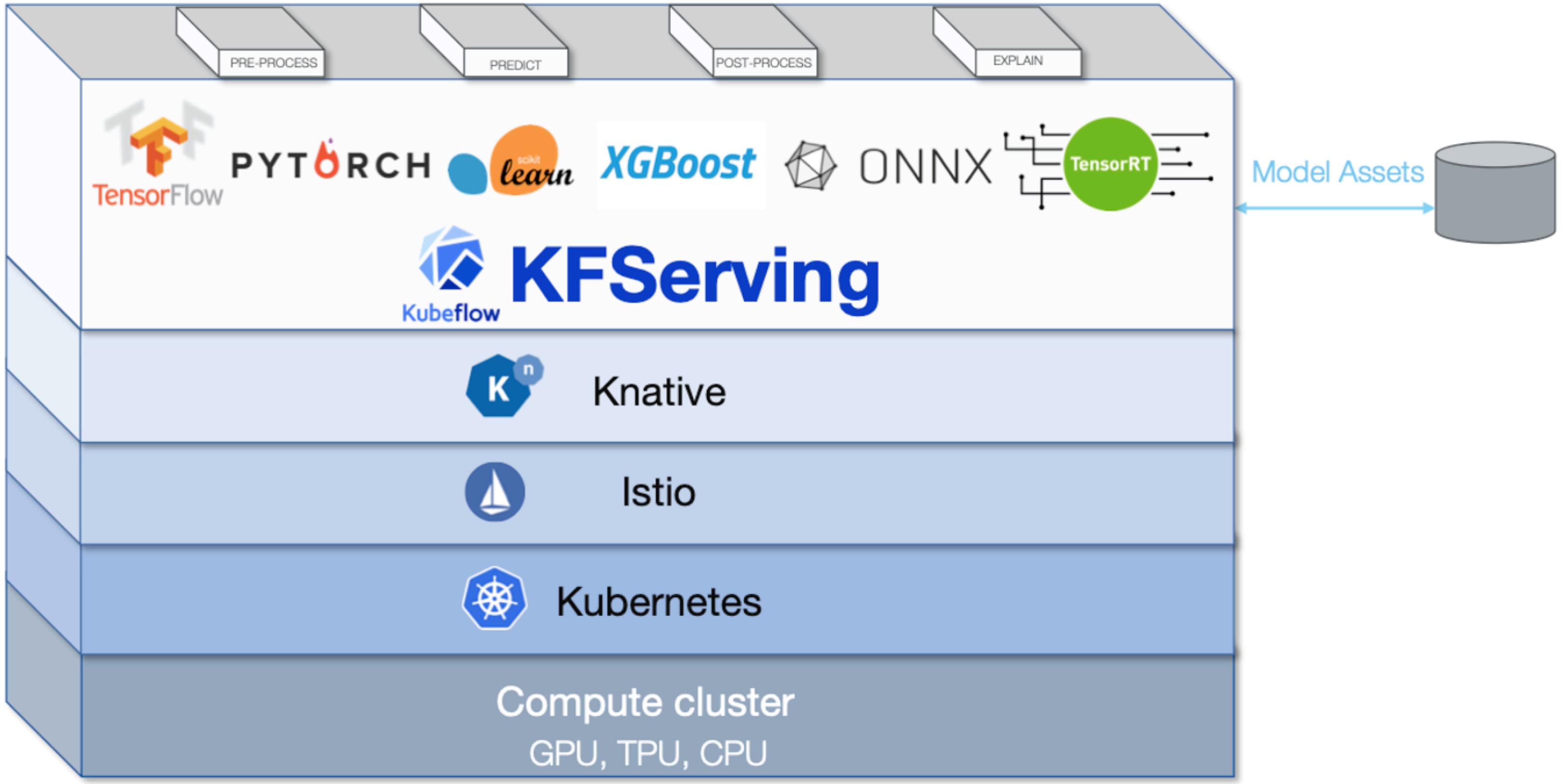
- Seldon Core was pioneering Graph Inferencing.
- IBM and Bloomberg were exploring serverless ML lambdas. IBM gave a talk on the ML Serving with Knative at last KubeCon in Seattle
- Google had built a common Tensorflow HTTP API for models.
- Microsoft Kubernetizing their Azure ML Stack

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

Model Serving - KFServing

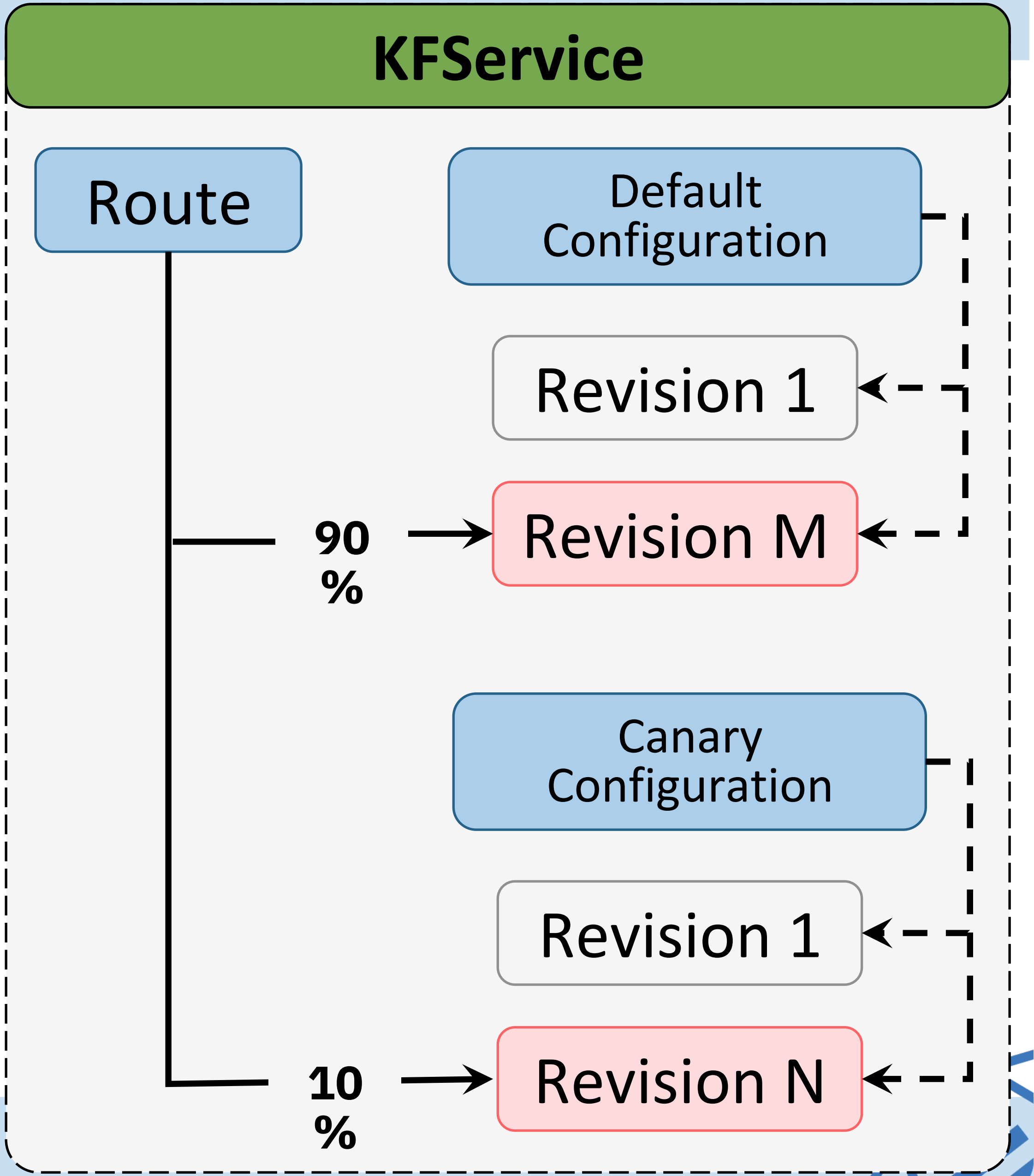
- Founded by Google, Seldon, IBM, Bloomberg and Microsoft
- Part of the Kubeflow project
- Focus on 80% use cases - single model rollout and update
- Kfserving 1.0 goals:
  - Serverless ML Inference
  - Canary rollouts
  - Model Explanations
  - Optional Pre/Post processing



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



Supported Frameworks, Components and Storage Subsystems

Model Servers

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

Components:

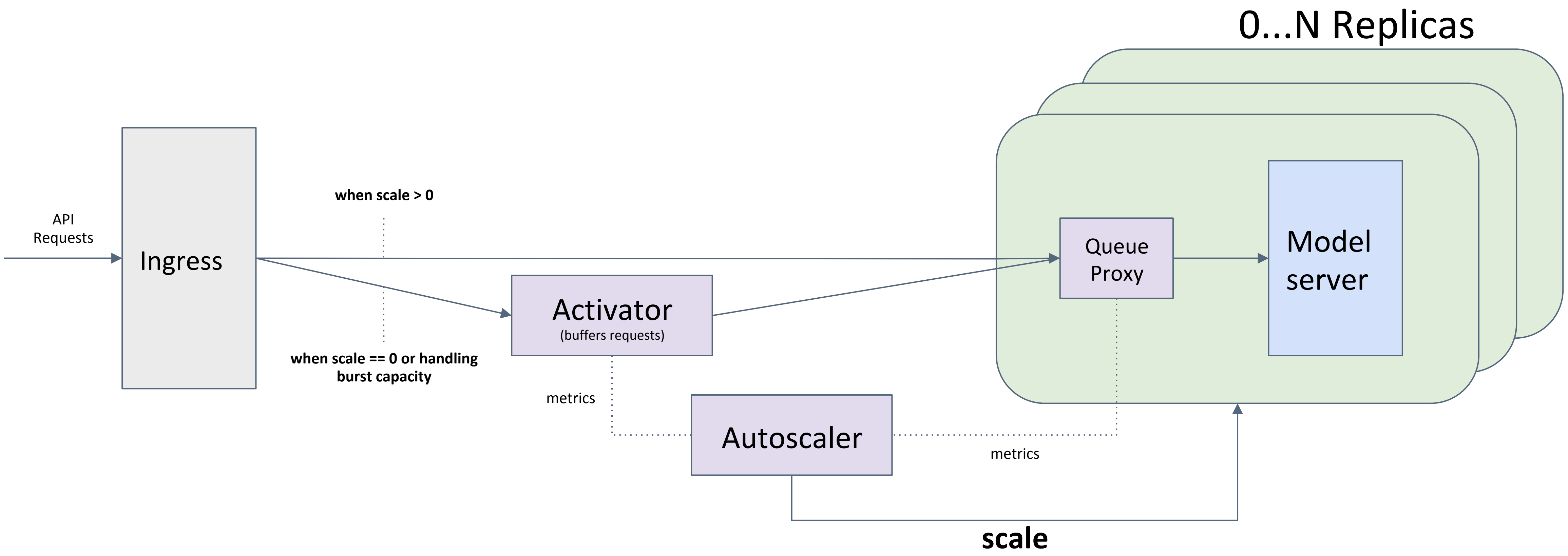
- - Predictor, Explainer, Transformer (pre-processor, post-processor)

Storage

- AWS/S3
- GCS
- Azure Blob
- PVC

GPU Autoscaling - KNative solution

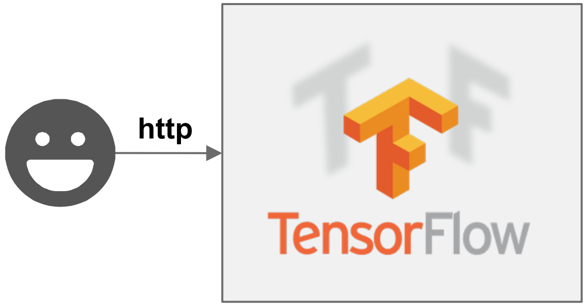
- Scale based on # in-flight requests against expected concurrency
- Simple solution for heterogeneous ML inference autoscaling



But the Data Scientist Sees...

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

=

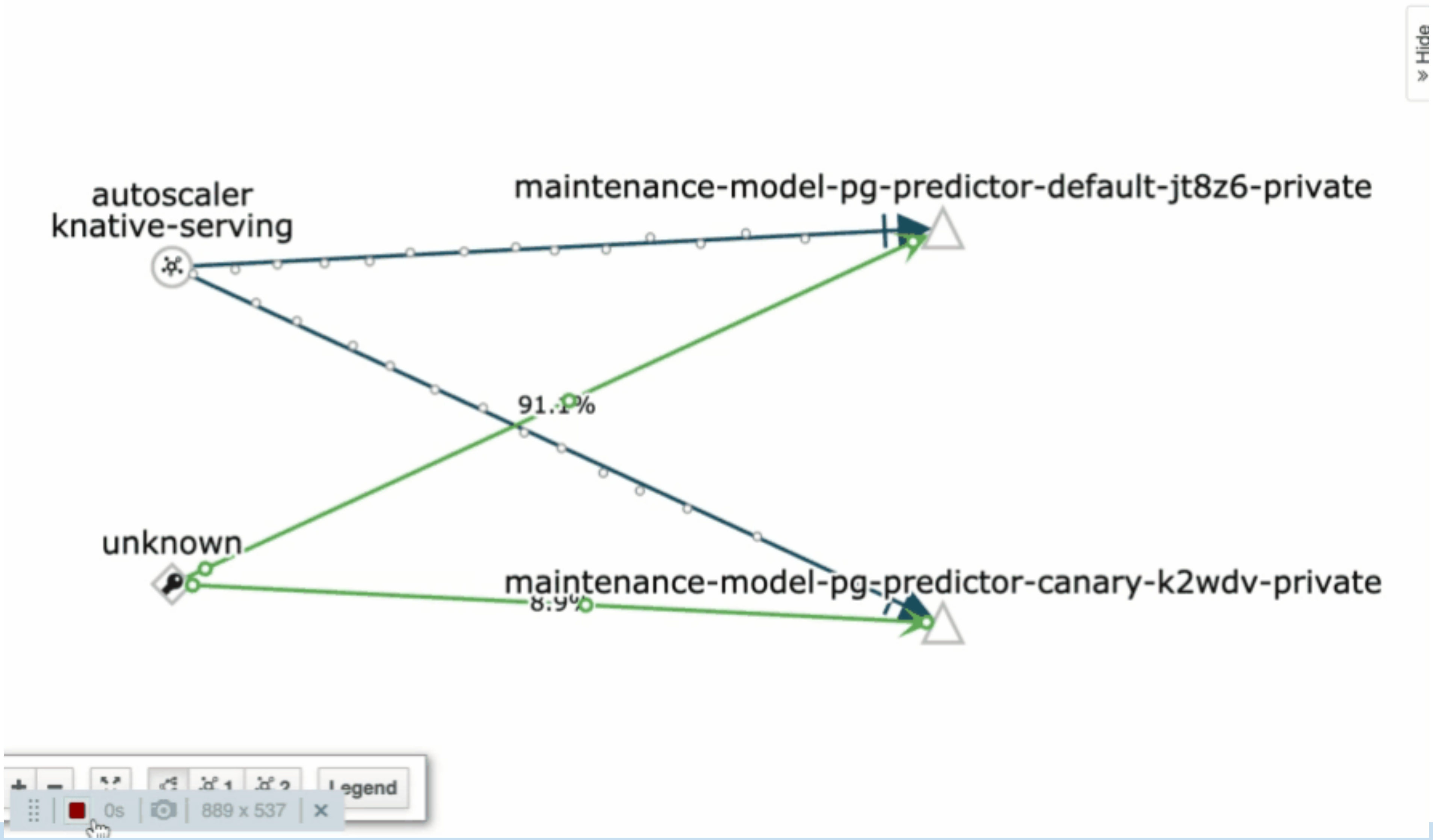


- A pointer to a Serialized Model File
- 9 lines of YAML
- A live model at an HTTP endpoint

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

Production users include: Bloomberg, CoreWeave, gojek

KFServing: Default, Canary and Autoscaler



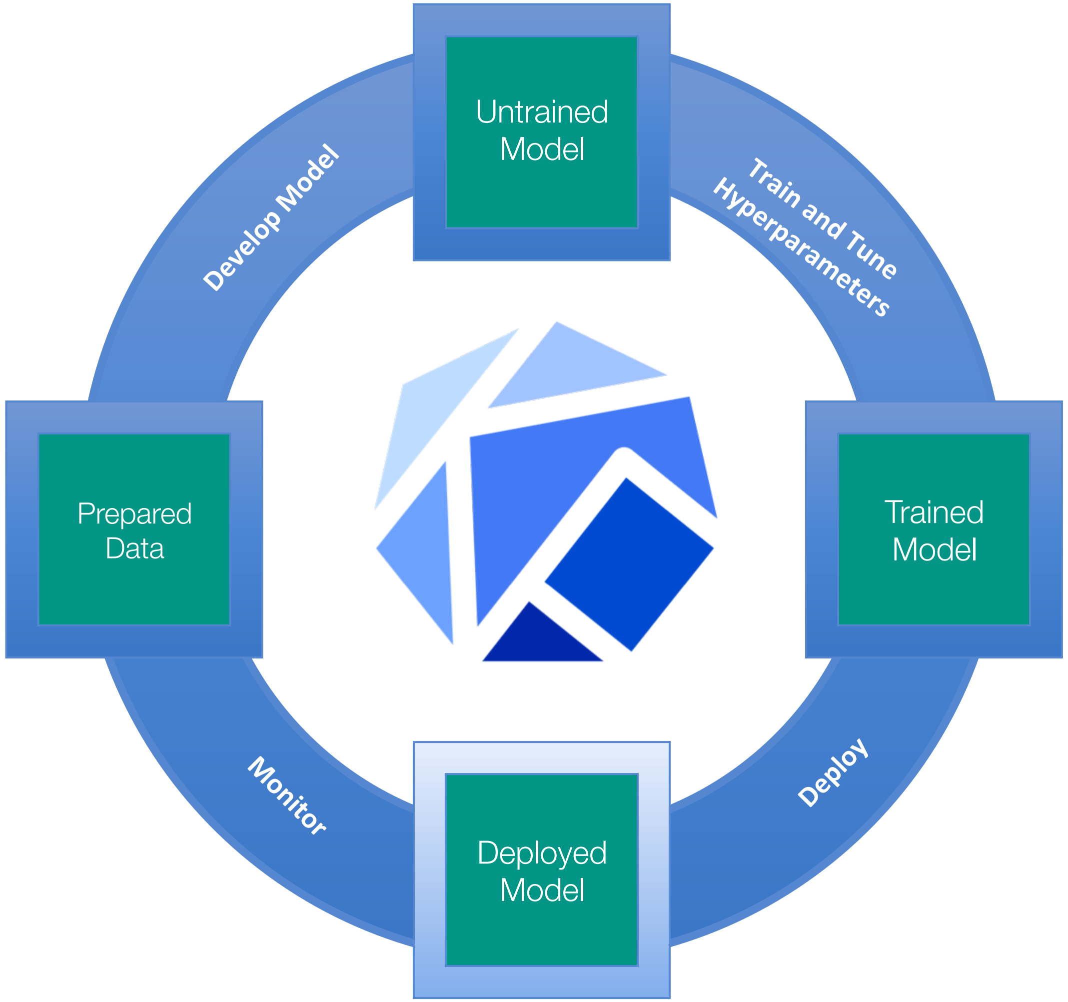
KFServing – Existing Features

- Crowd sourced capabilities – Contributions by AWS, Bloomberg, Google, Seldon, IBM, NVidia and others.
- Support for multiple runtimes pre-integrated (TFServing, Nvdia 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

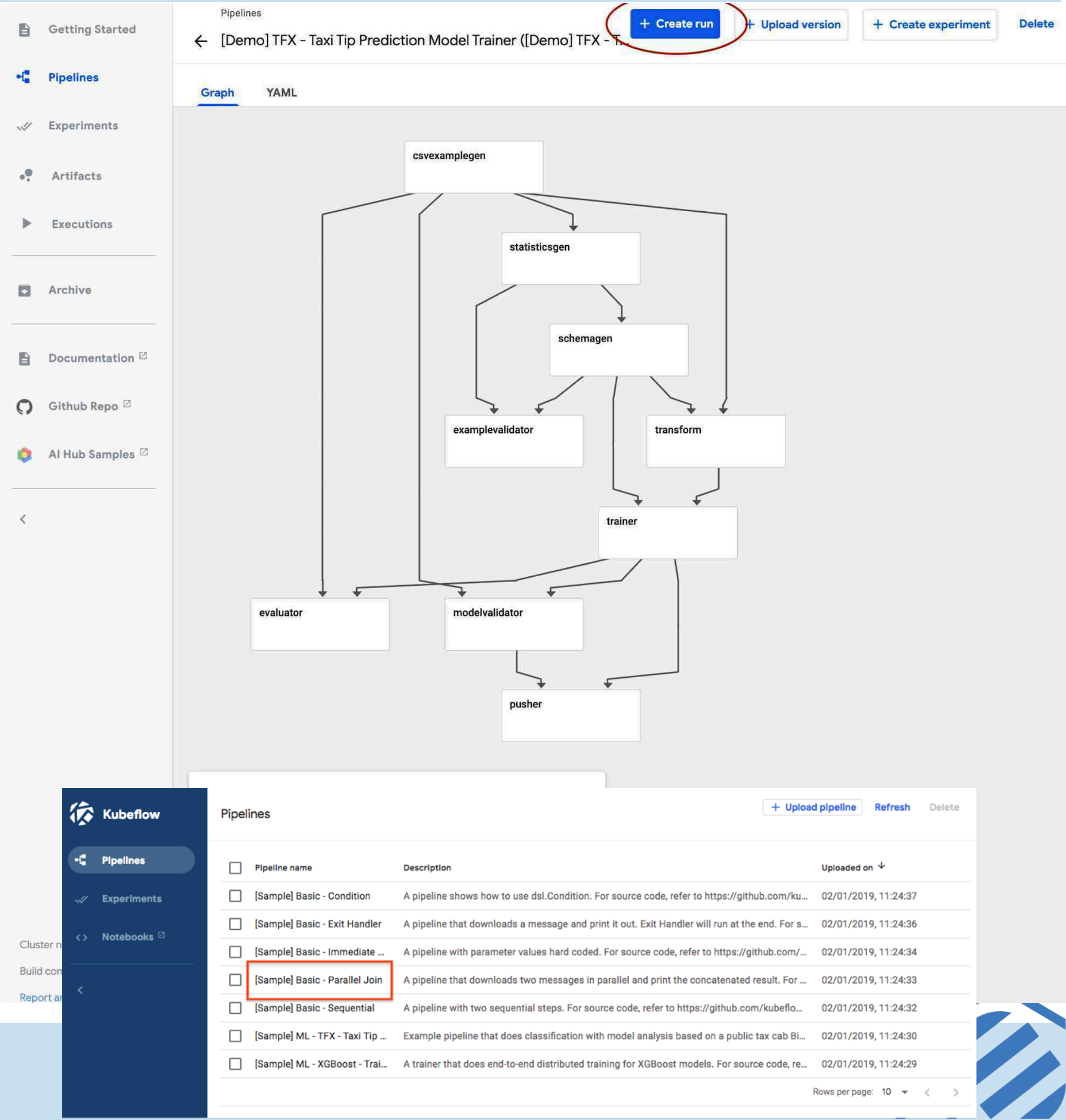
- MMS: Multi-Model-Serving for serving multiple models per custom KFService instance
- More Data Plane v2 API Compliant Servers: SKLearn, XGBoost, PyTorch…
- Multi-Model-Graphs and Pipelines: Support chaining multiple models together in a Pipelines
- PyTorch support via AWS TorchServe
- gRPC Support 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

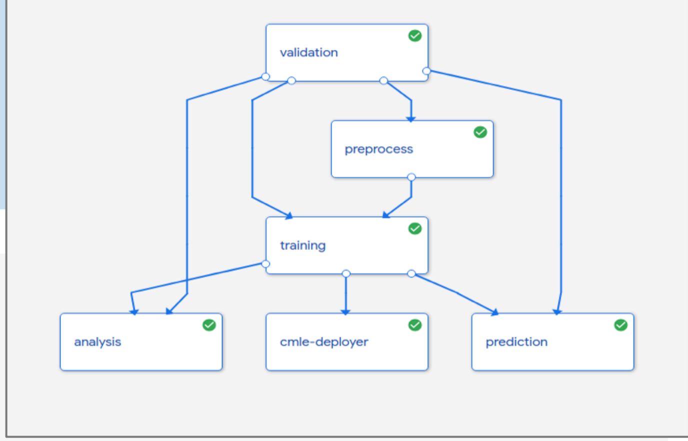


Kubeflow Pipelines

- 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



Define Pipeline with Python SDK



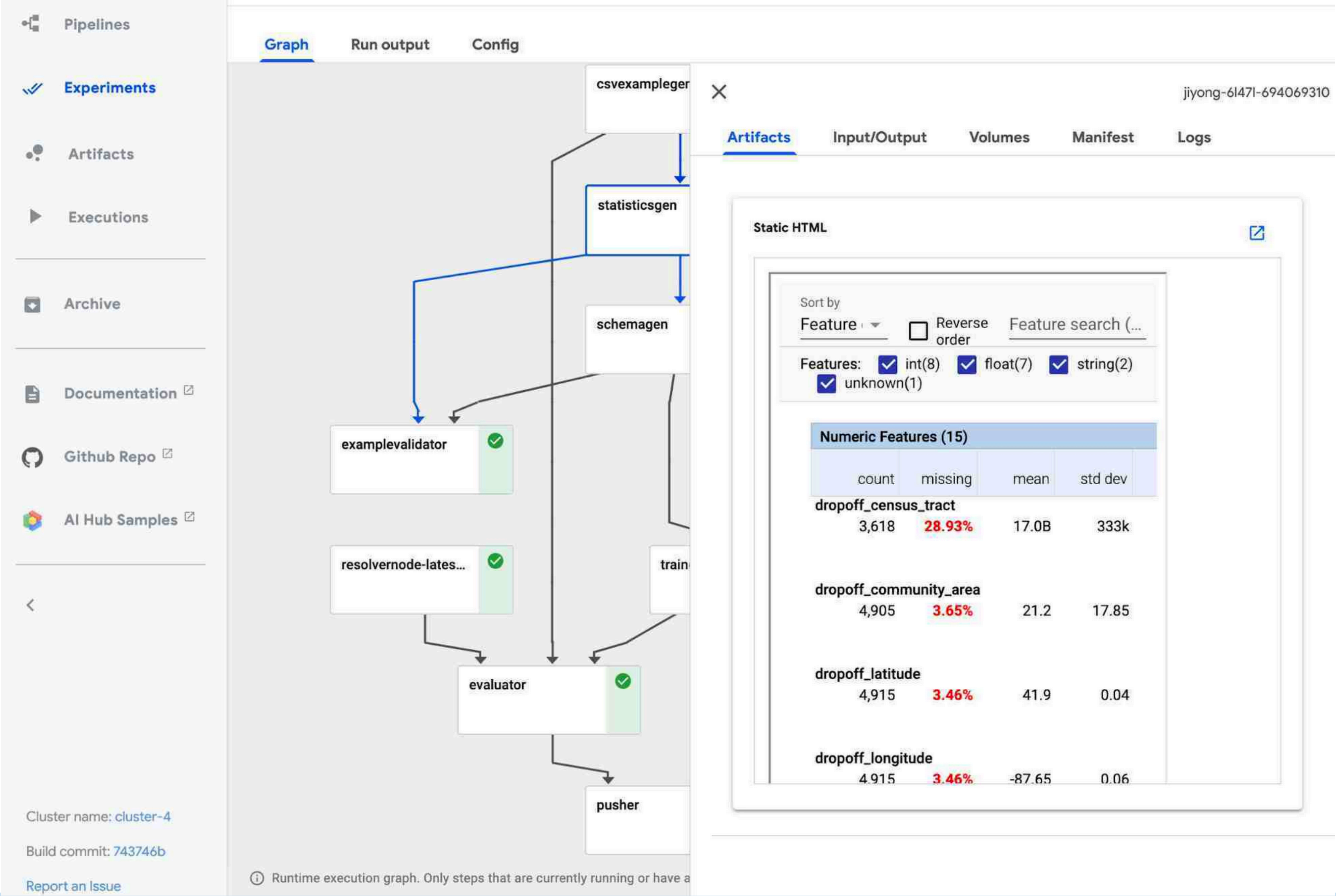

```
@dsl.pipeline(name='Taxi Cab Classification Pipeline Example’)
def taxi_cab_classification(
    output_dir,
    project,
    Train_data      = 'gs://bucket/train.csv',
    Evaluation_data = 'gs://bucket/eval.csv',
    Target          = 'tips',
    Learning_rate   = 0.1, hidden_layer_size = '100,50’,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

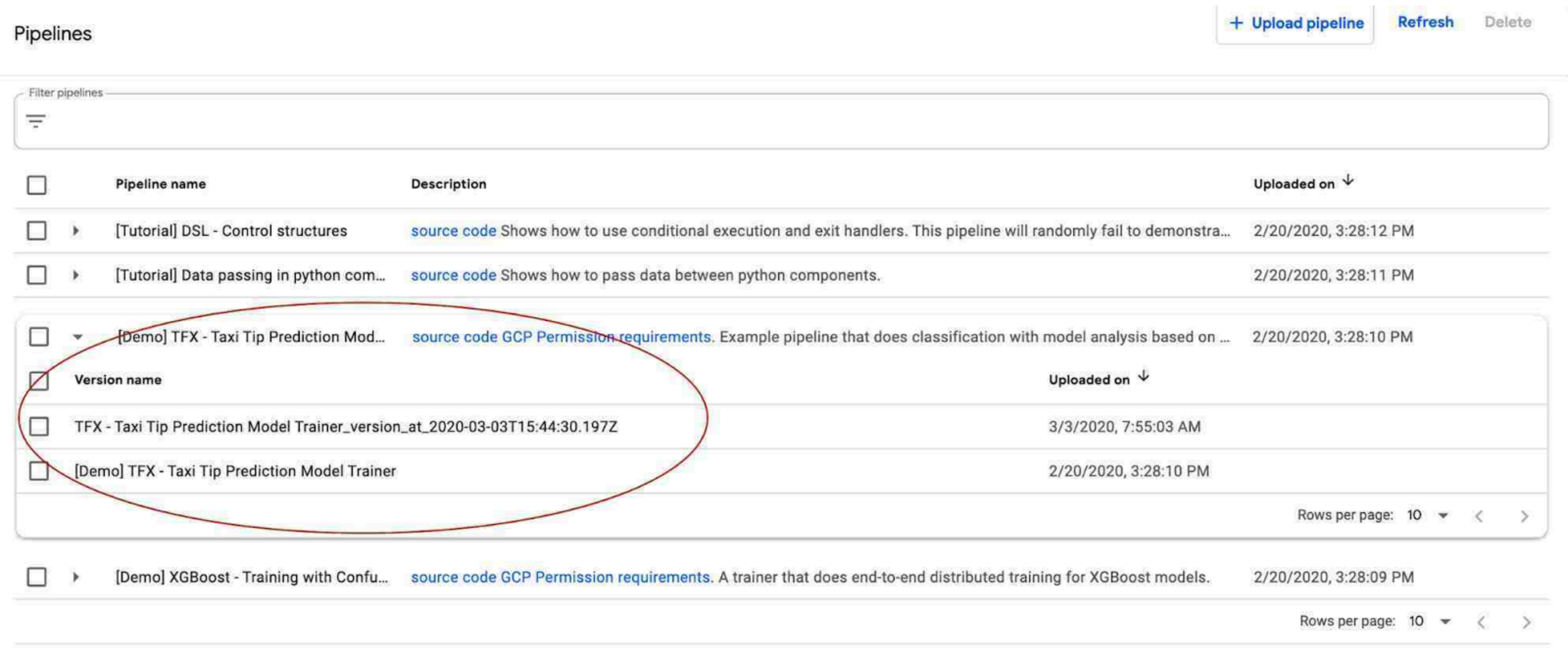
```
dsl.compile(taxi_cab_classification,  'tfx.tar.gz')
run = client.run_pipeline(
      'tfx_run', 'tfx.tar.gz', params={'output': ‘gs://dpa22’, 'project': ‘my-project-33’})
```

Visualize the state of various components



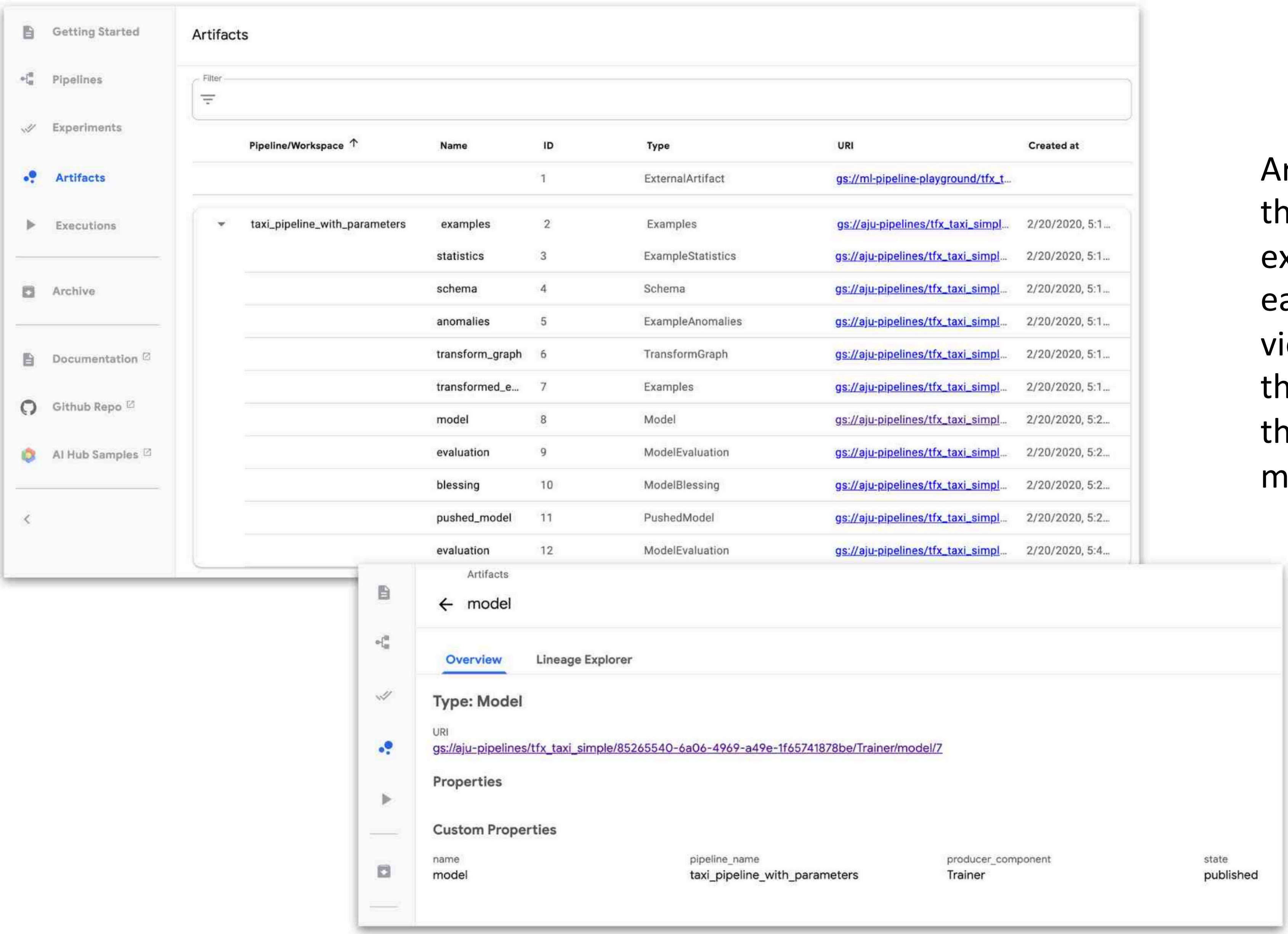
Pipelines versioning

Kubeflow



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

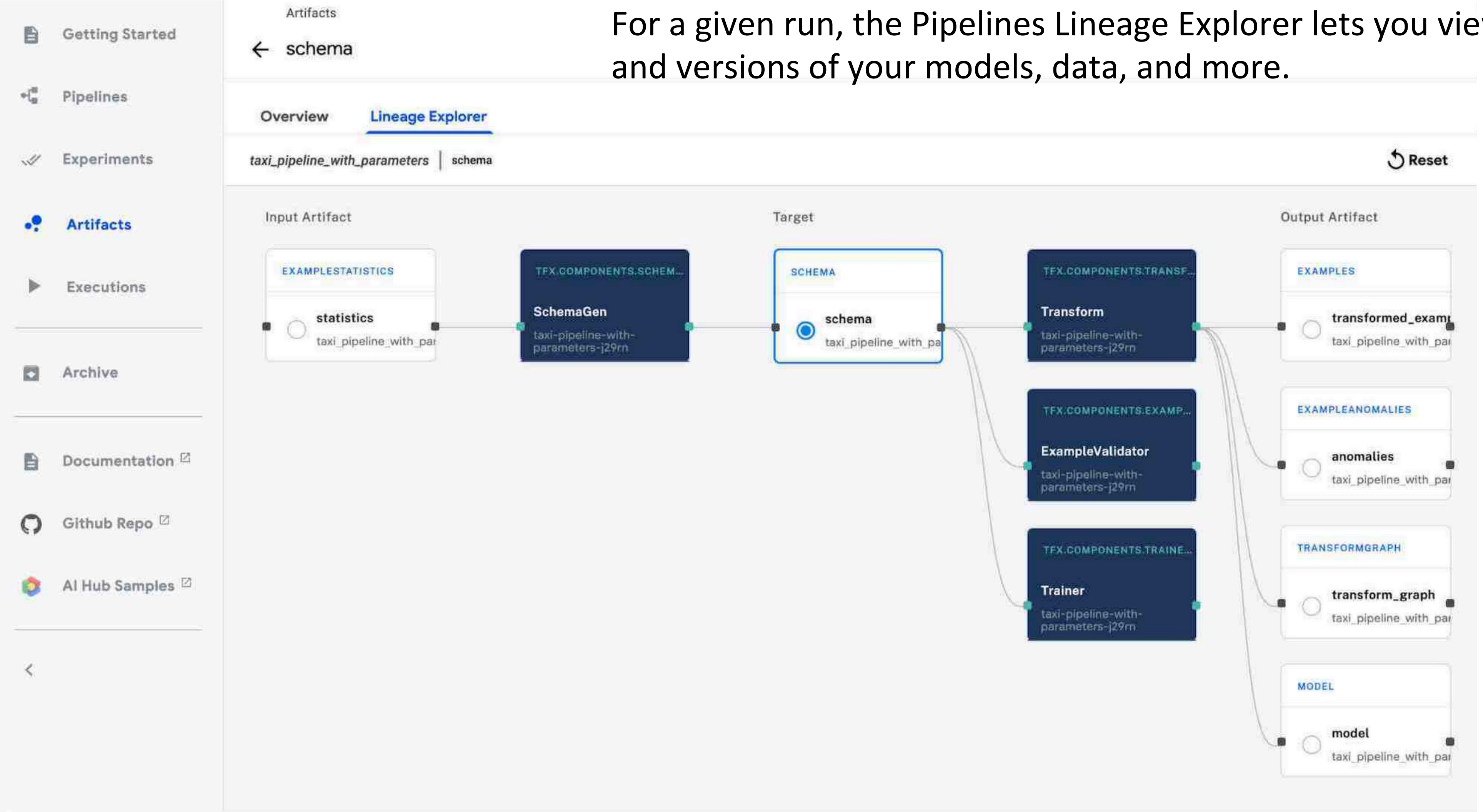
Artifact Tracking



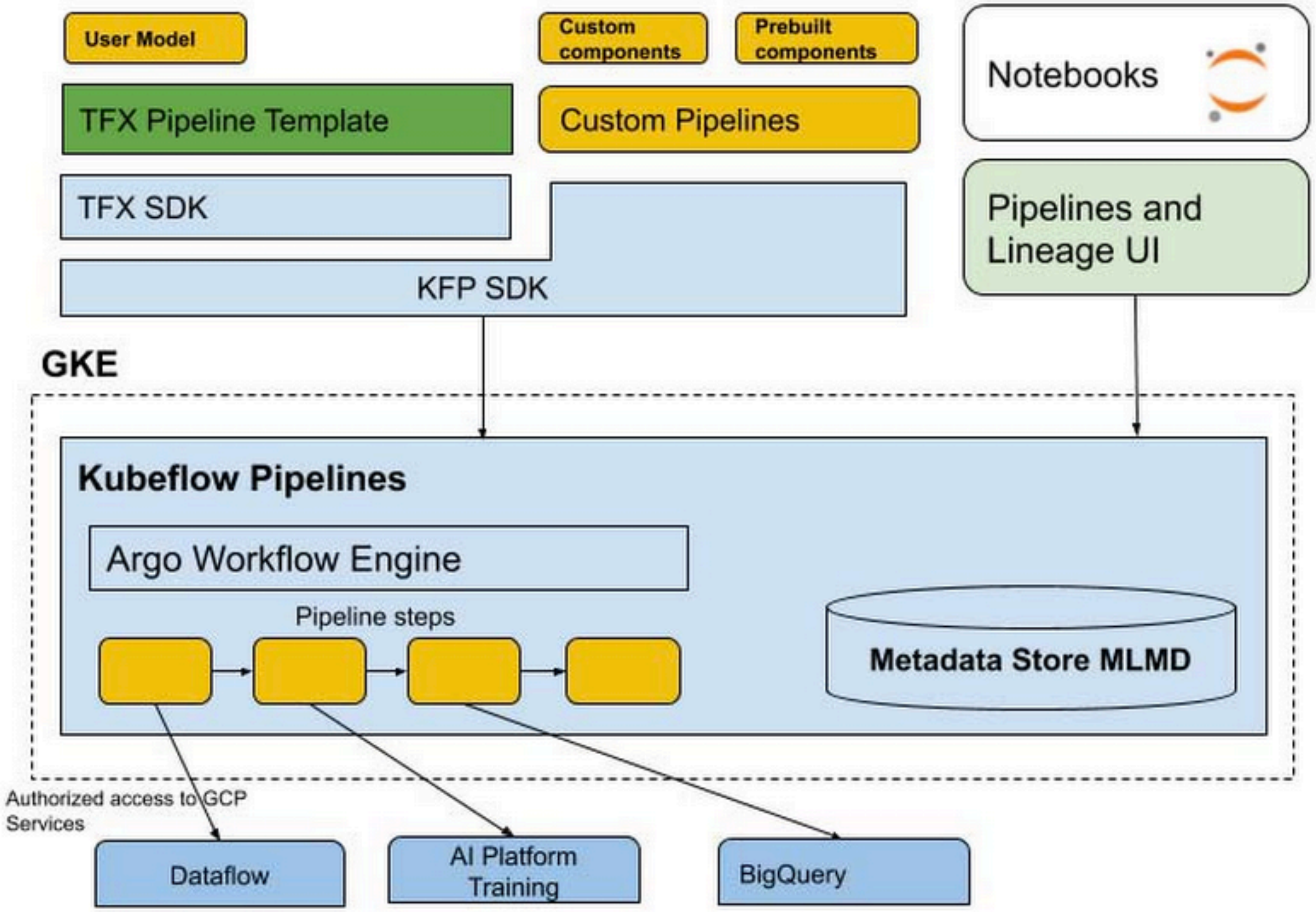
Artifacts for a run of the “TFX Taxi Trip” example pipeline. For each artifact, you can view details and get the artifact URL—in this case, for the model.

Lineage Tracking

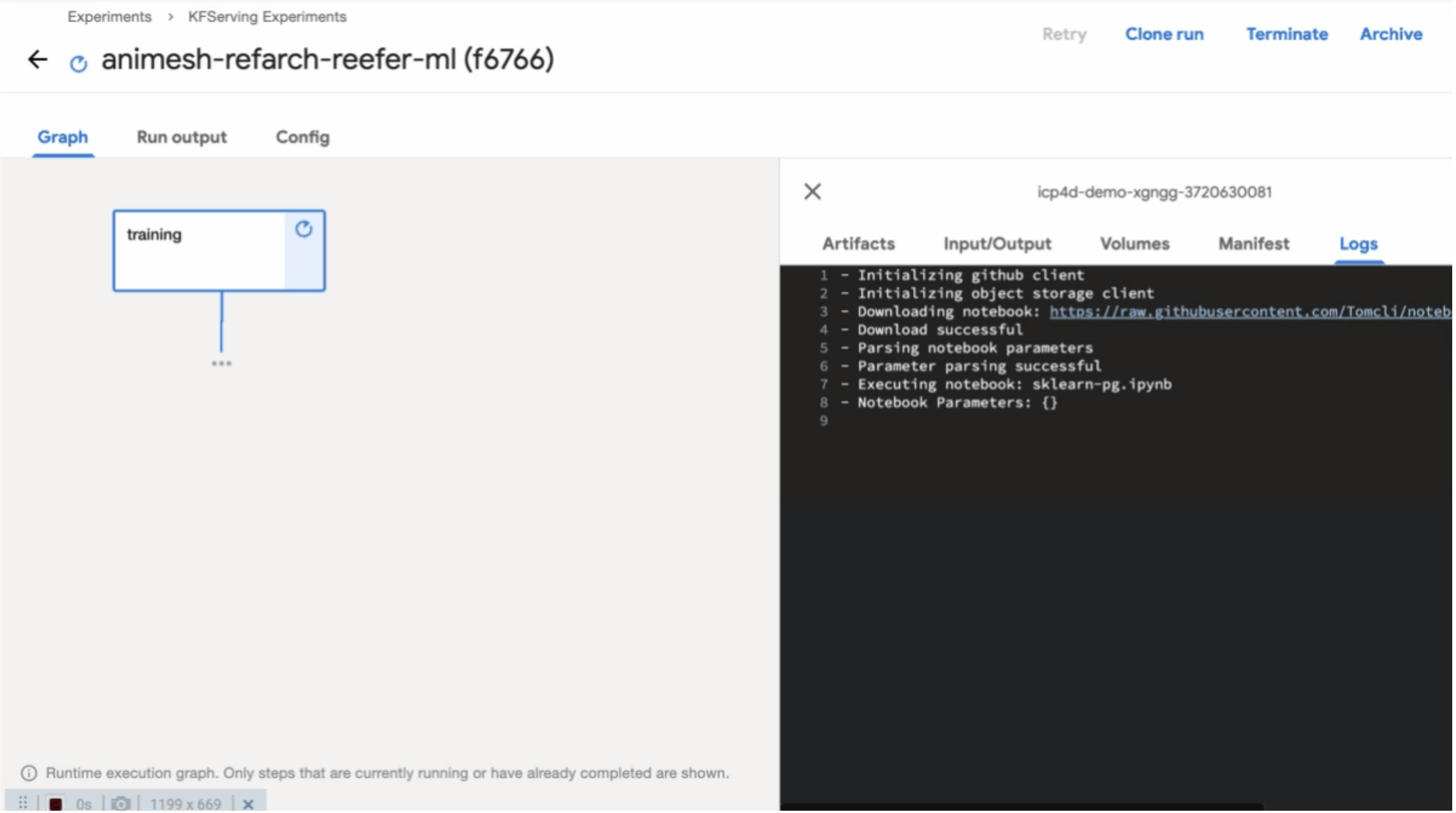
For a given run, the Pipelines Lineage Explorer lets you view the history and versions of your models, data, and more.

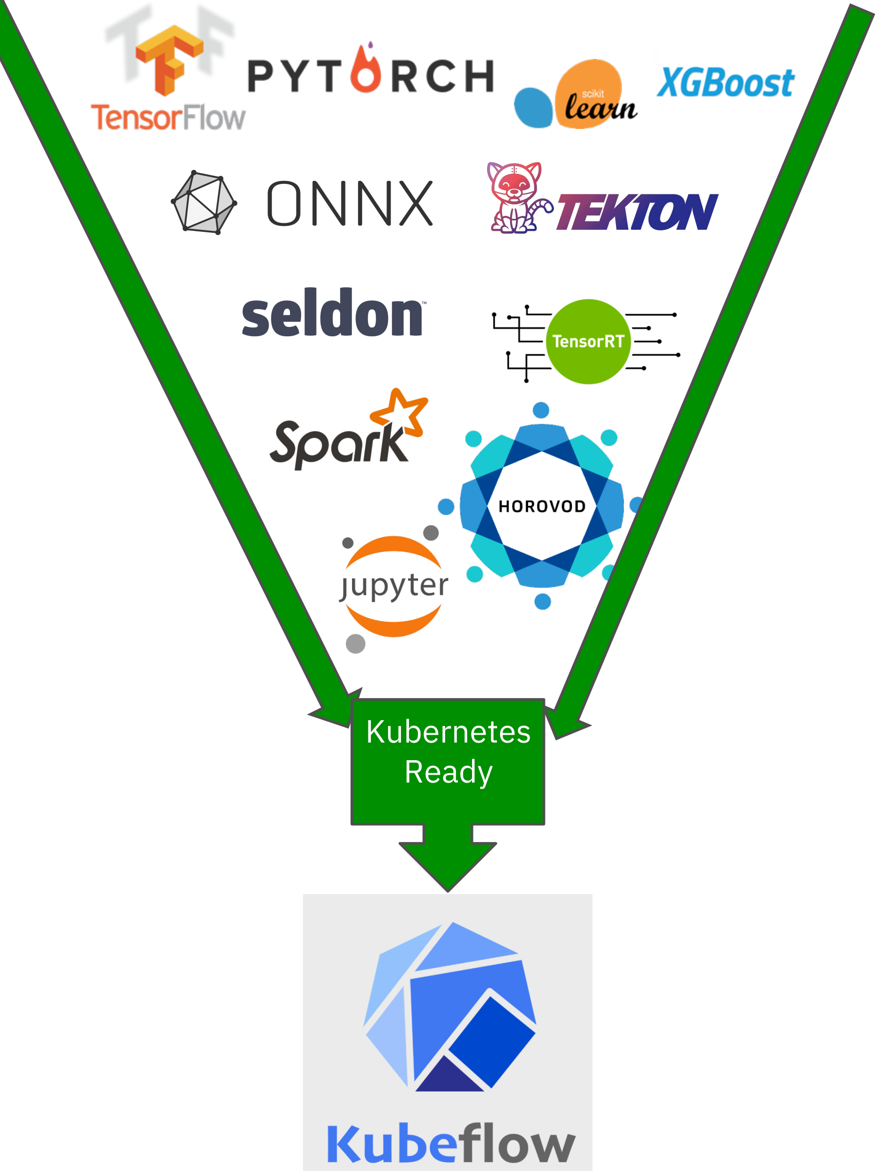


Kubeflow Pipeline Architecture



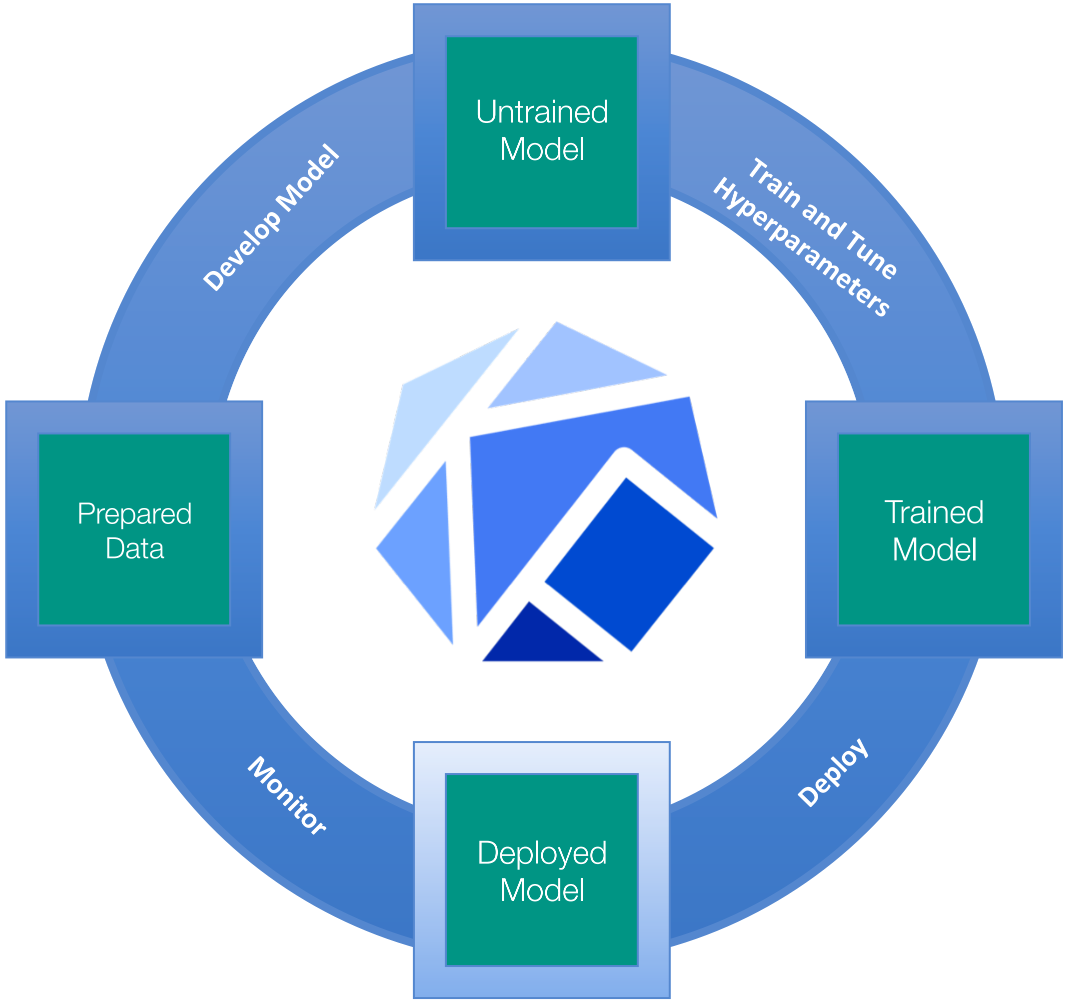
Kubeflow Pipelines can train, deploy and serve





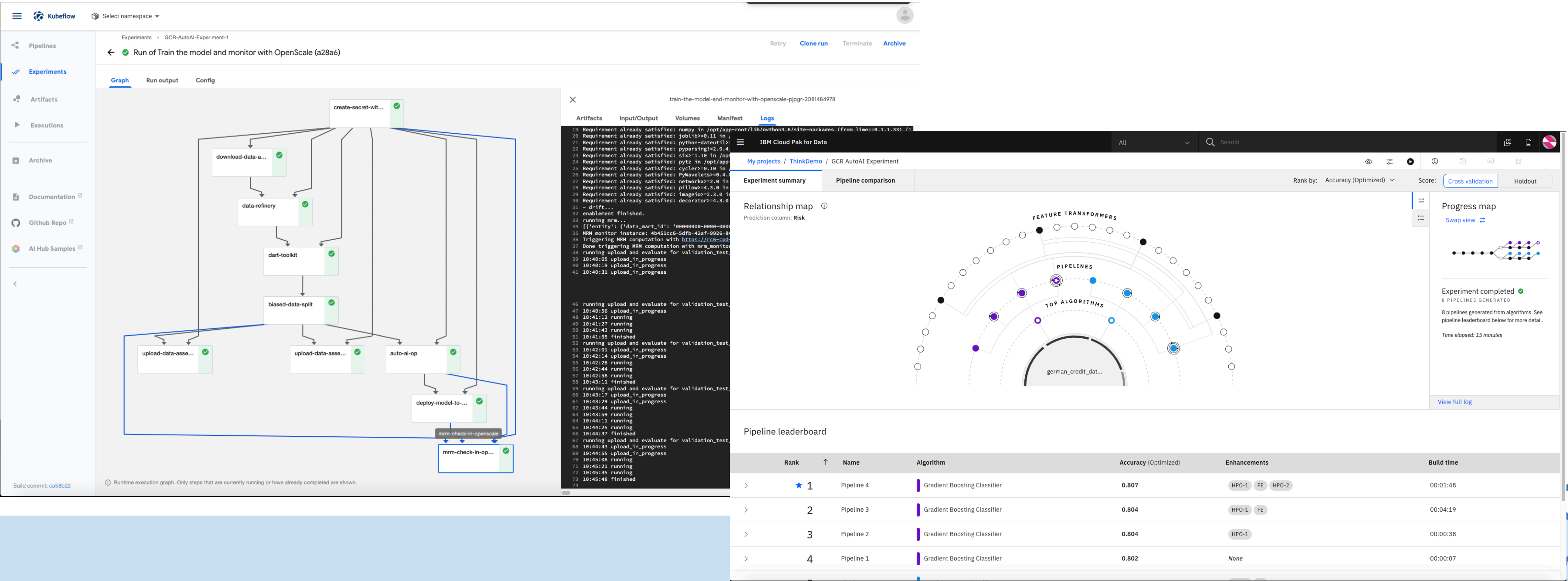
ML and AI Platform

Watson Productization of Kubeflow Pipelines

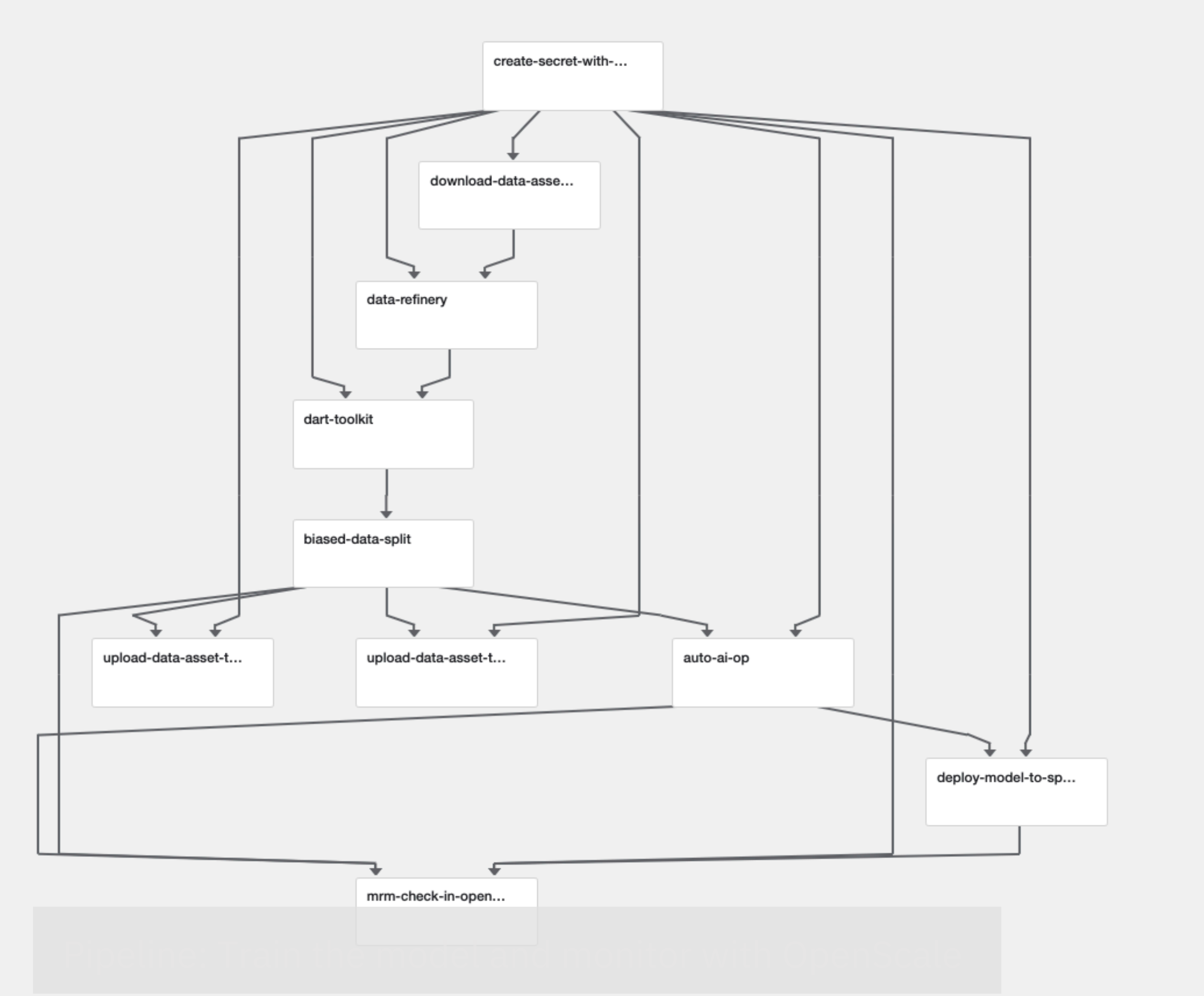


Watson AI Pipelines

- 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



Graph Source



Show summary ⓘ Static pipeline graph

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.github.ibm.com/AI-Lifecycle-Poland/kubeflow-pipelines-credentials/master/config_cpd

catalog_name

DataCatalog

asset_id

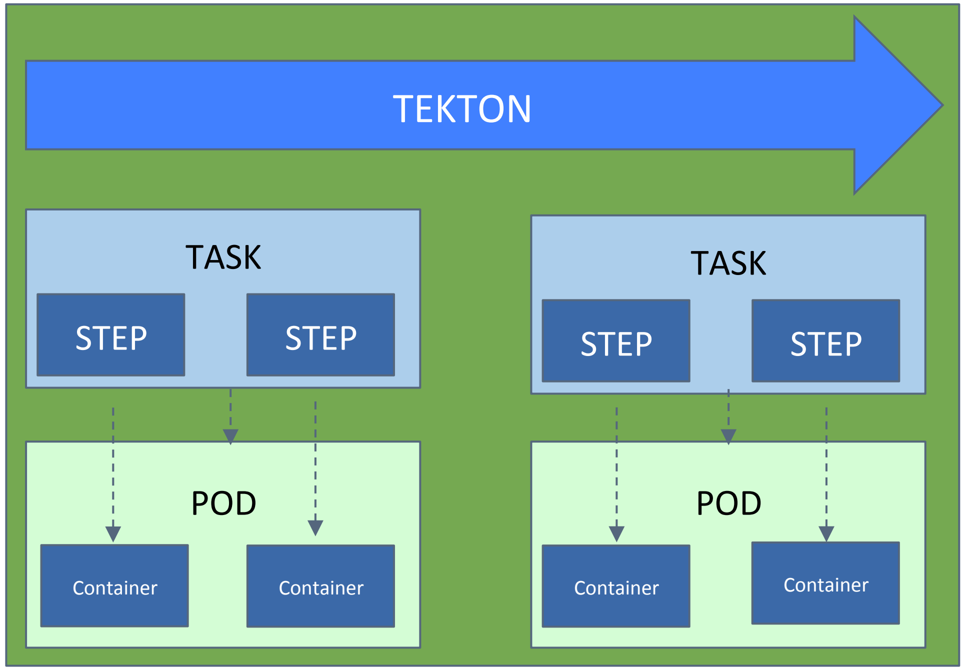
2737bafc-3f78-4e2d-850a-e7f352b3d6b8

pre_production_space_uid

1dd2aaec-781a-4712-a7ff-ae1862cf7a84

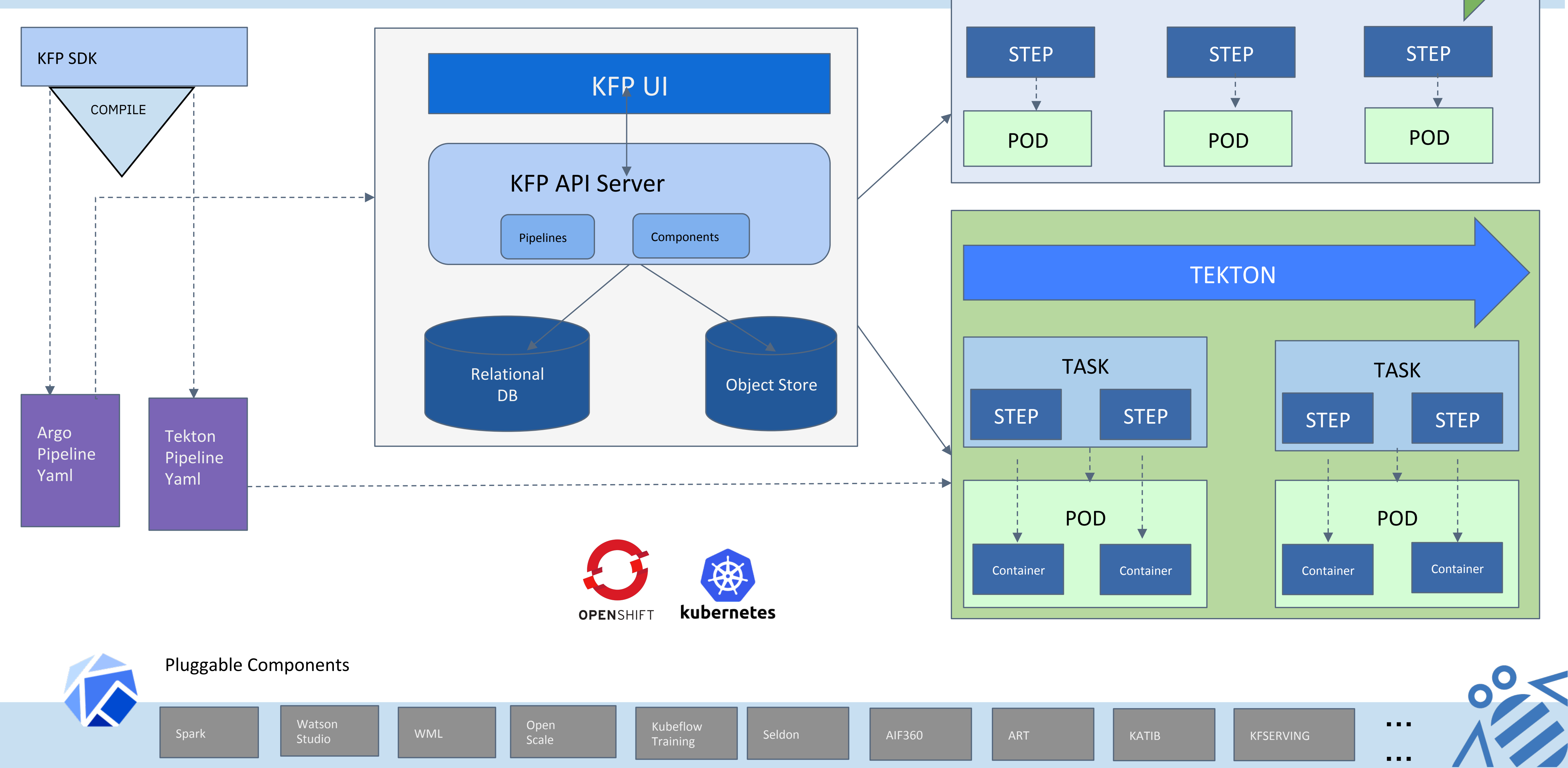
Tekton

- 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.

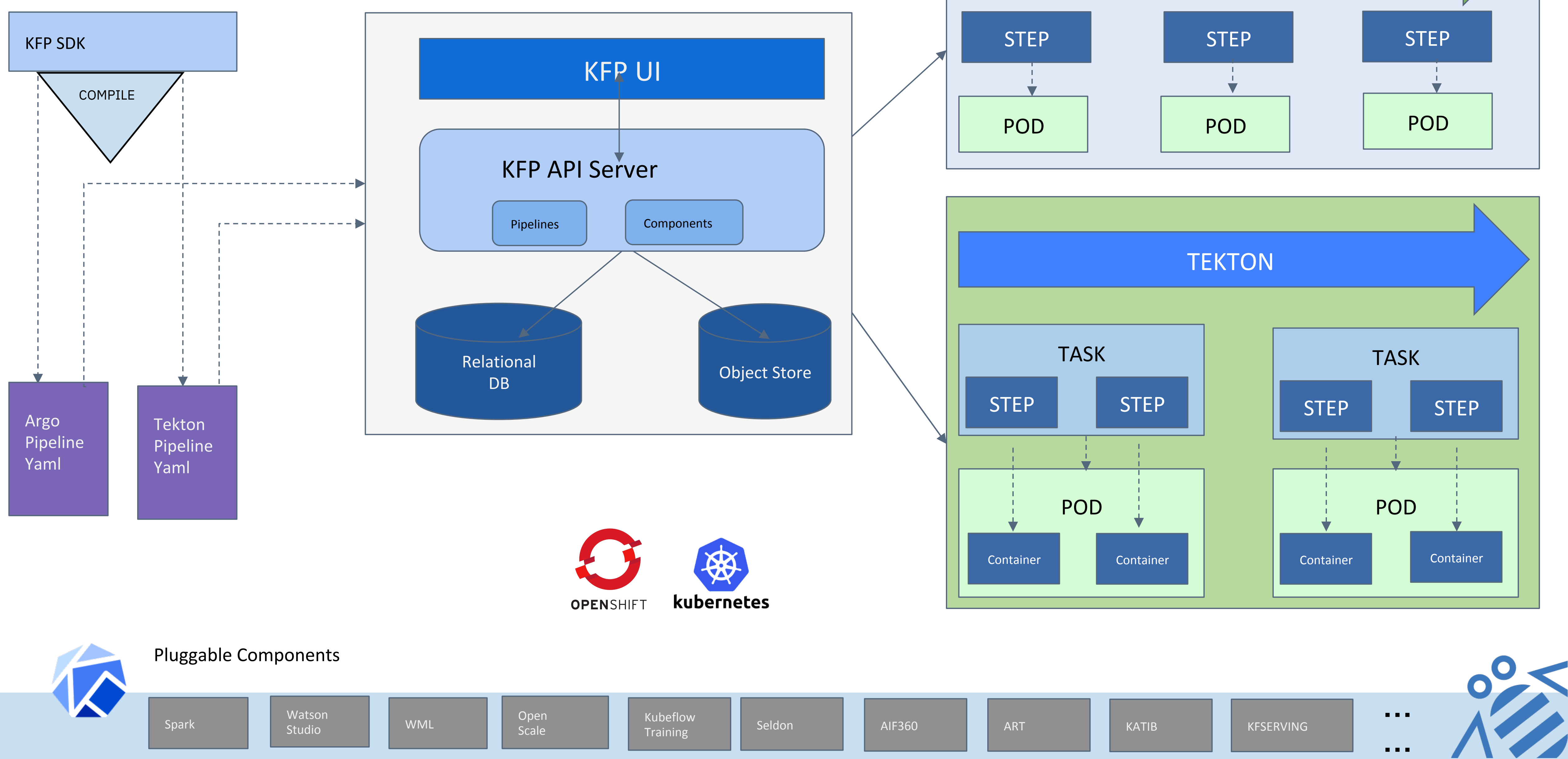


- A **PipelineResource** defines an object that is an input (such as a git repository) or an output (such as a docker image) of the pipeline.
- A **PipelineRun** defines an execution of a pipeline. It references the Pipeline to run and the PipelineResources to use as inputs and outputs.
- A **Pipeline** 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



KFP – Tekton Phase Two



KFP – Tekton Challenges

Multiple Moving parts, with different stakeholders

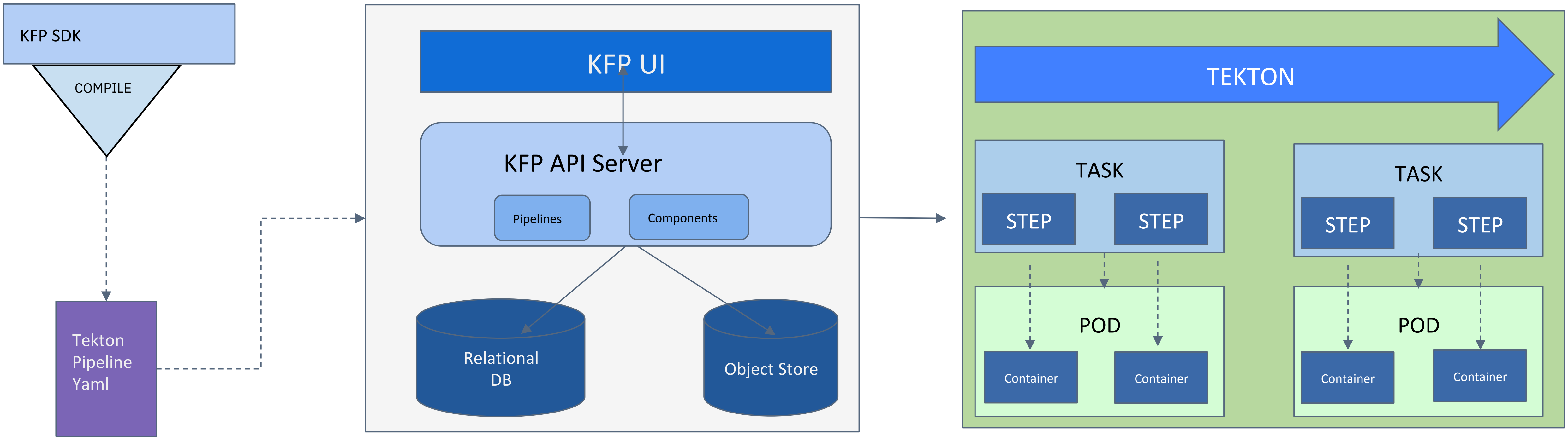
Tekton Community: Argo with version 2.6 much more mature than Tekton v0.11 (alpha) when the work started around 5 months ago

- 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

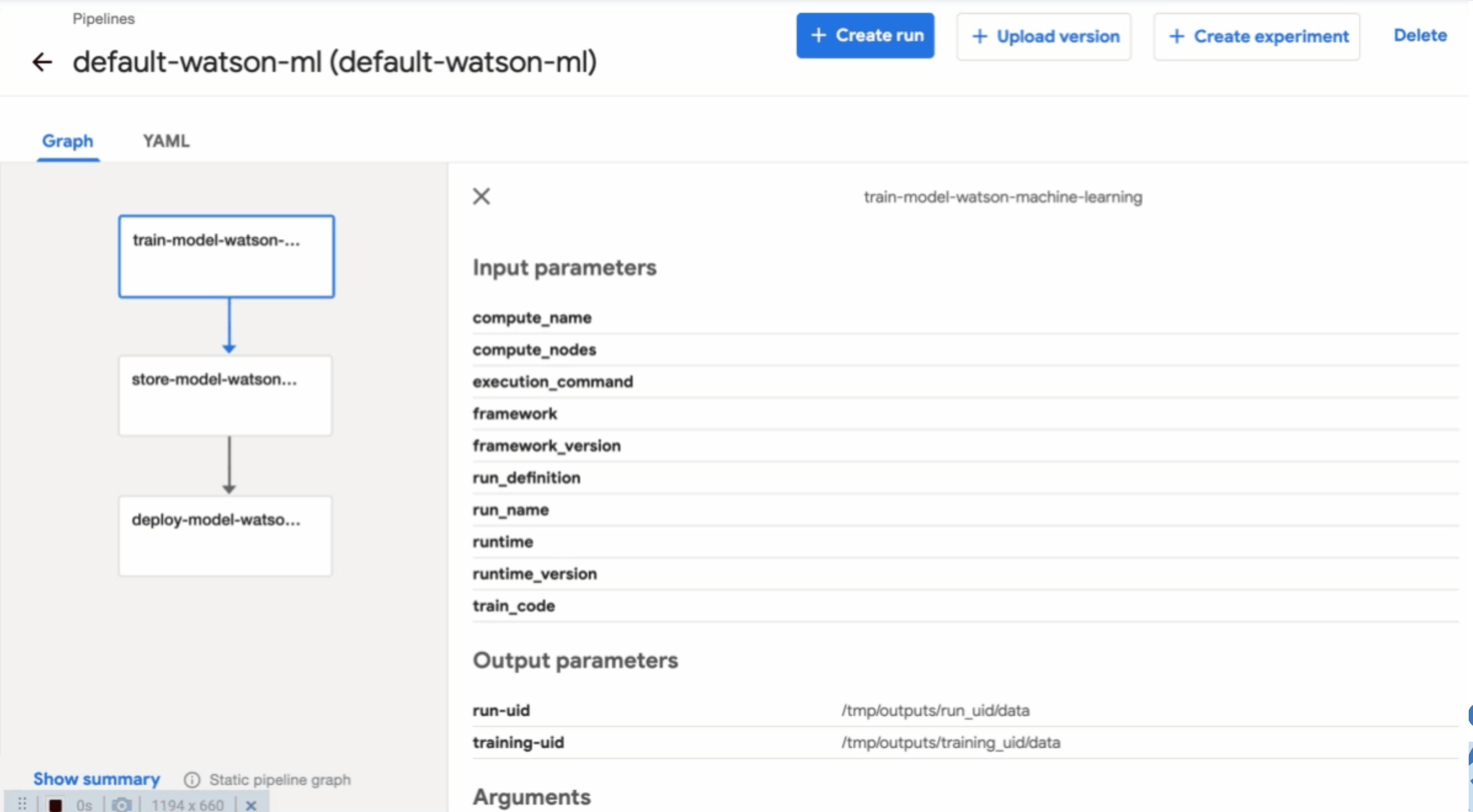
Kubeflow Pipeline and TFX Community: Open source team needed to be formed for the specific mission. And trained. Additionally Google needed to be brought up on the same page, and convinced the validity of integration.

- 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

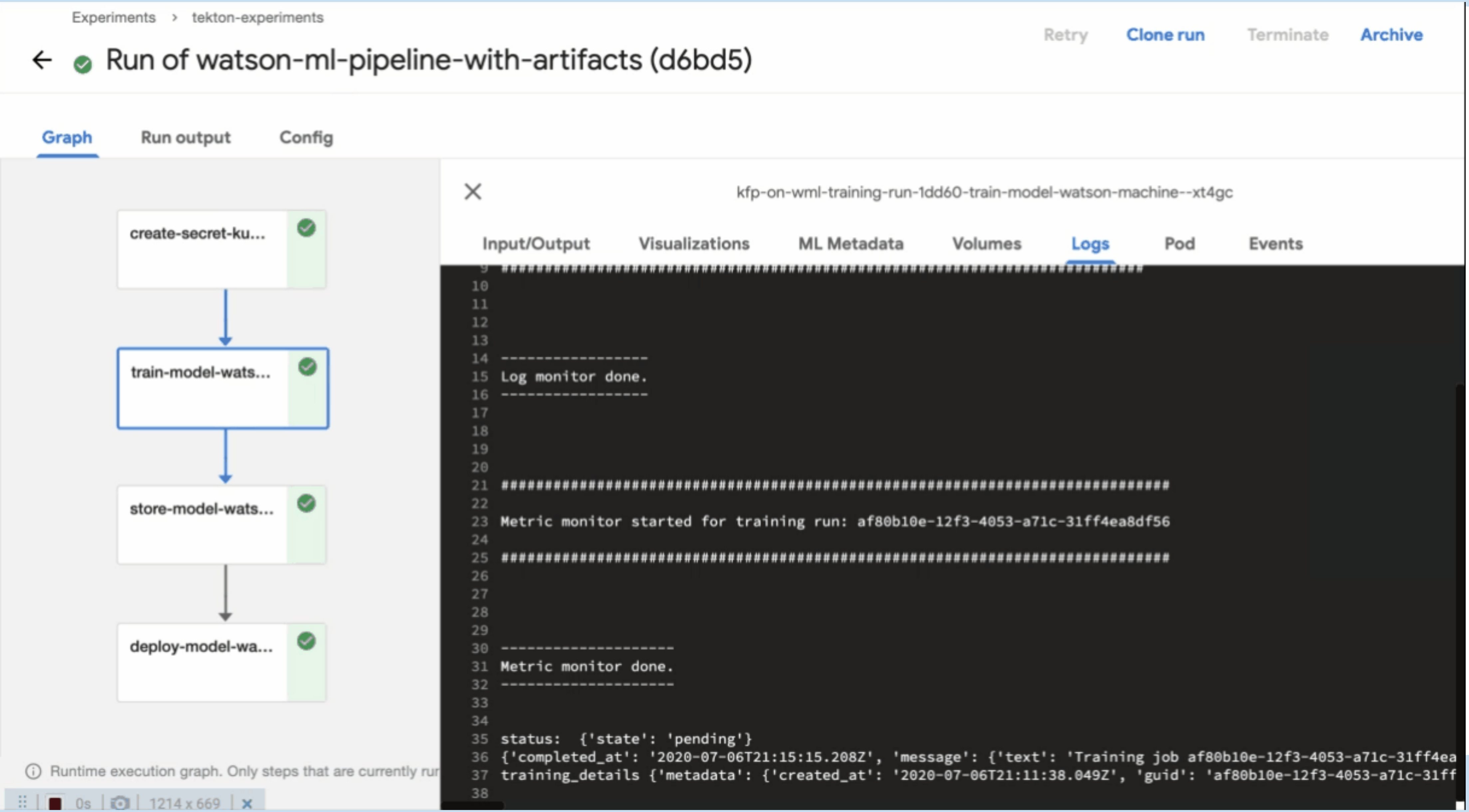
KFP – Tekton: Delivered



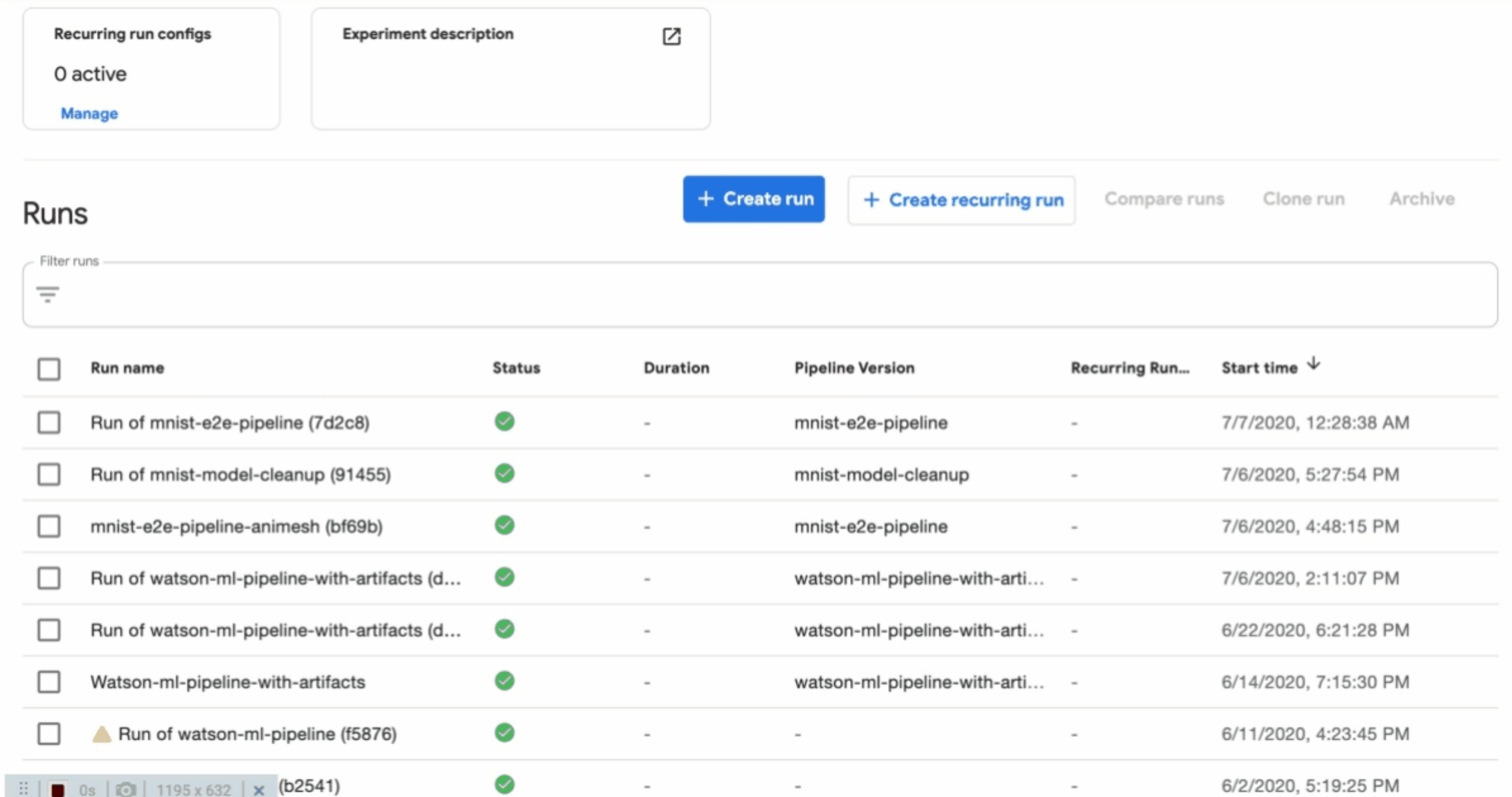
Same KFP Experience: DAG, backed by Tekton YAML



Same KFP Exp: Logs, Lineage Tracking and Artifact Tracking



End to end Kubeflow Components : With KFP-Tekton


Recurring run configs
0 active
Manage

Experiment description

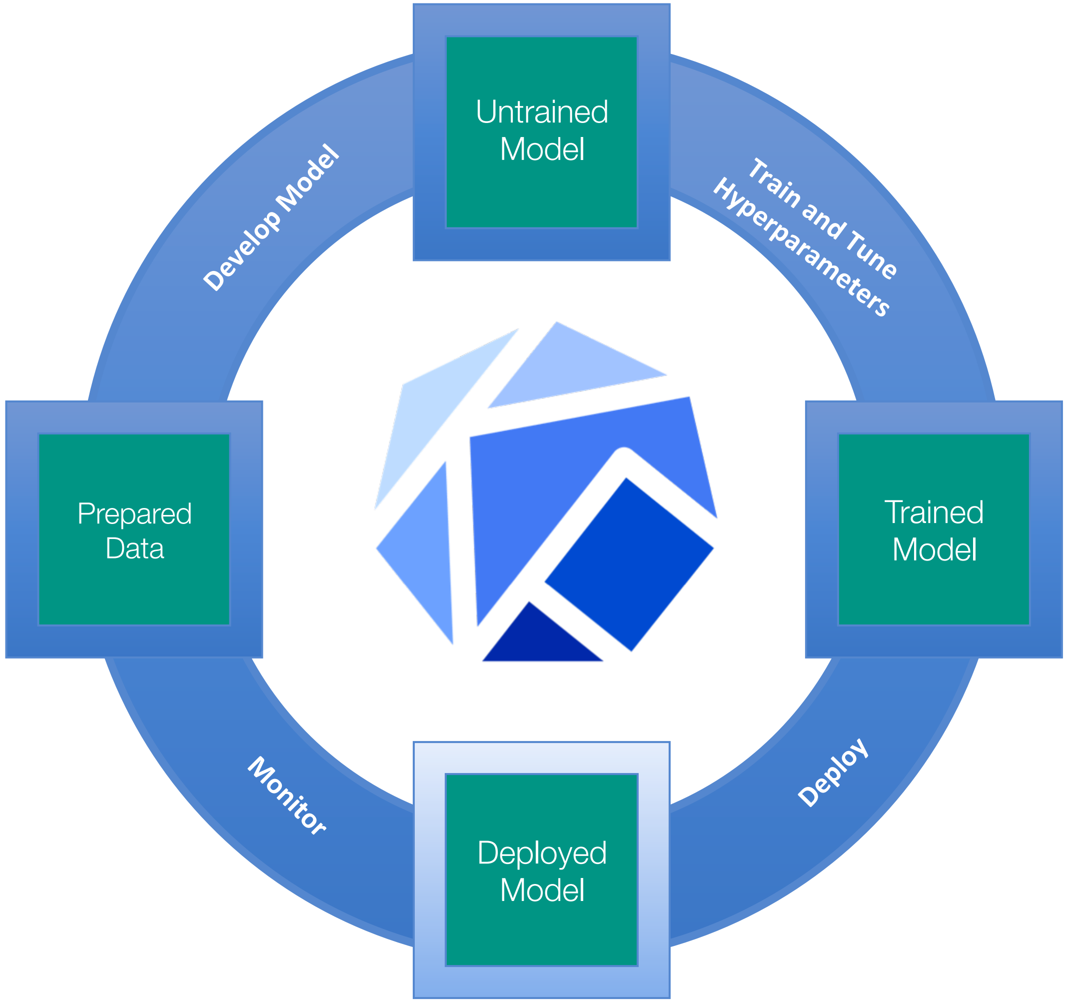
Runs

+ Create run
+ Create recurring run
Compare runs
Clone run
Archive

Filter runs

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
(b2541)	✓	-	-	-	6/2/2020, 5:19:25 PM

Kubeflow Adoption: External and Internal

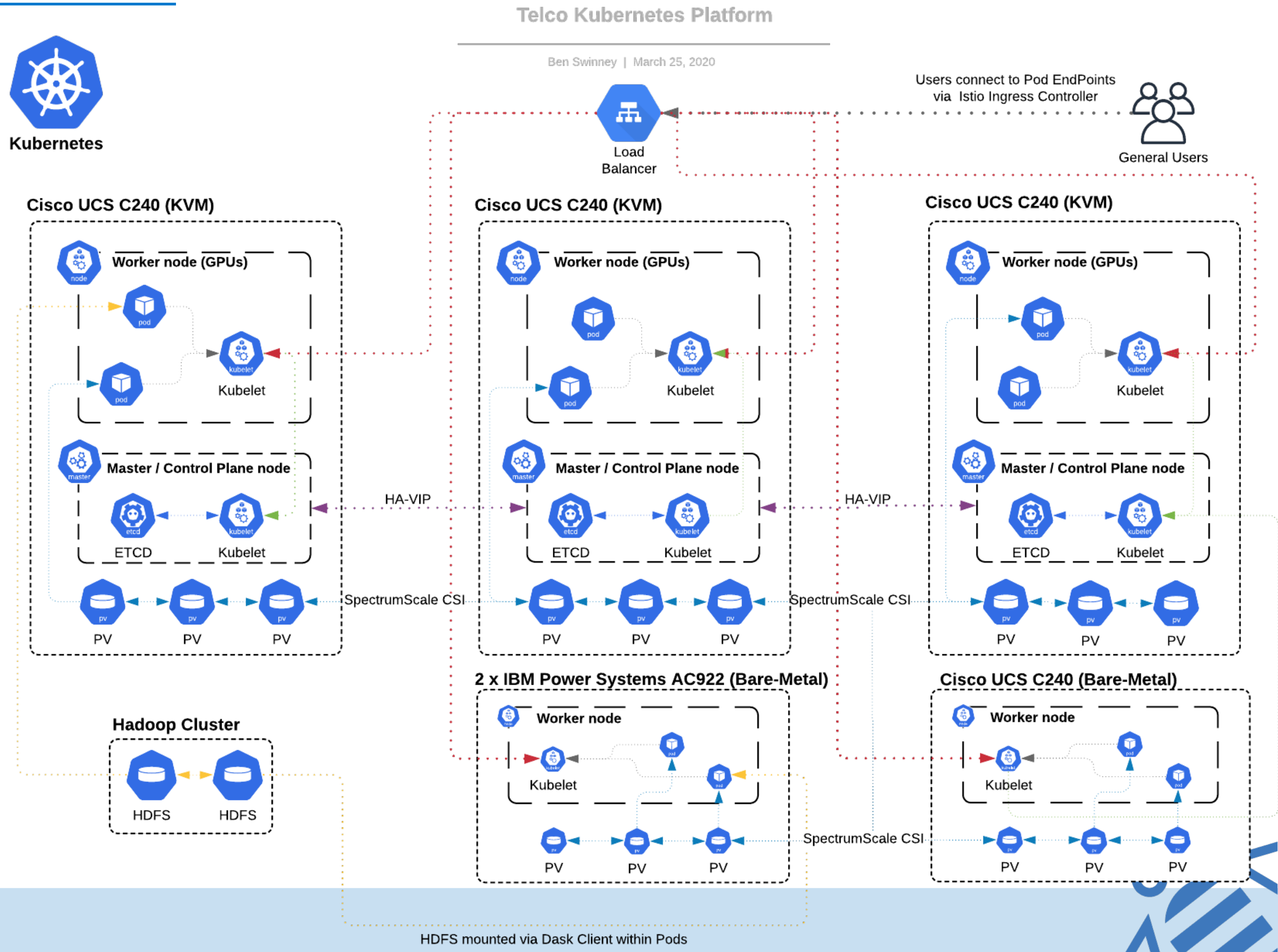


Telstra: Collaborating with IBM to build an Open Source based OneAnalytics Platform leveraging Kubeflow

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

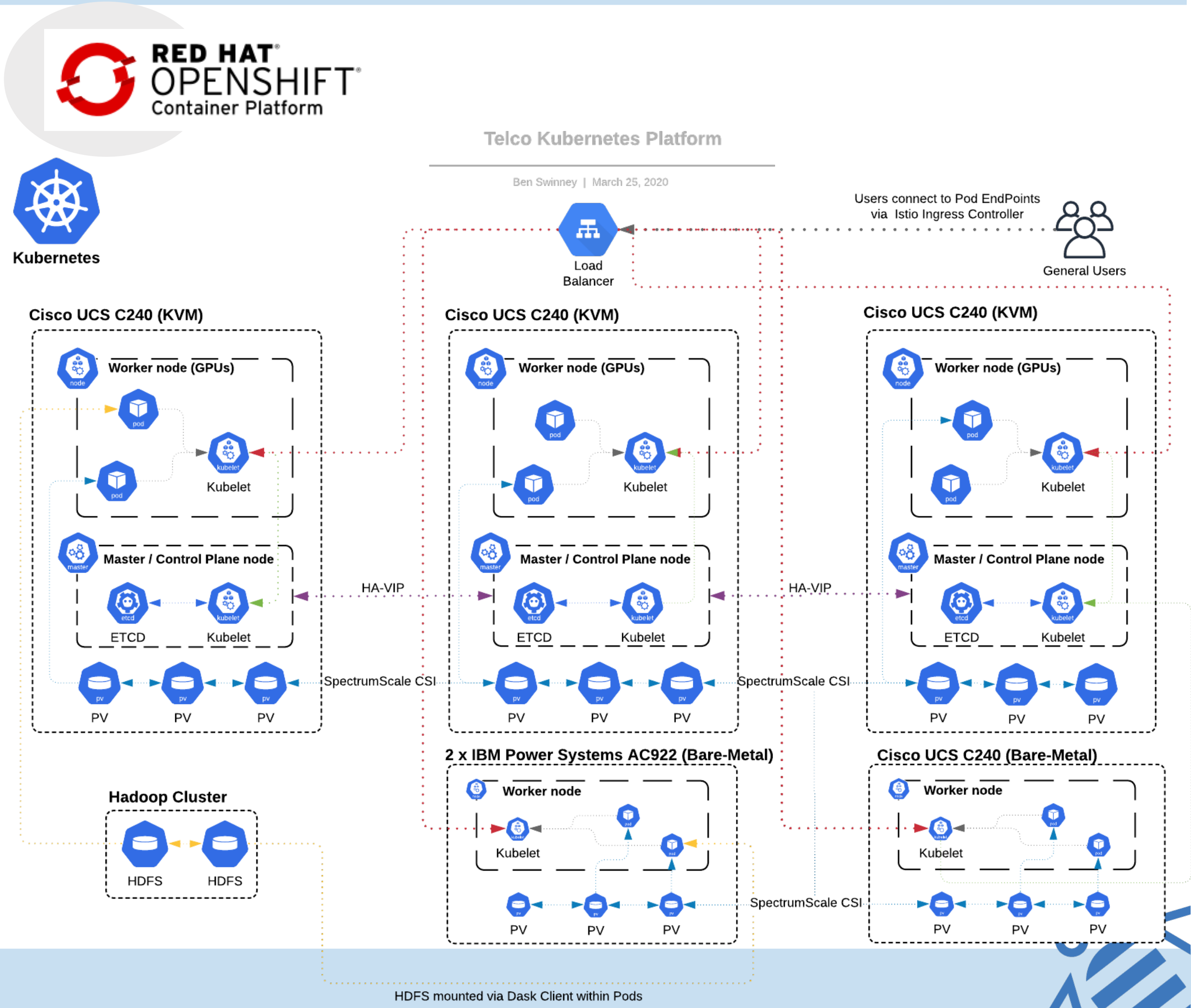
Telstra AI Lab - (TAIL) - Configuration

- Kubernetes – 1.15
- Spectrum Scale CSI Driver
- MetalLB 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)



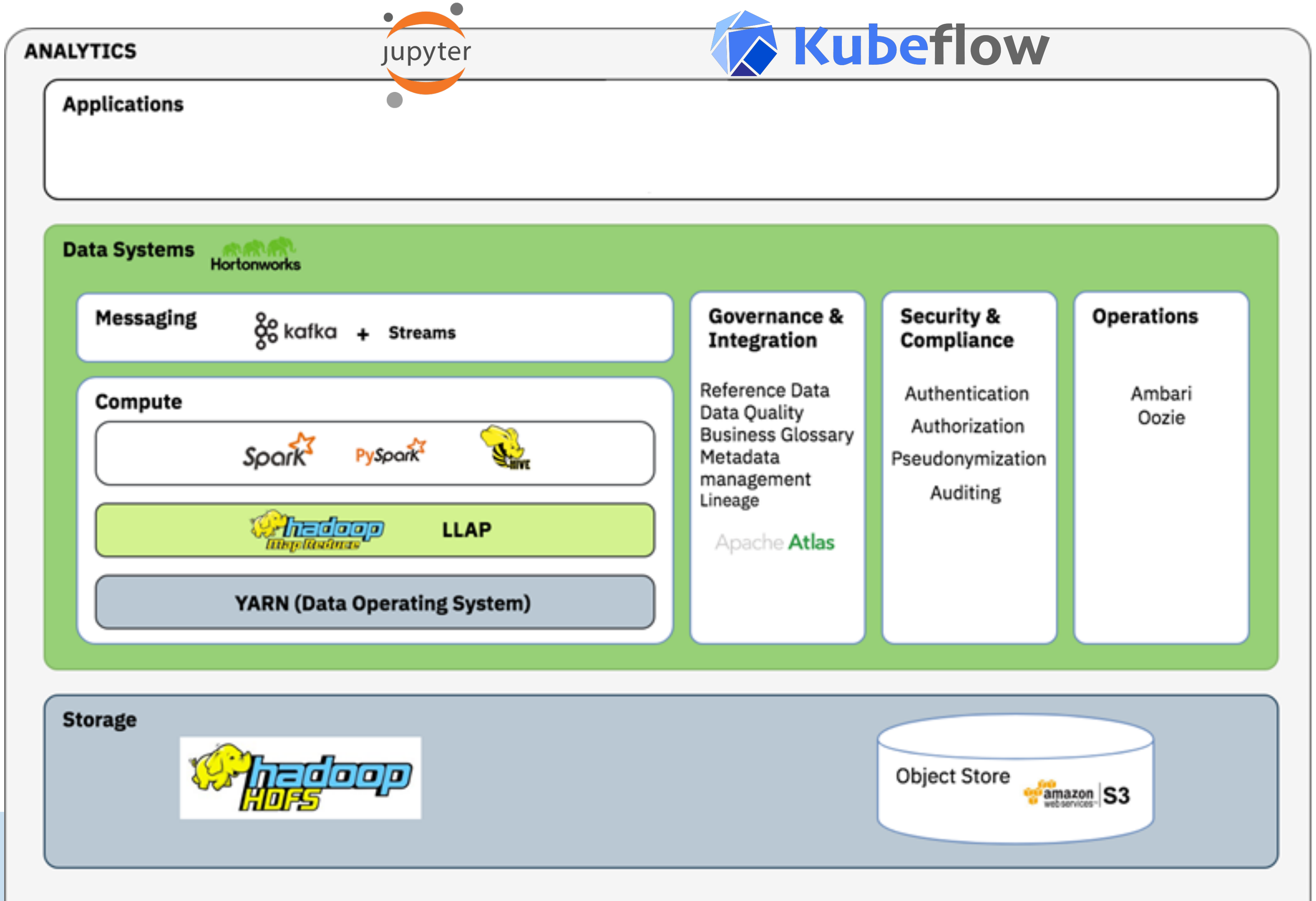
Telstra AI Lab - (TAIL) – Future state

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

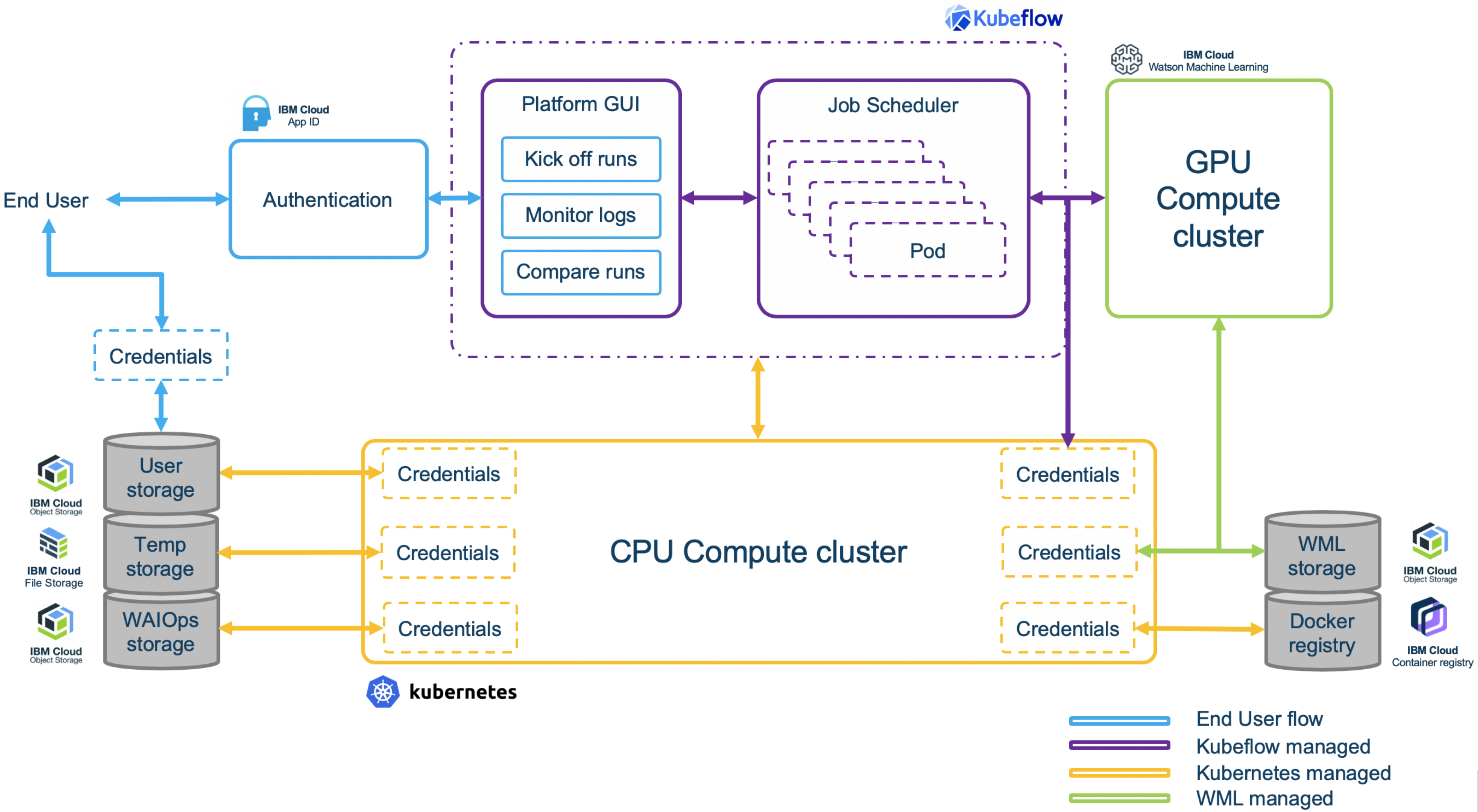


Yara – Working with IBM to build a Data Science Platform for Digital Farming ML use cases based on Kubeflow

THINK 2020 Session: Enable Smart Farming using Kubeflow
https://www.ibm.com/events/think/watch/replay/126494864

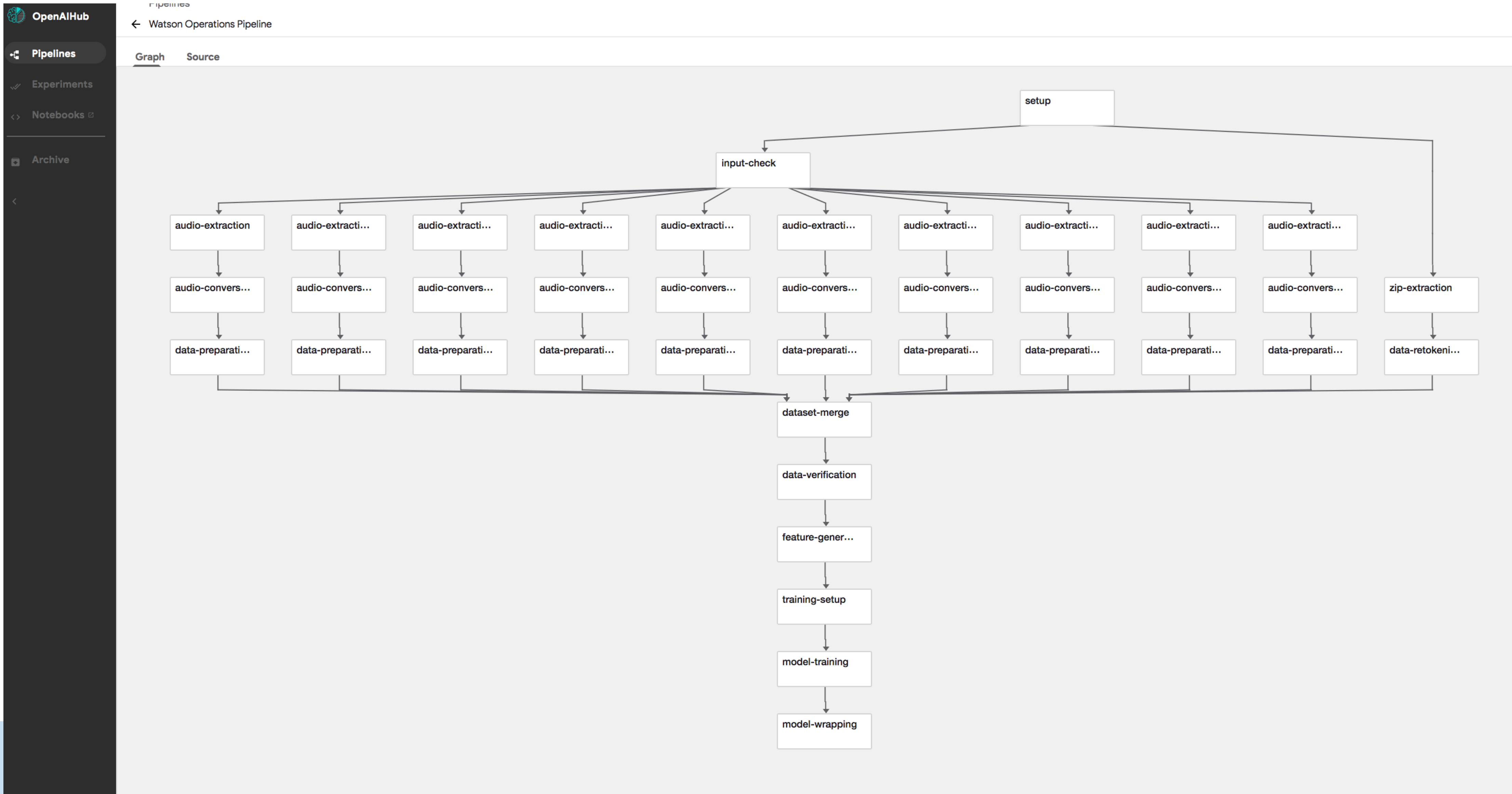


Watson STT: Kubeflow Pipelines running Operations

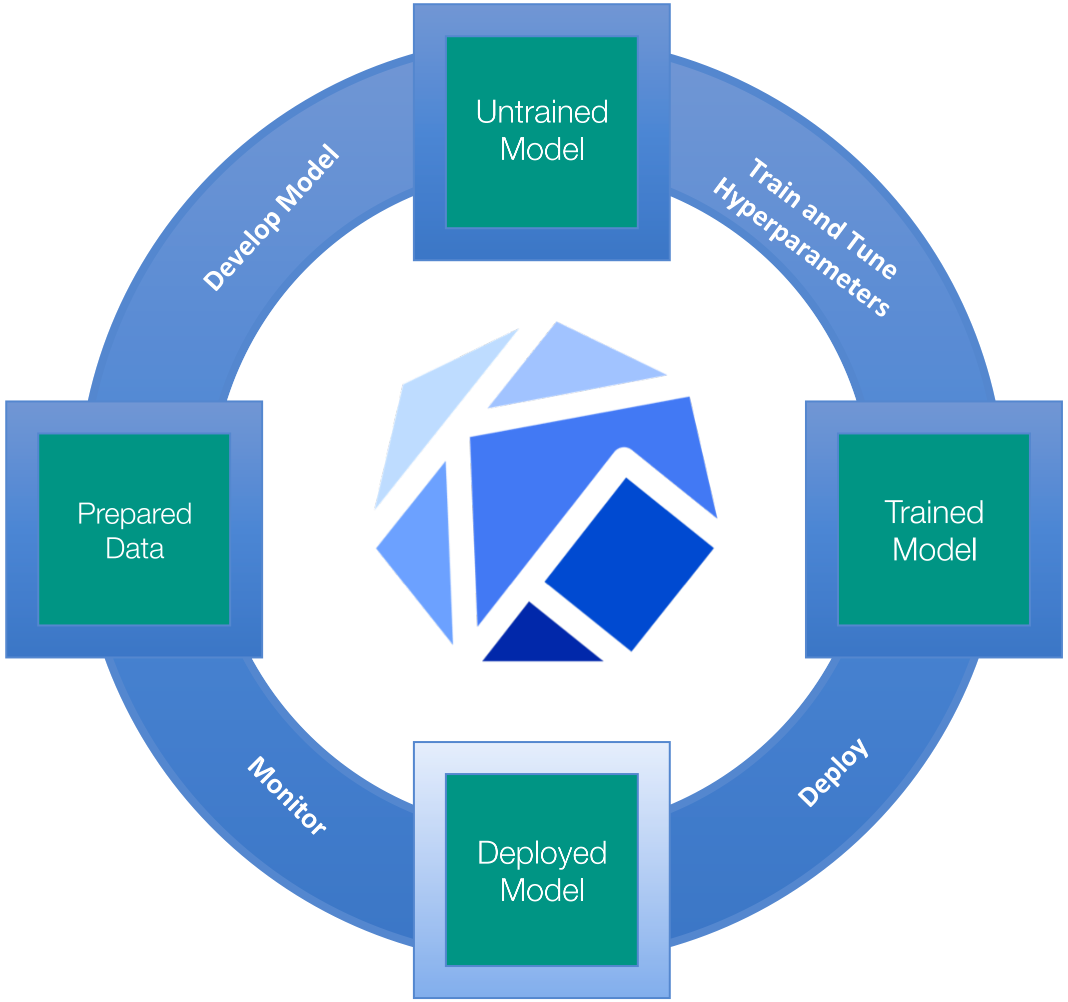


Watson SpeechToText training Kubeflow pipeline

https://stt-payload-kubeflow.us-east.containers.appdomain.cloud/_/pipeline-dashboard



OpenDataHub



Upstream, Midstream and Downstream

'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

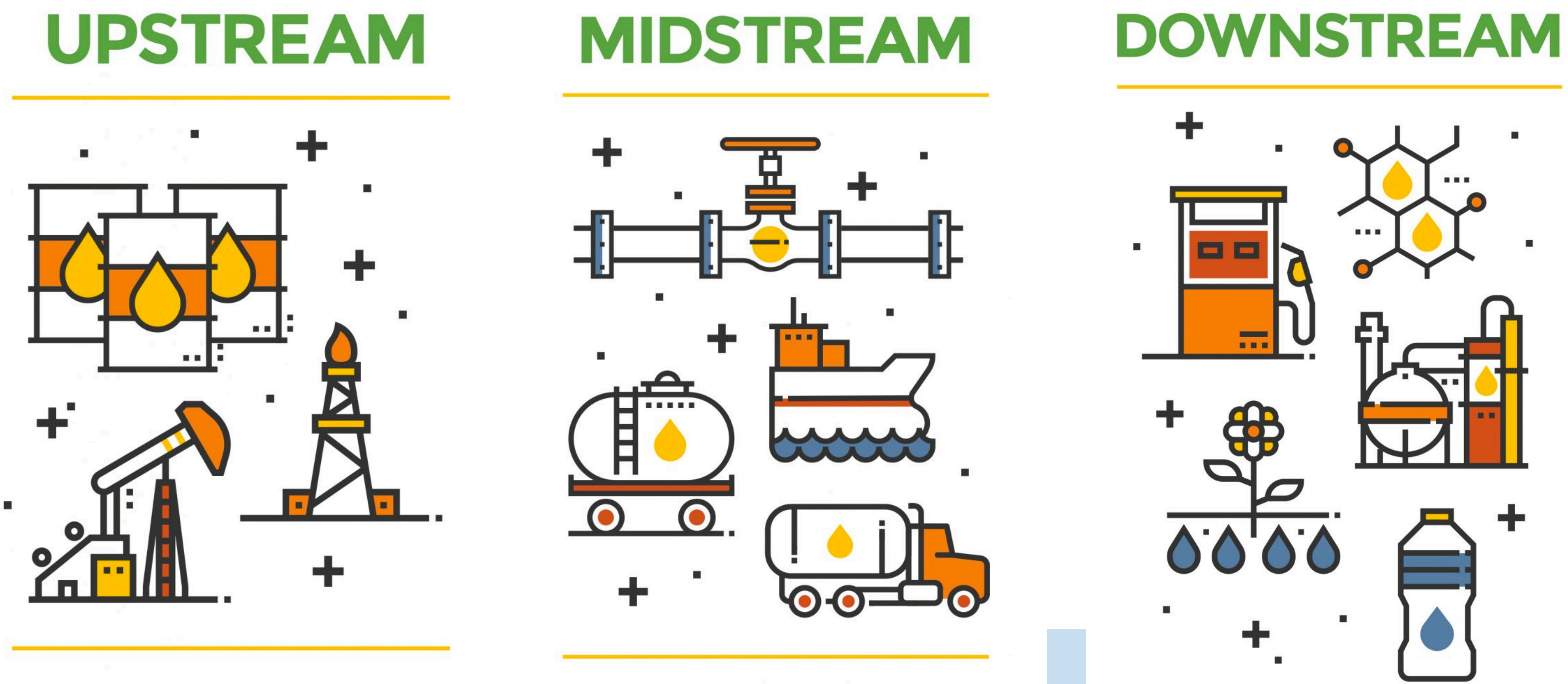
When a well is producing, oil or gas enters the midstream juncture.The middle part of the process requires multiple cooperation.

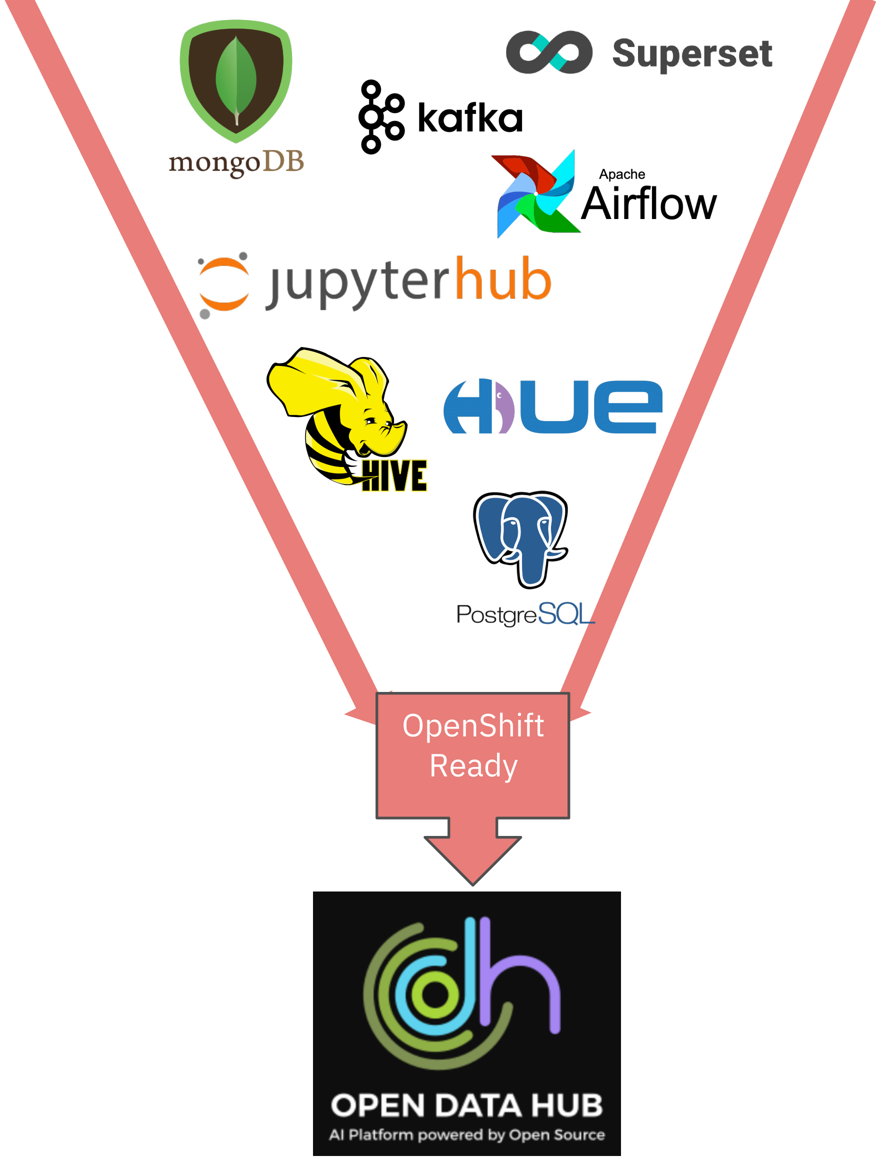
Downstream

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

Upstream, Midstream and Downstream

'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.

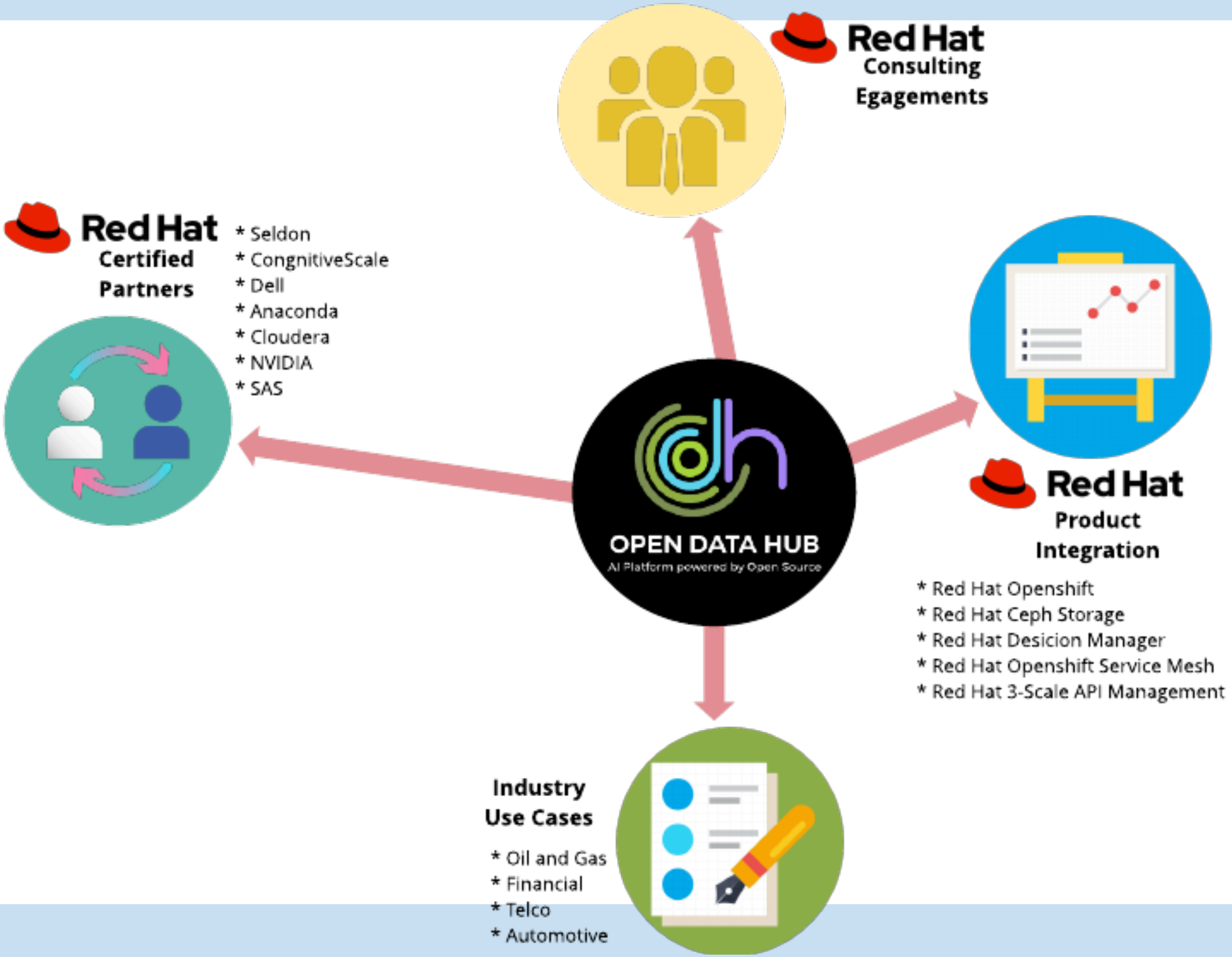




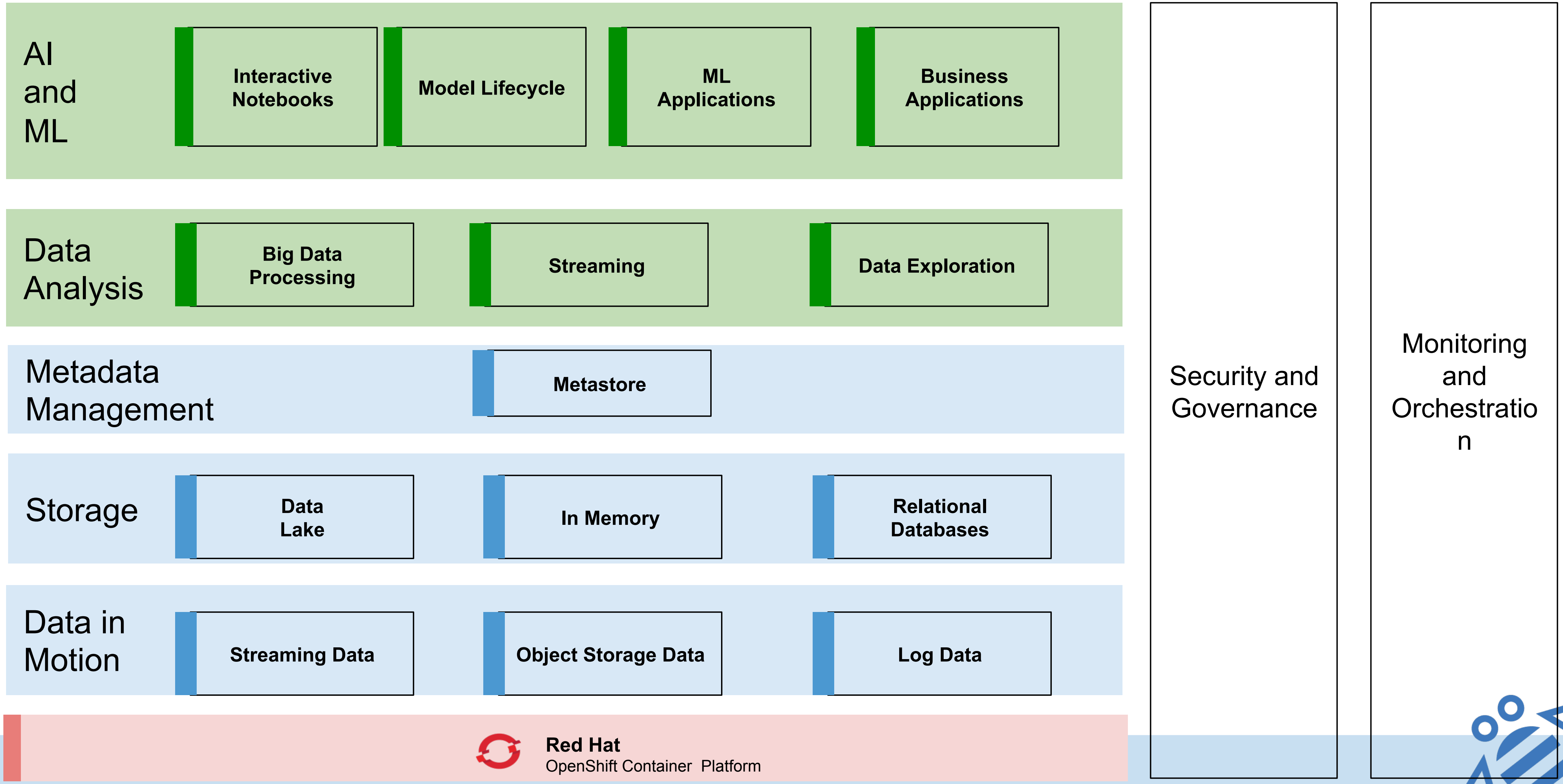
Data Platform

Operator Hub - operatorhub.io

OPEN DATA HUB - Ecosystem



OPEN DATA HUB
REFERENCE ARCHITECTURE



IBM

OPEN DATA HUB
REFERENCE IMPLEMENTATION

Kubeflow

AI and ML
- **Interactive Notebooks**: JupyterHub, Hue
- **Model Lifecycle**: Kubeflow, Seldon, MLFlow
- **ML Applications**: OpenDataHub AI Library
- **Business Applications**: Superset

Data Analysis
- **Big Data Processing**: Spark, SparkSQL, Thrift
- **Streaming**: Kafka Streams, Elastic Search
- **Data Exploration**: Hue, Kibana

Metadata Management
- **Metastore**: Hive

Storage
- **Data Lake**: RedHat Ceph Storage
- **In Memory**: RedHat Data Grid (Infinispan)
- **Relational Databases**: PostgreSQL, MySQL

Data in Motion
- **Streaming Data**: RedHat AMQ Streams, Kafka Connect
- **Object Storage Data**: RedHat Ceph S3 API
- **Log Data**: FluentD, Logstash

Red Hat
OpenShift Container Platform

Security and Governance
OpenShift Oauth

OpenShift Single SignOn (Keycloak)

RedHat Ceph Object Gateway

RedHat 3scale

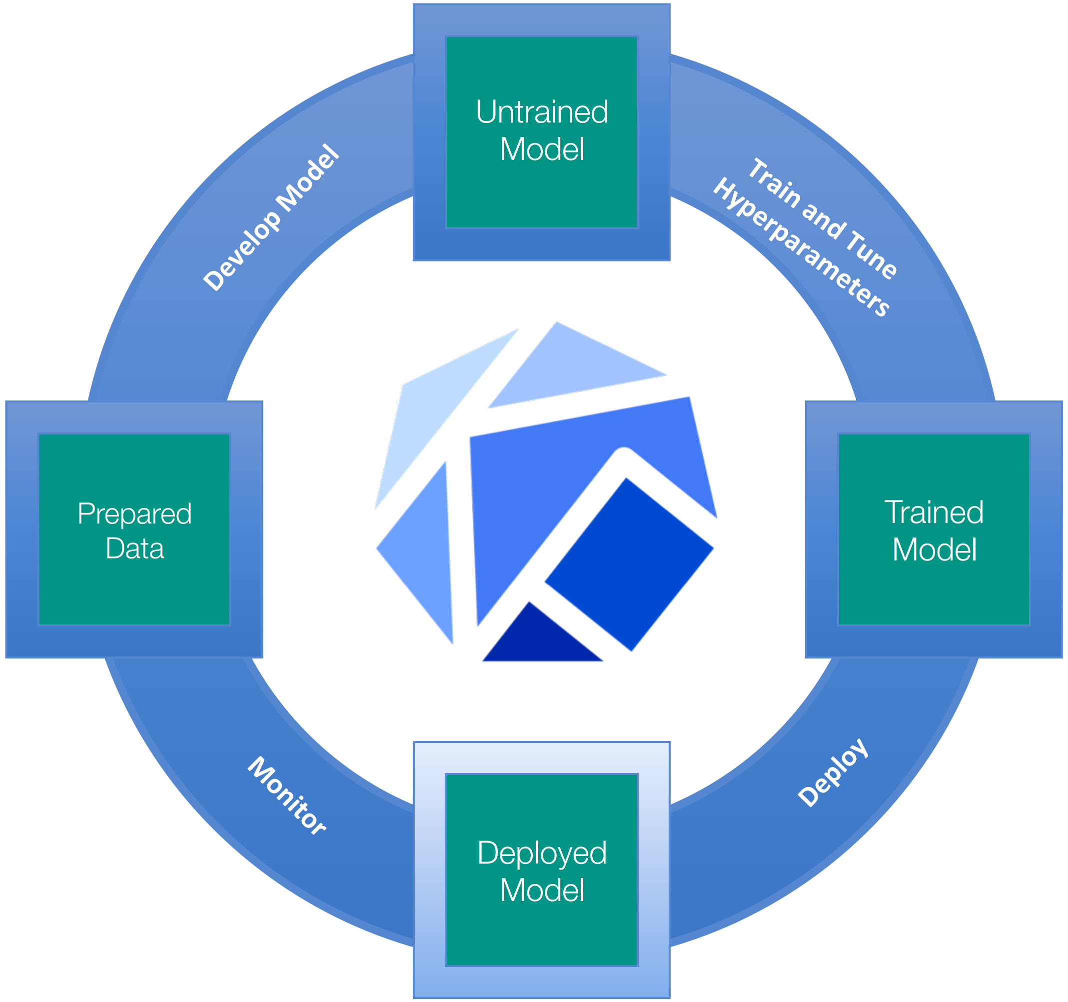
Monitoring and Orchestration
Prometheus

Grafana

Kubeflow Pipelines

Jenkins CI/CD

OpenDataHub and Kubeflow: Relationship



Initial Goals: OpenDataHub and Kubeflow

Initial Goals:

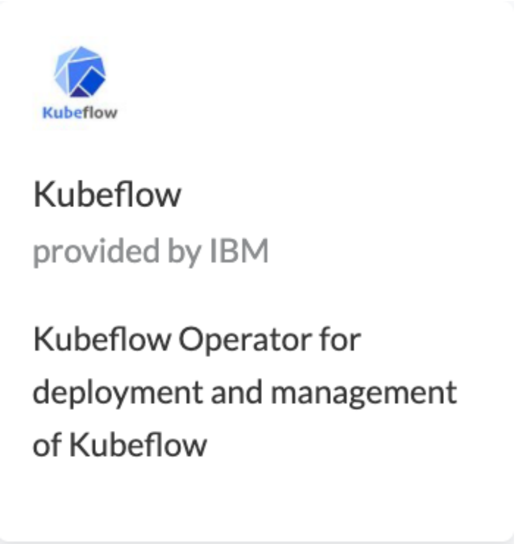
- Kubeflow has a great traction, Make it available for OpenShift users

  Done in https://github.com/opendatahub-io/manifests

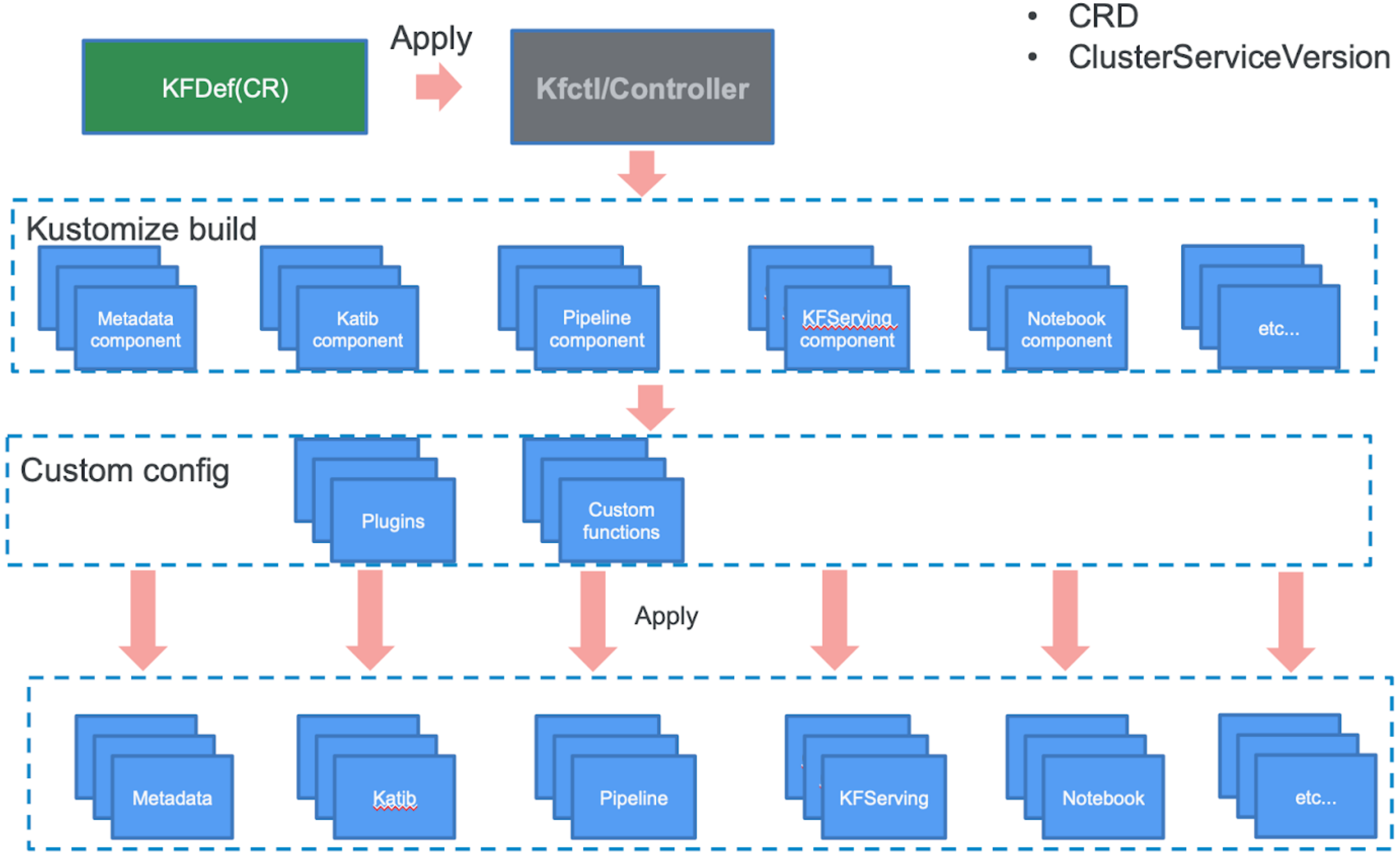
- Offer ODH users components installed by KF
- And offer components from ODH (Kafka, Apache SuperSet, Hive…) to KF community
- Decide if we can leverage KF project and community as upstream for ODH
- Think Kubernetes -> OpenShift
- Frees up ODH maintainers time to make sure KF keeps running well on OpenShift

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



Controller deployment files
- CRD
- ClusterServiceVersion

Outcome: Kubeflow an Upstream for OpenDataHub

Kubeflow

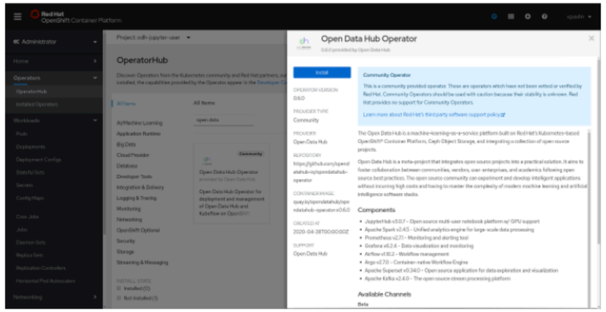
- A version of the Operator based on Kubeflow Architecture released: https://developers.redhat.com/blog/2020/05/07/open-data-hub-0-6-brings-component-updates-and-kubeflow-architecture/?sc_cid=7013a000002DTqEAAW
- Most of the components converted: https://github.com/opendatahub-io/odh-manifests
- Still 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

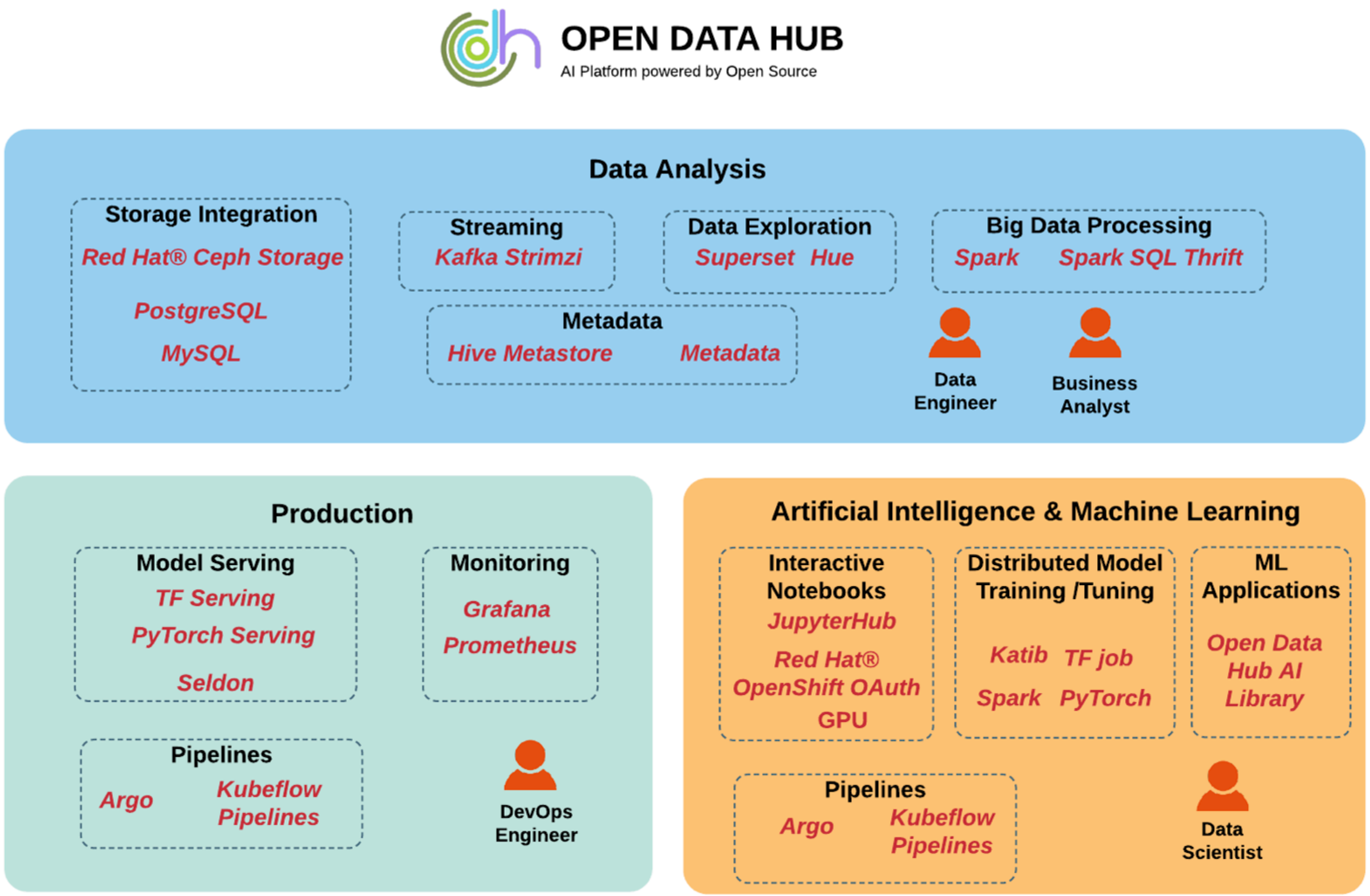
Open Data Hub 0.6 brings component updates and Kubeflow architecture

By Václav Pavlín May 7, 2020

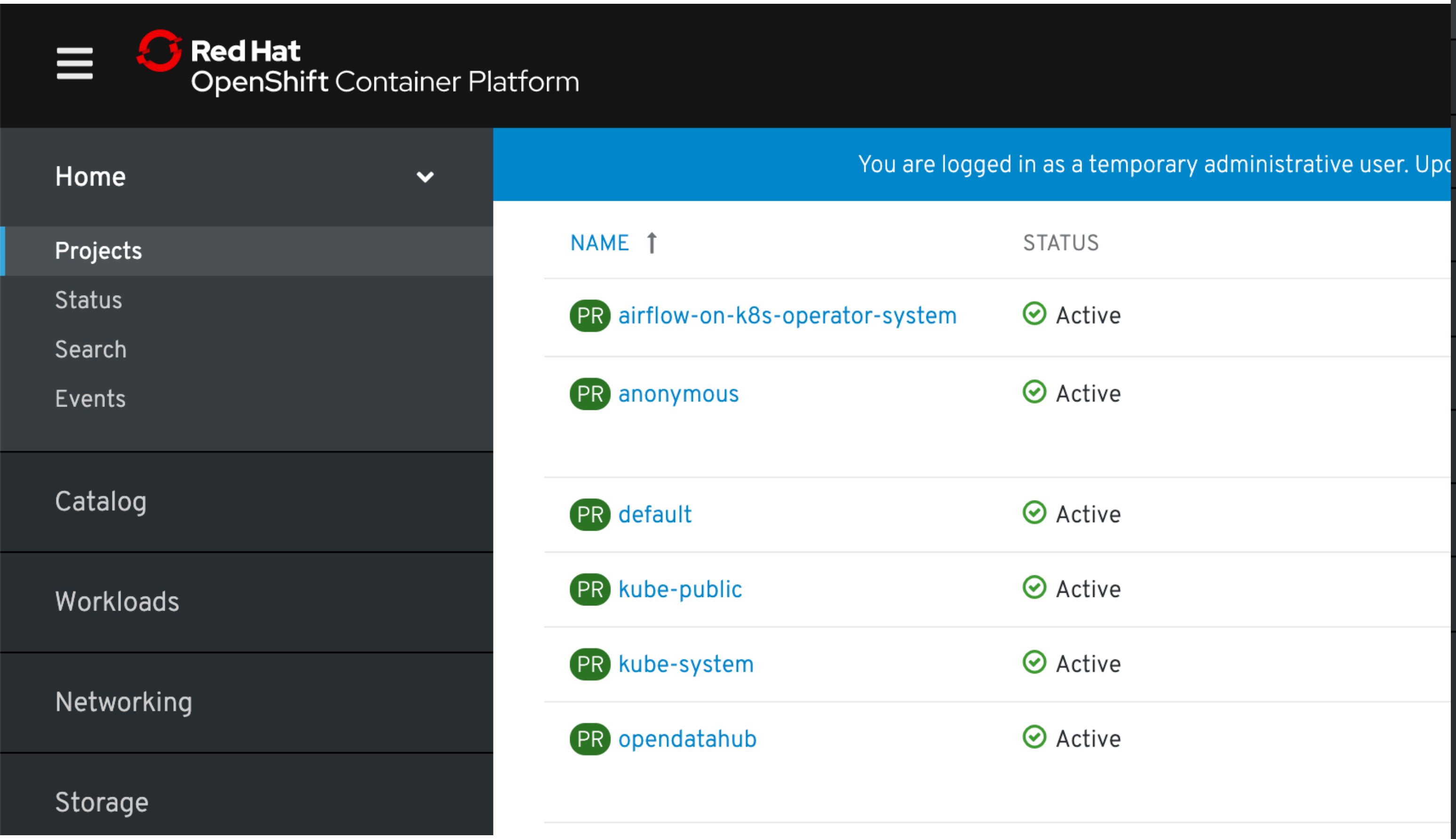


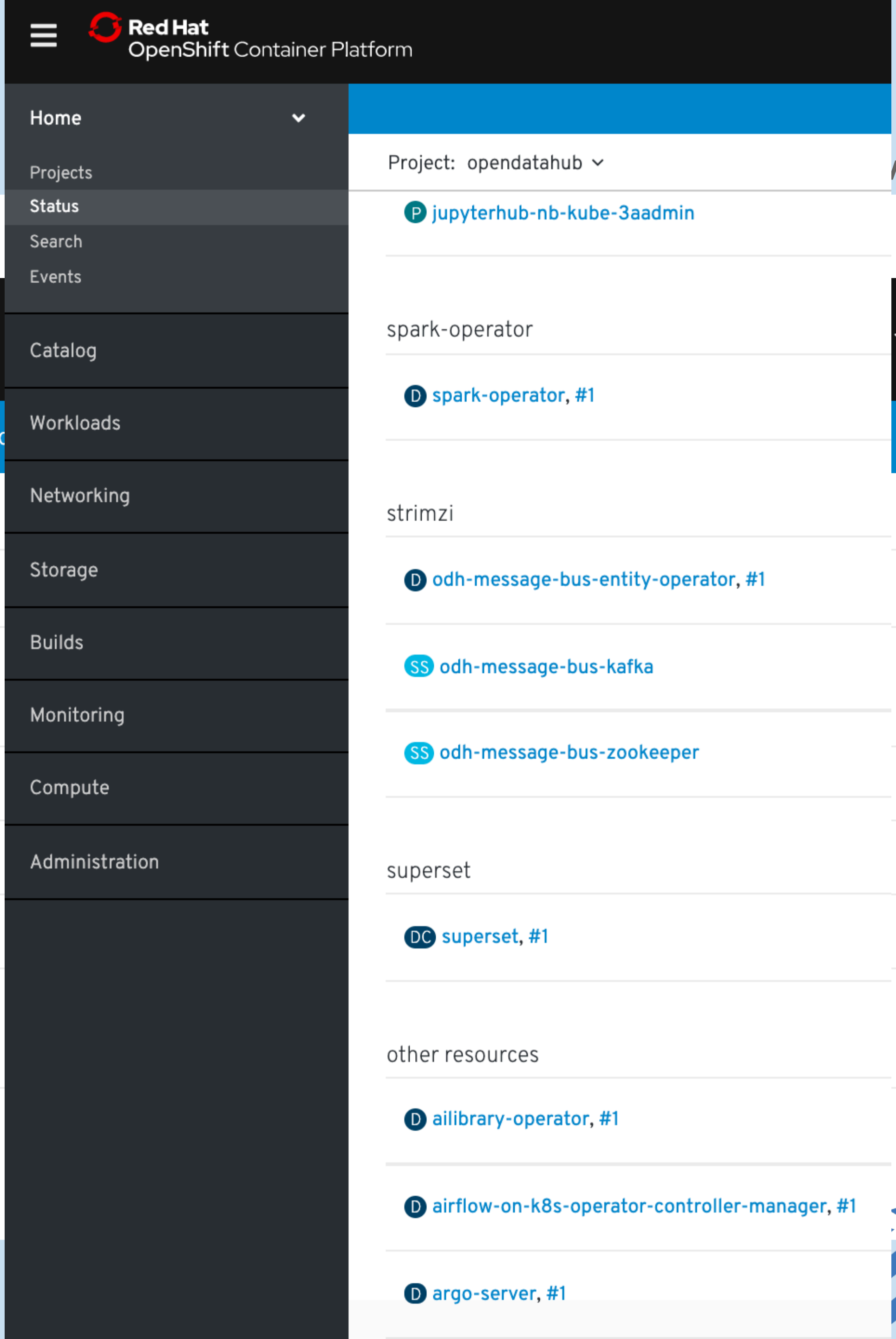
Open Data Hub (ODH) is a blueprint for building an AI-as-a-service platform on Red Hat's Kubernetes-based OpenShift 4.x. Version 0.6 of Open Data Hub comes with significant changes to the overall architecture as well as component updates and additions. In this article, we explore these changes.

OPEN DATA HUB 0.6.x

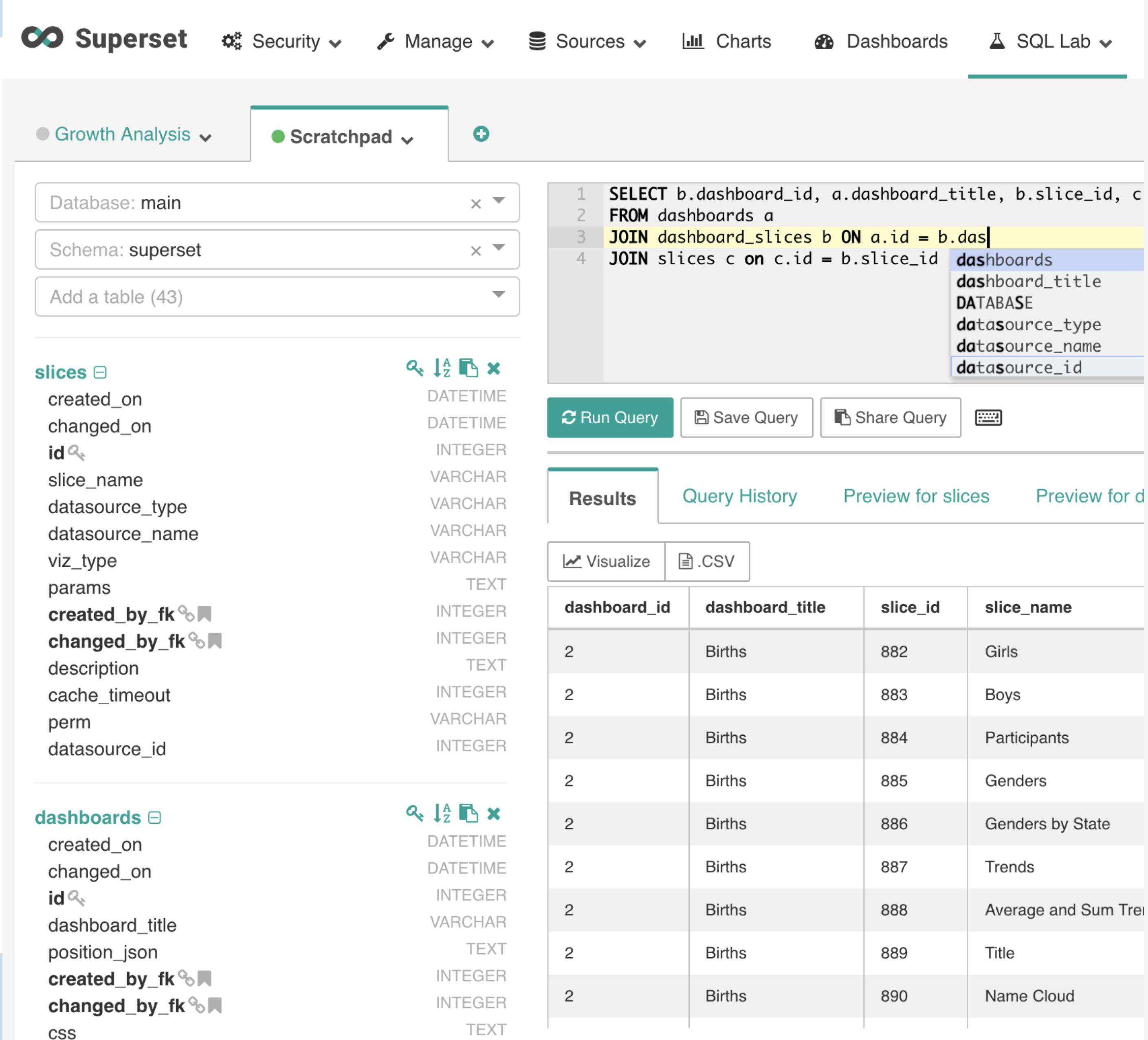


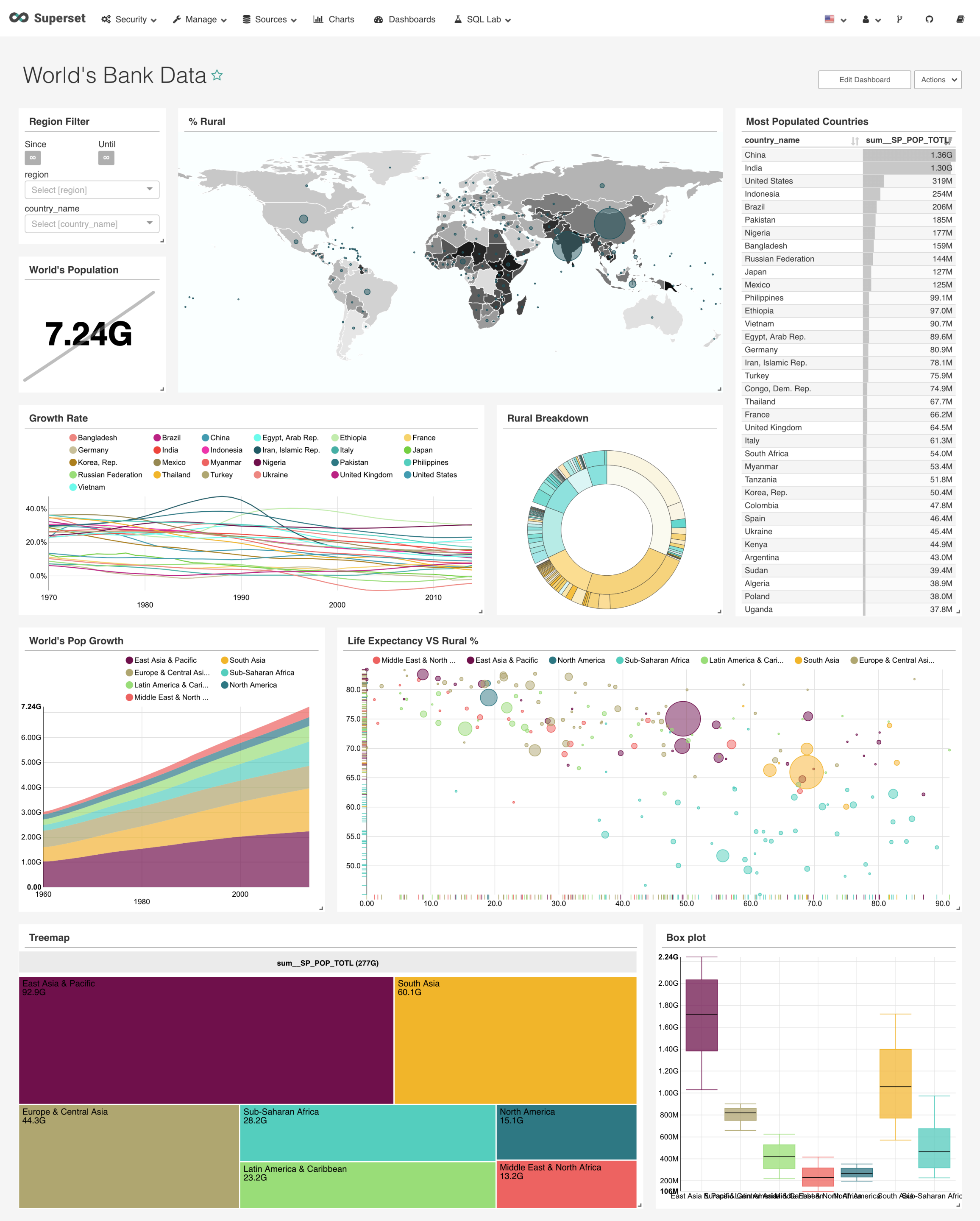
Open Data Hub in OpenShift





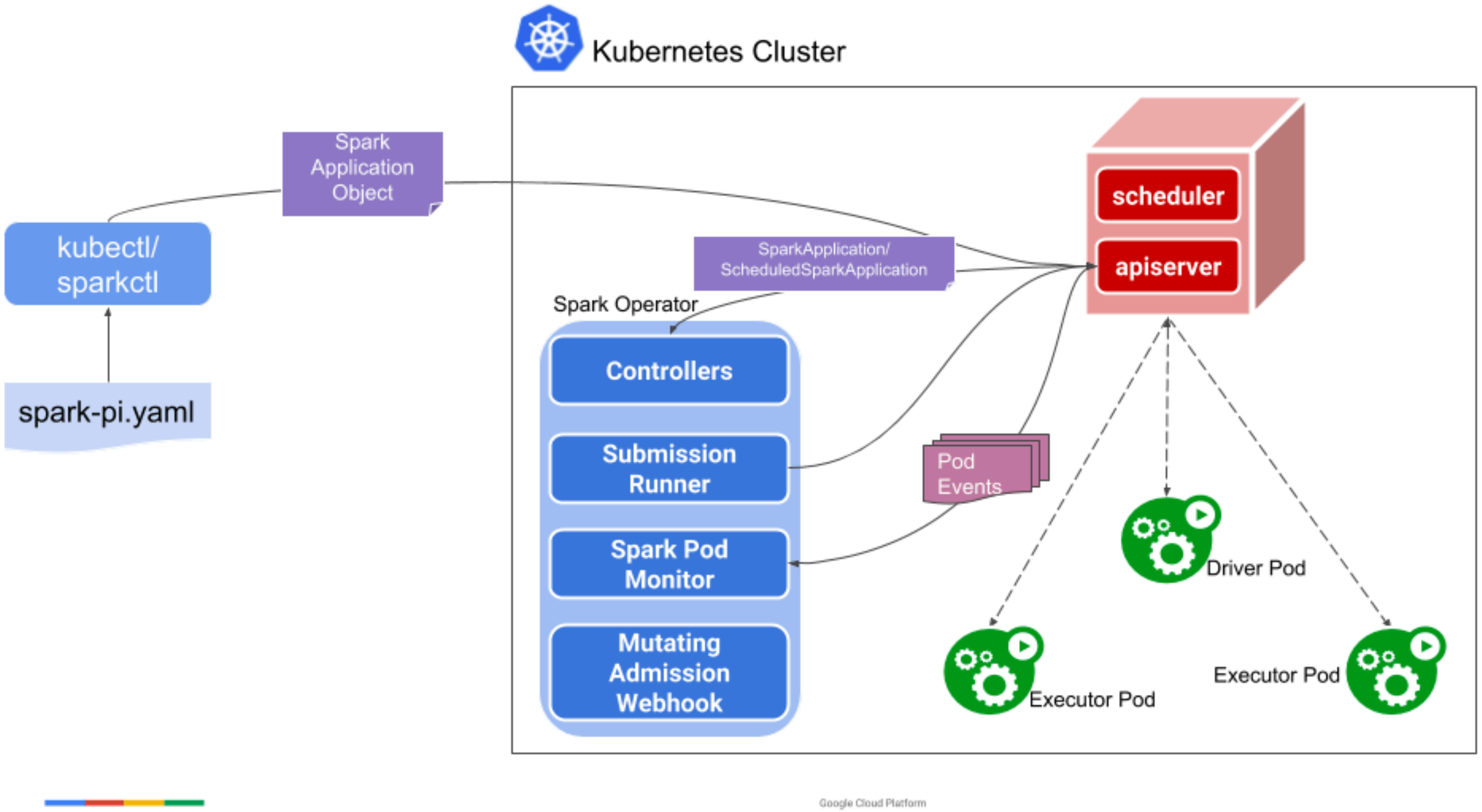
Apache Superset





Spark with Open Data Hub

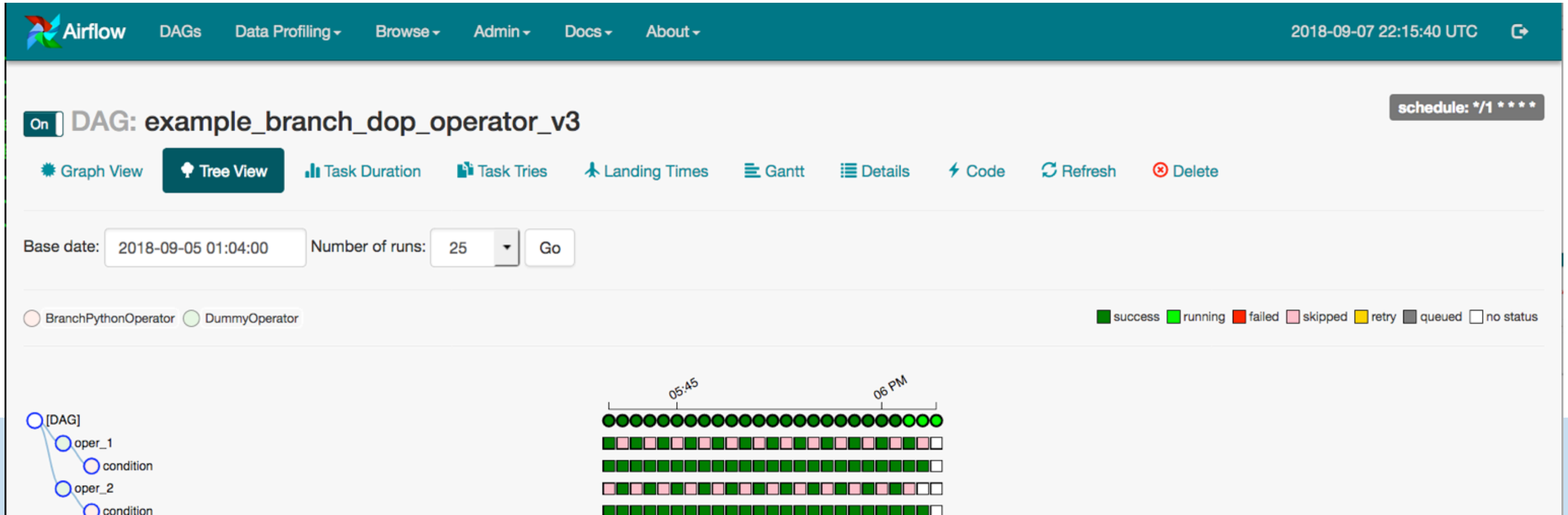
- Open Data Hub will also deploy the Spark Operator to manage Spark as an application.
- Two versions of Spark – Spark in dedicated mode and Spark on K8s
- Currently moving towards Spark on K8s Operator from Google for serverless Spark. IBM Hummingbird team investigating this



Airflow integration with Open Data Hub

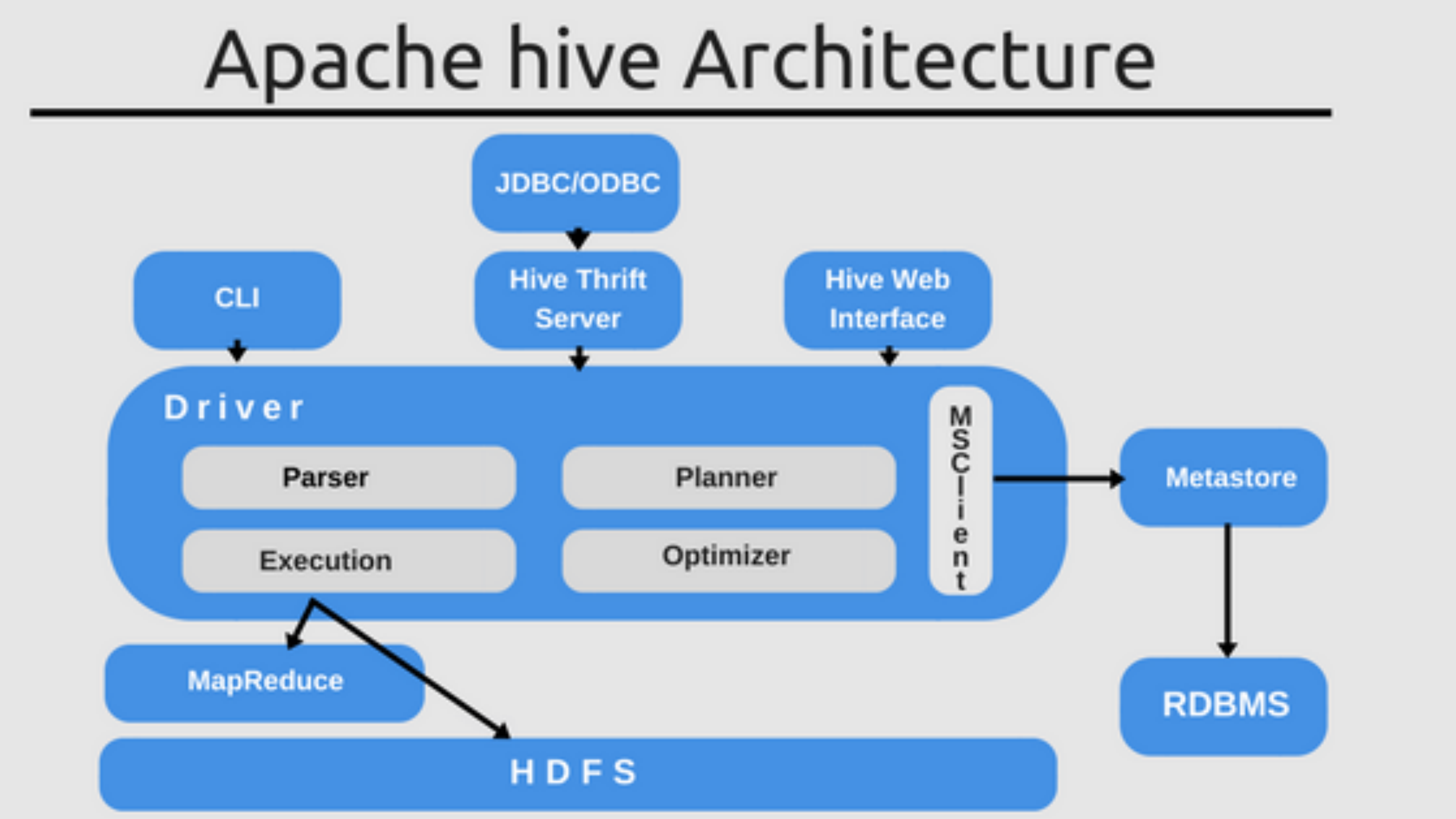
Kubeflow

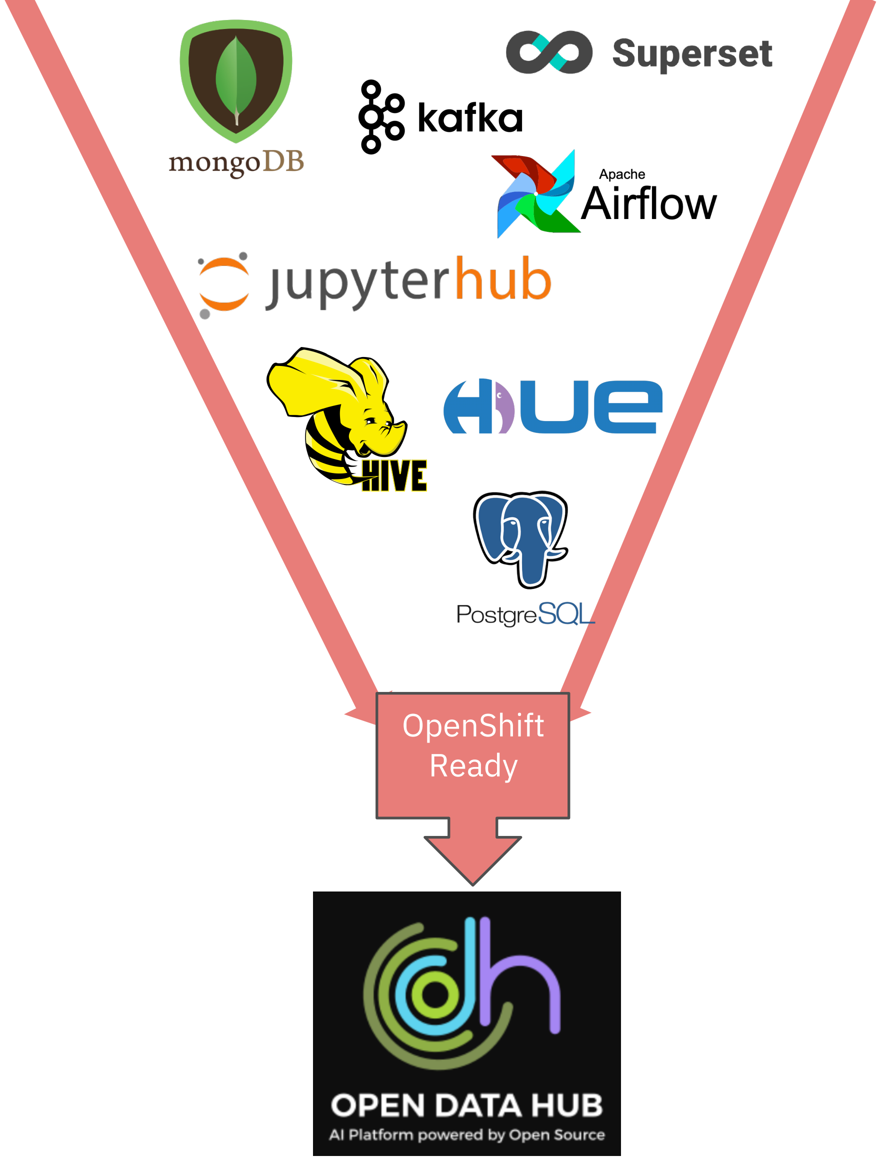
- Open Data Hub will also deploy the Airflow Operator to manage Airflow as an application.
- Using the Airflow Operator originally developed in the GoogleCloudPlatform repository and later donated to Apache.
- The Operator creates a controller-manager pod which will be created as a part of the Open Data 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.)



Apache Hive with OpenDataHub

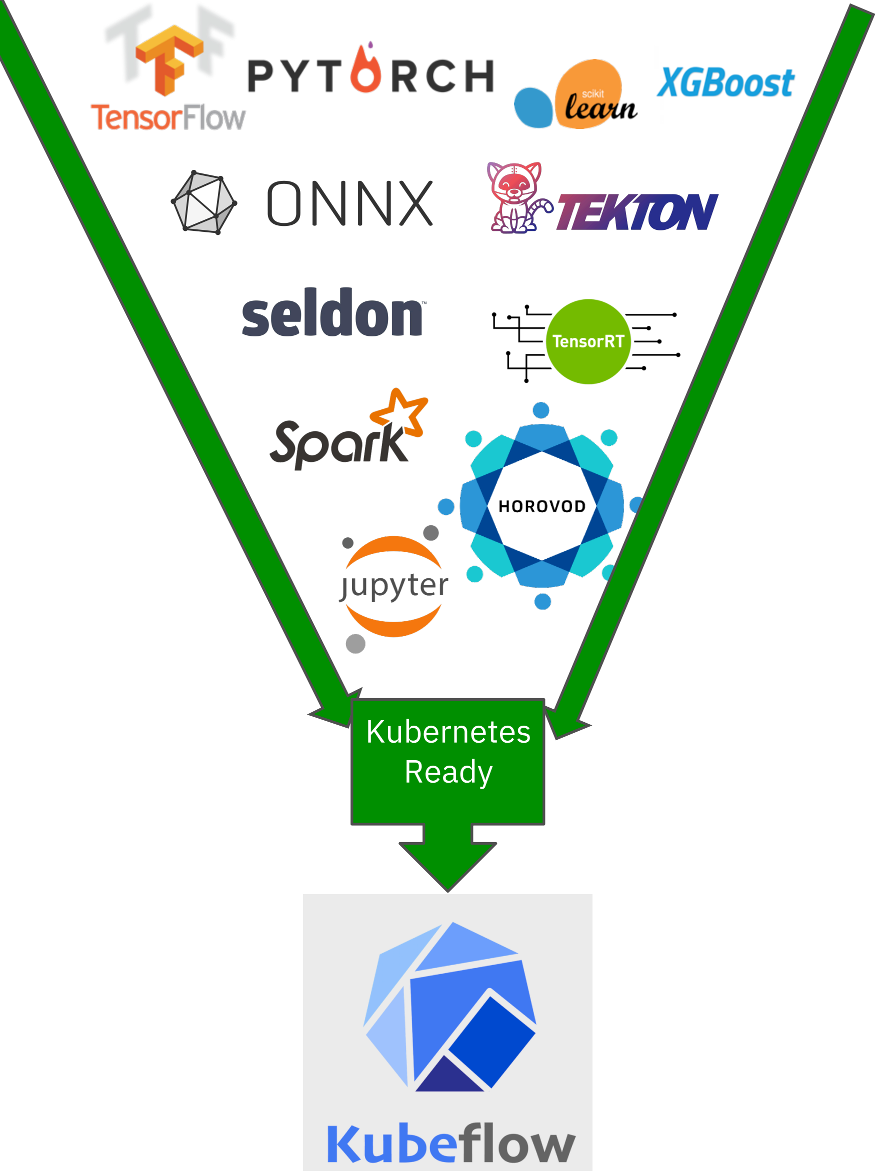
- Hive was one of the first abstraction engines to be built on top of MapReduce.
- Started at Facebook to enable data analysts to analyse data in Hadoop by using familiar SQL syntax without having to learn how to write MapReduce.
- Hive an essential tool in the Hadoop ecosystem that provides an SQL dialect 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.



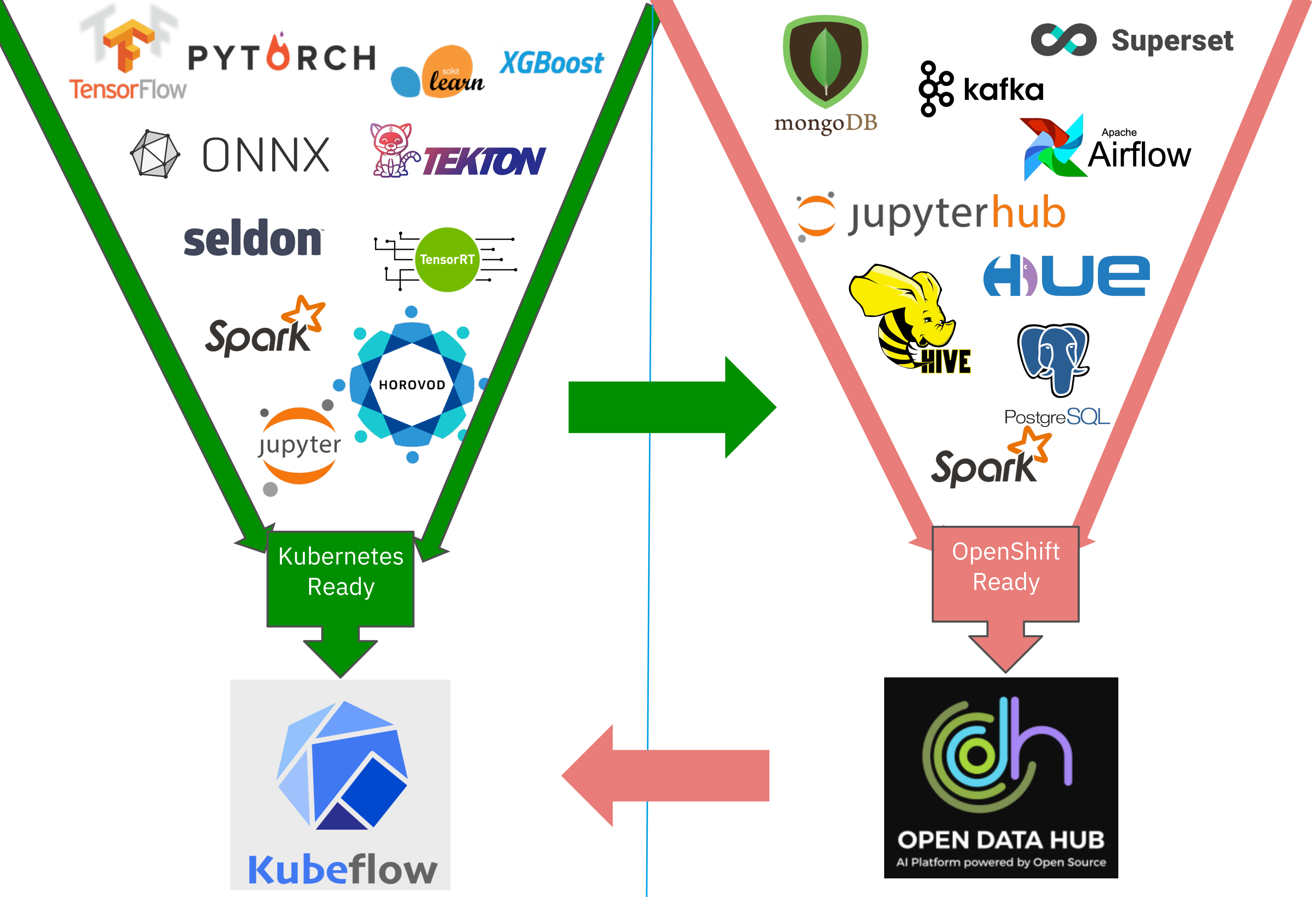


Data Platform

Operator Hub - operatorhub.io



ML and AI Platform



Upstream Kubeflow

Midstream OpenDataHub

OpenDataHub

Kubeflow

Open Source End To End

Data and AI Platform

Operator Hub - operatorhub.io

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

IBM

Coming Next: Kubeflow Dojo

Kubeflow

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/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

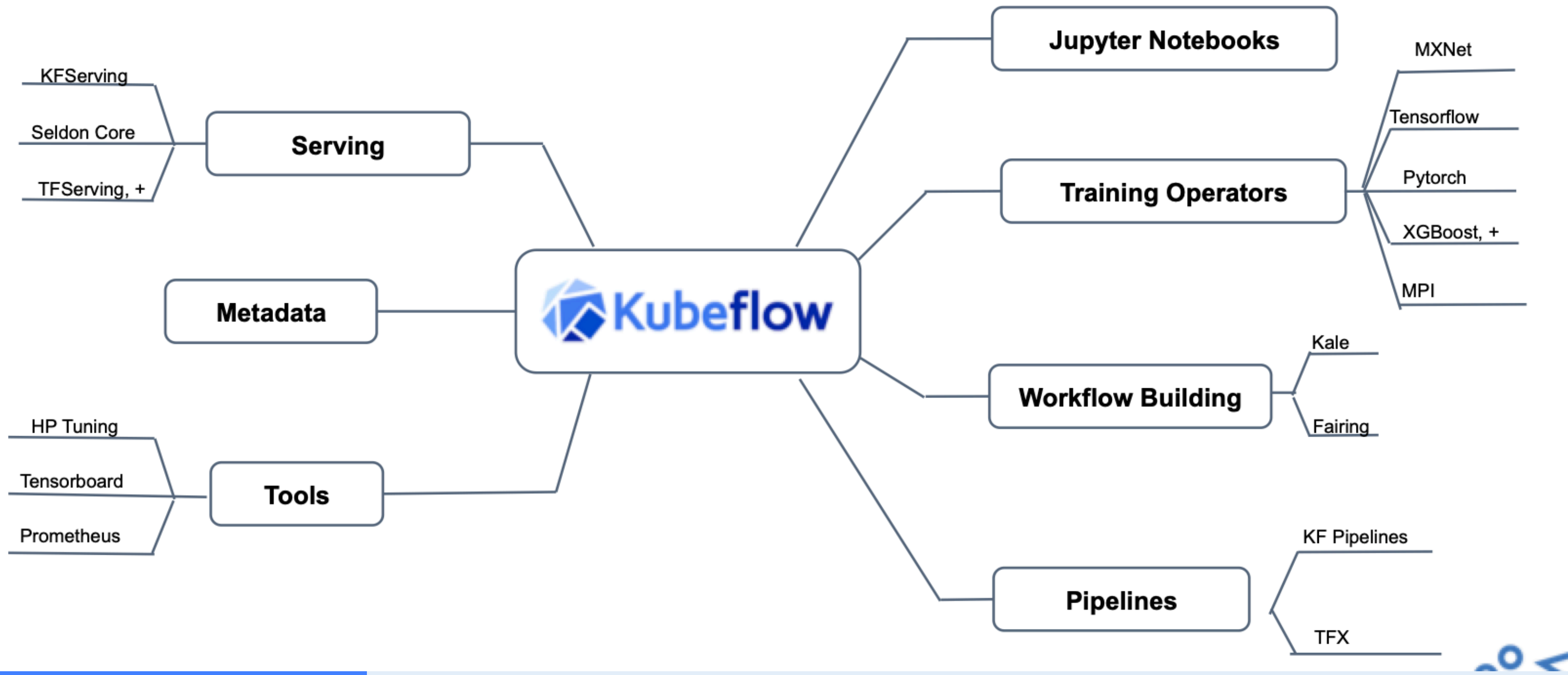
Kubeflow Dojo: Tips for success

- 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