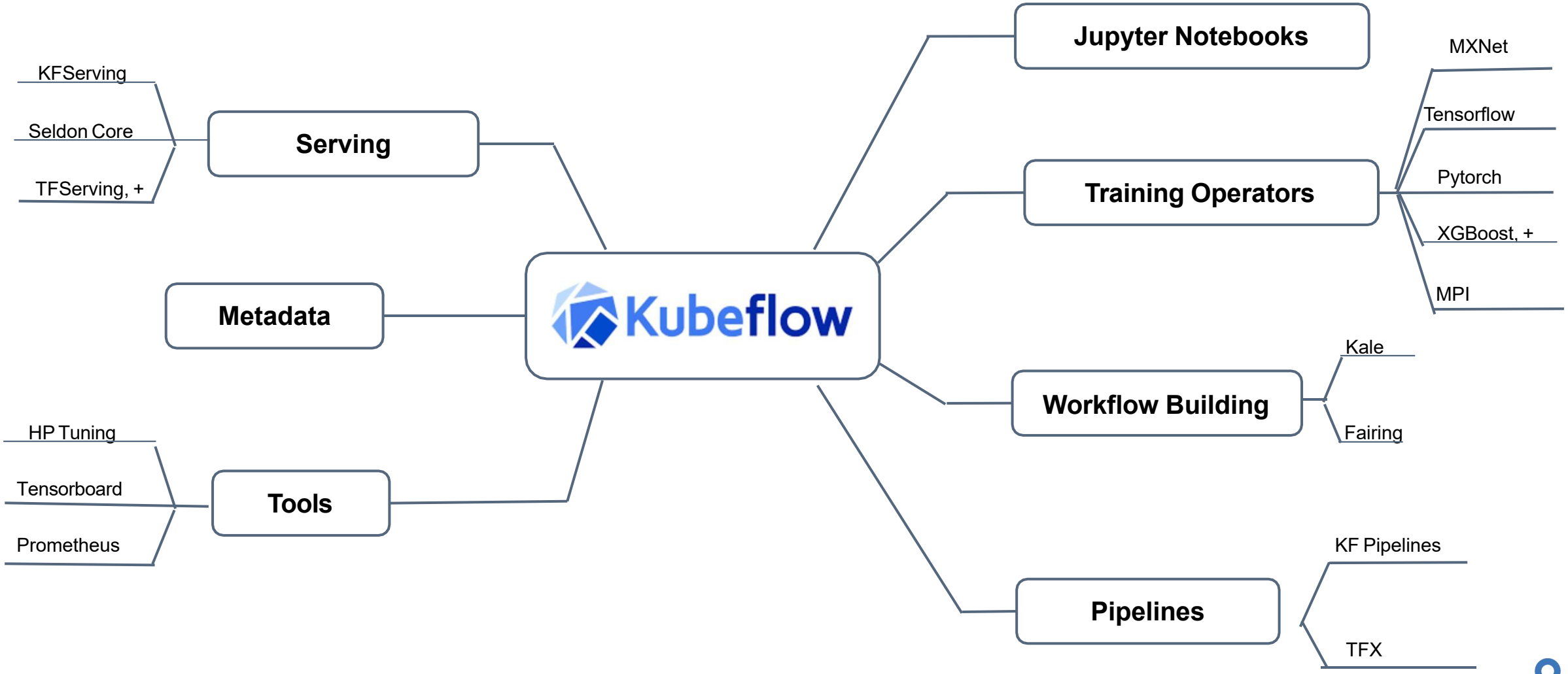
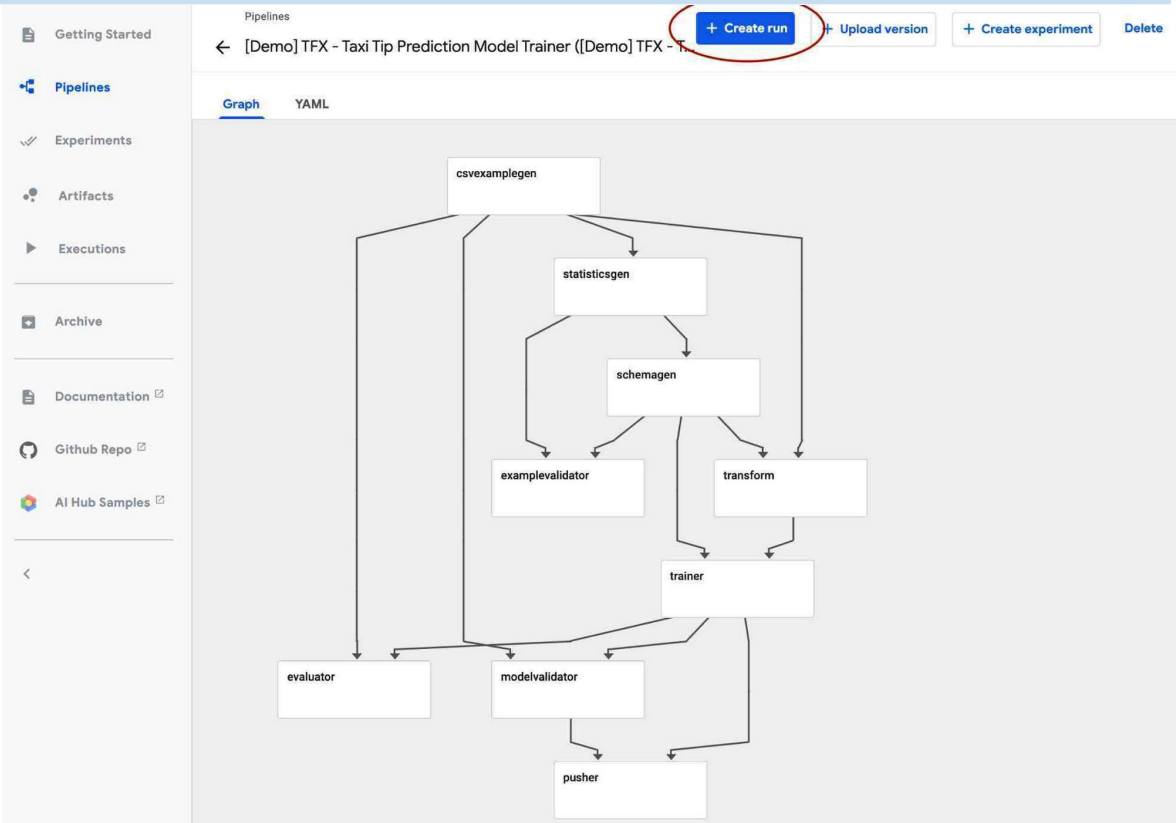
Kubeflow Pipelines and Tekton

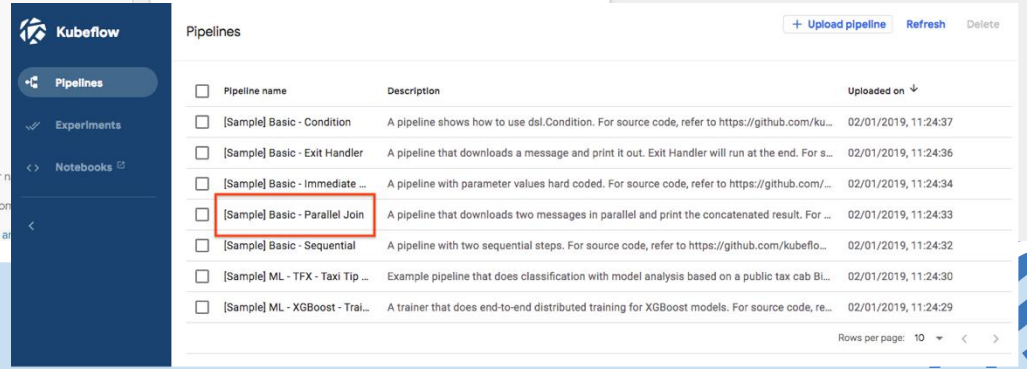
Tommy Li, Feng Li



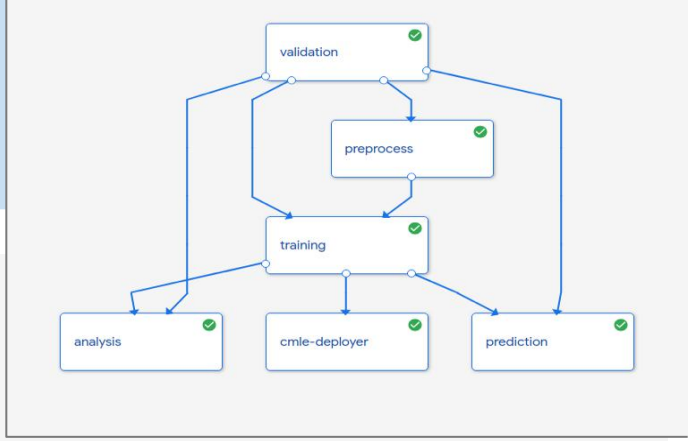
Kubeflow Pipelines

- 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

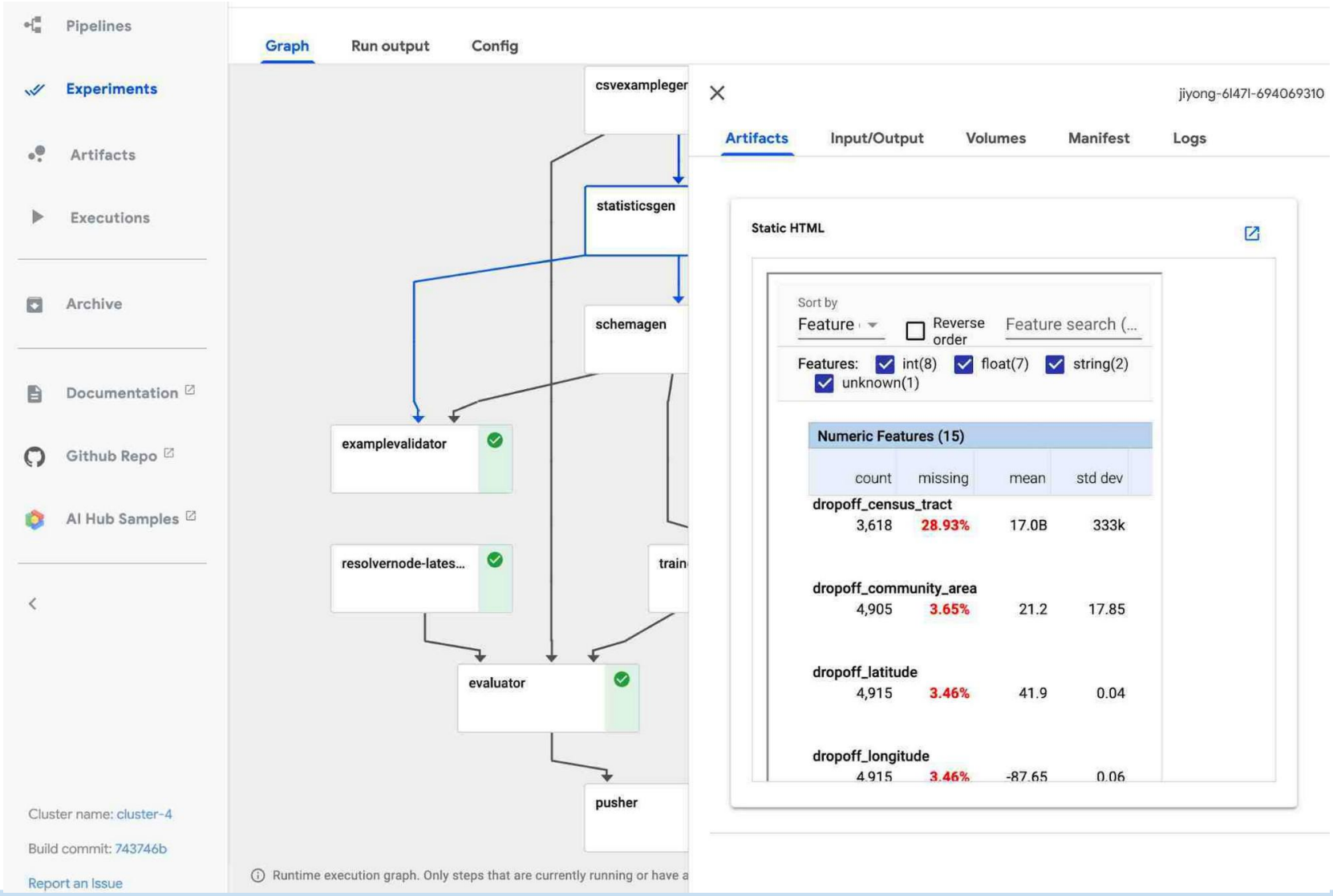



```
@dsl.pipeline(name='Taxi Cab Classification Pipeline Example’)
def taxi_cab_classification(  output_dir,
    project,
    Train_data      = 'gs://bucket/train.csv',
    Evaluation_data = 'gs://bucket/eval.csv',

    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)
```

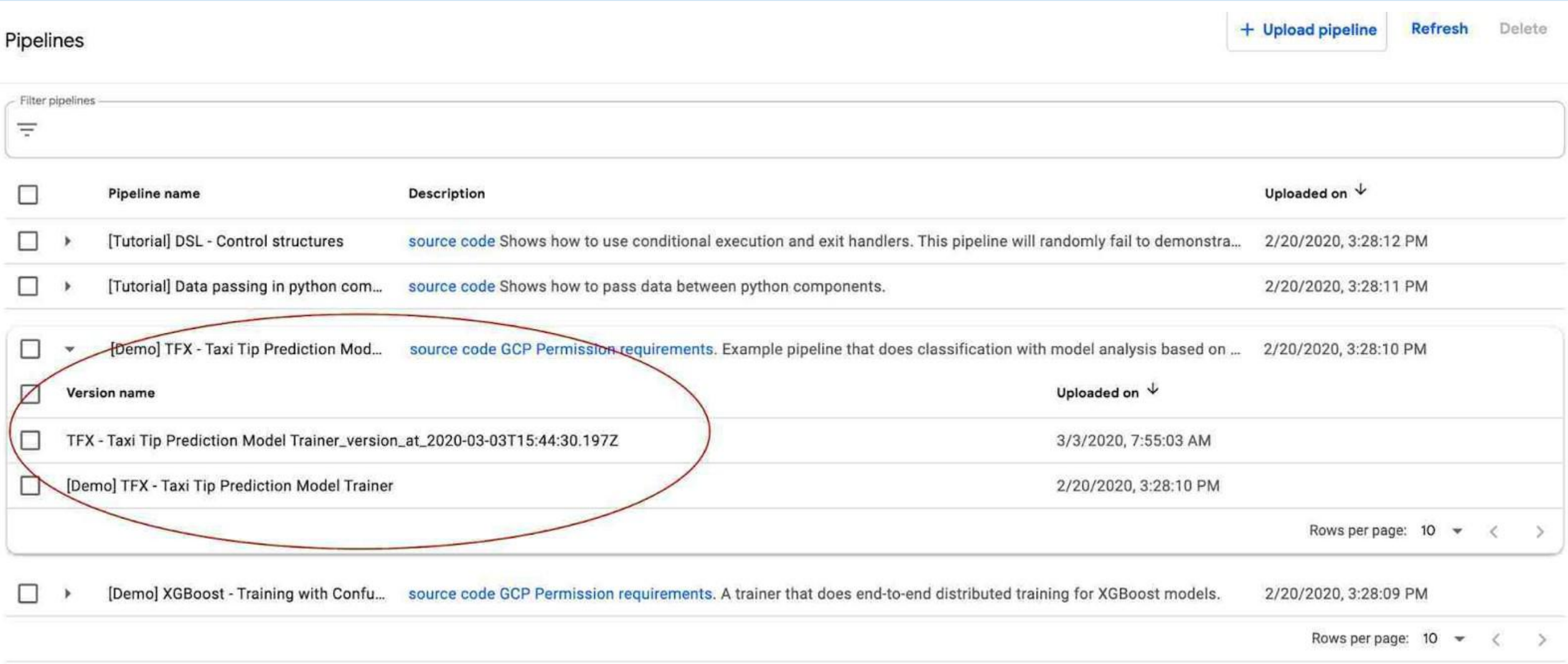
Compile and Submit Pipeline Run

```
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

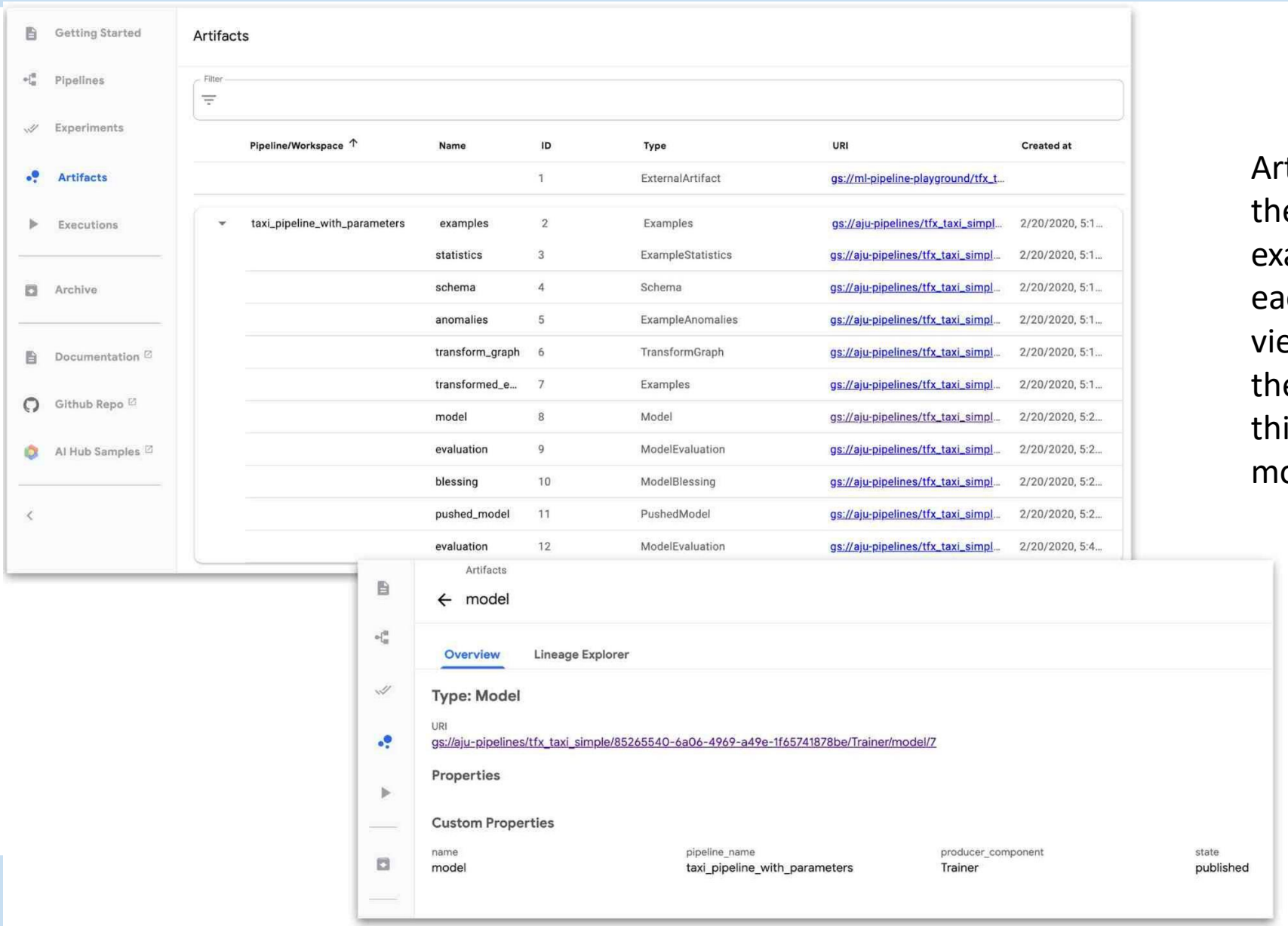


Pipelines versioning



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

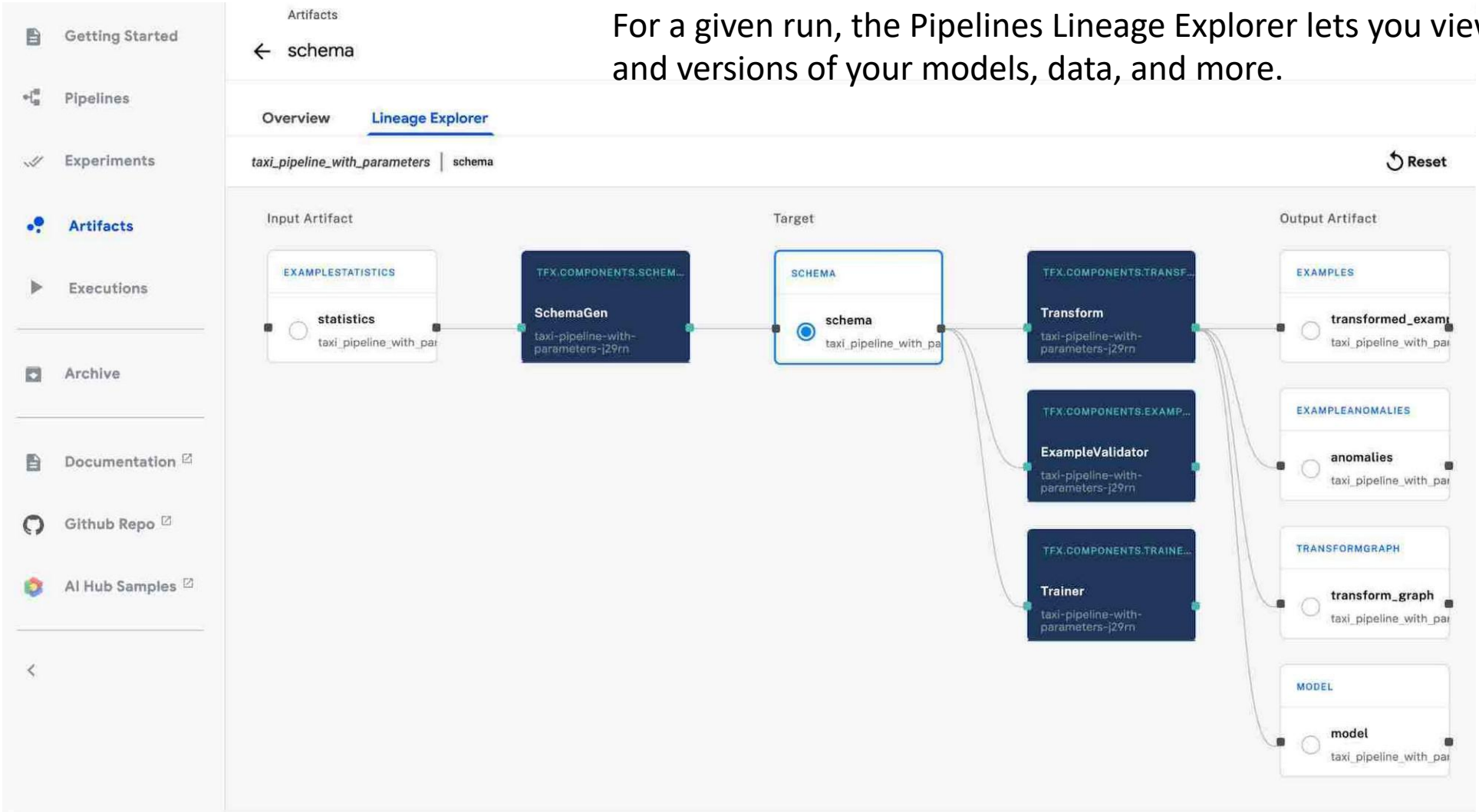
Artifact Tracking



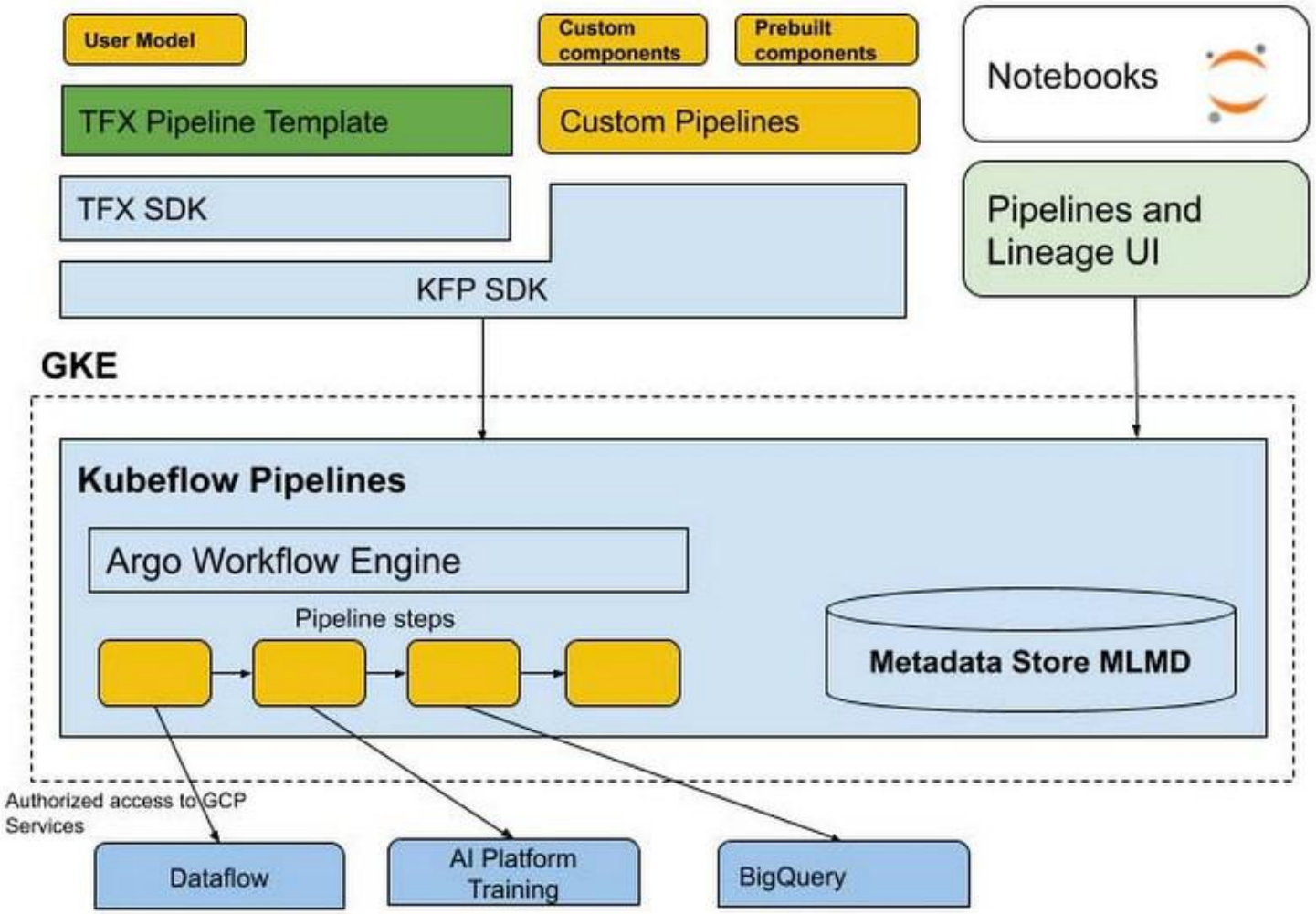
Artifacts for a run of the “TFX Taxi Trip” example pipeline. For each artifact, you can view details and get the artifact URL—in this case, for the model.

Lineage Tracking

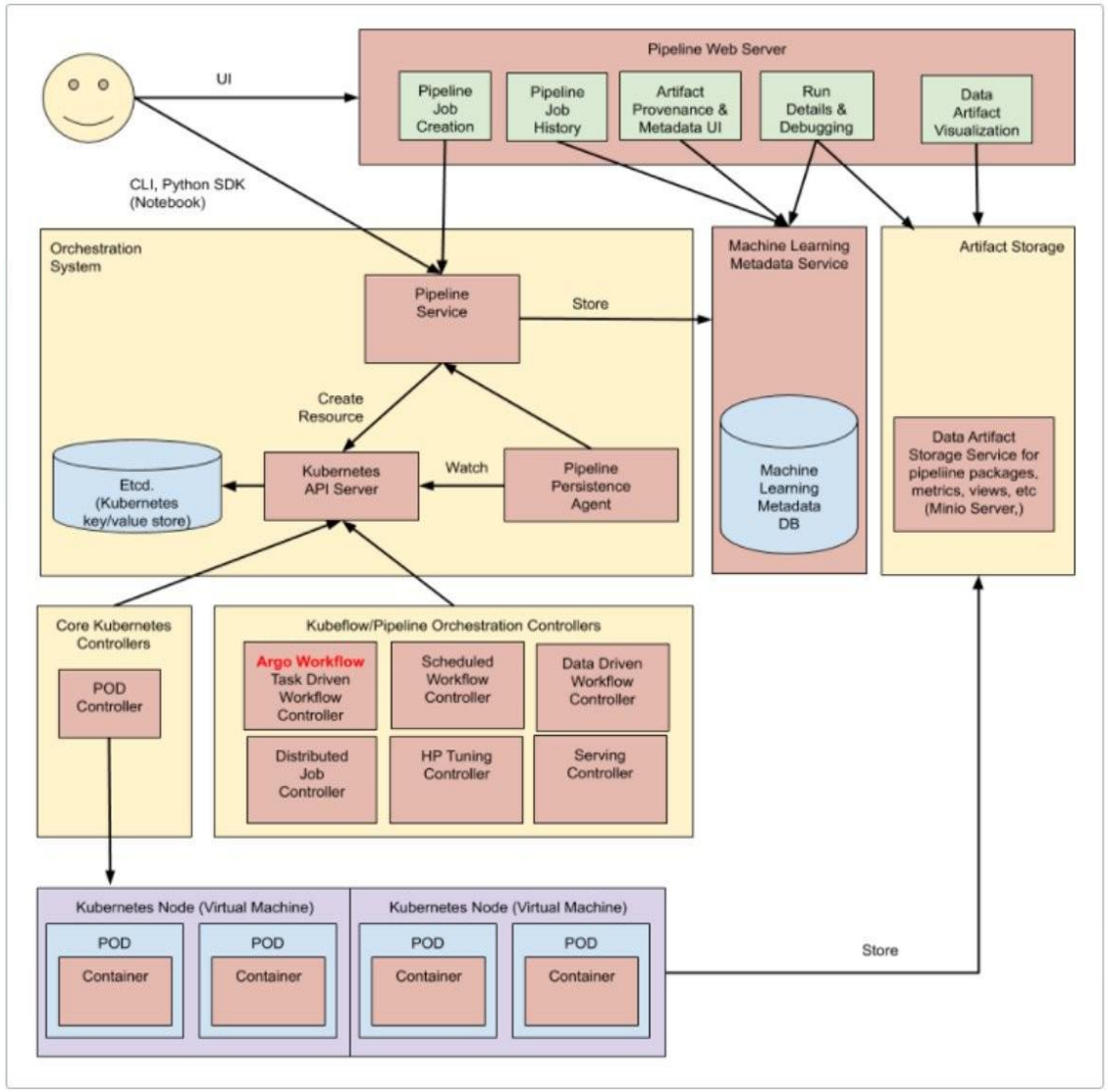
For a given run, the Pipelines Lineage Explorer lets you view the history and versions of your models, data, and more.



Kubeflow Pipeline Architecture

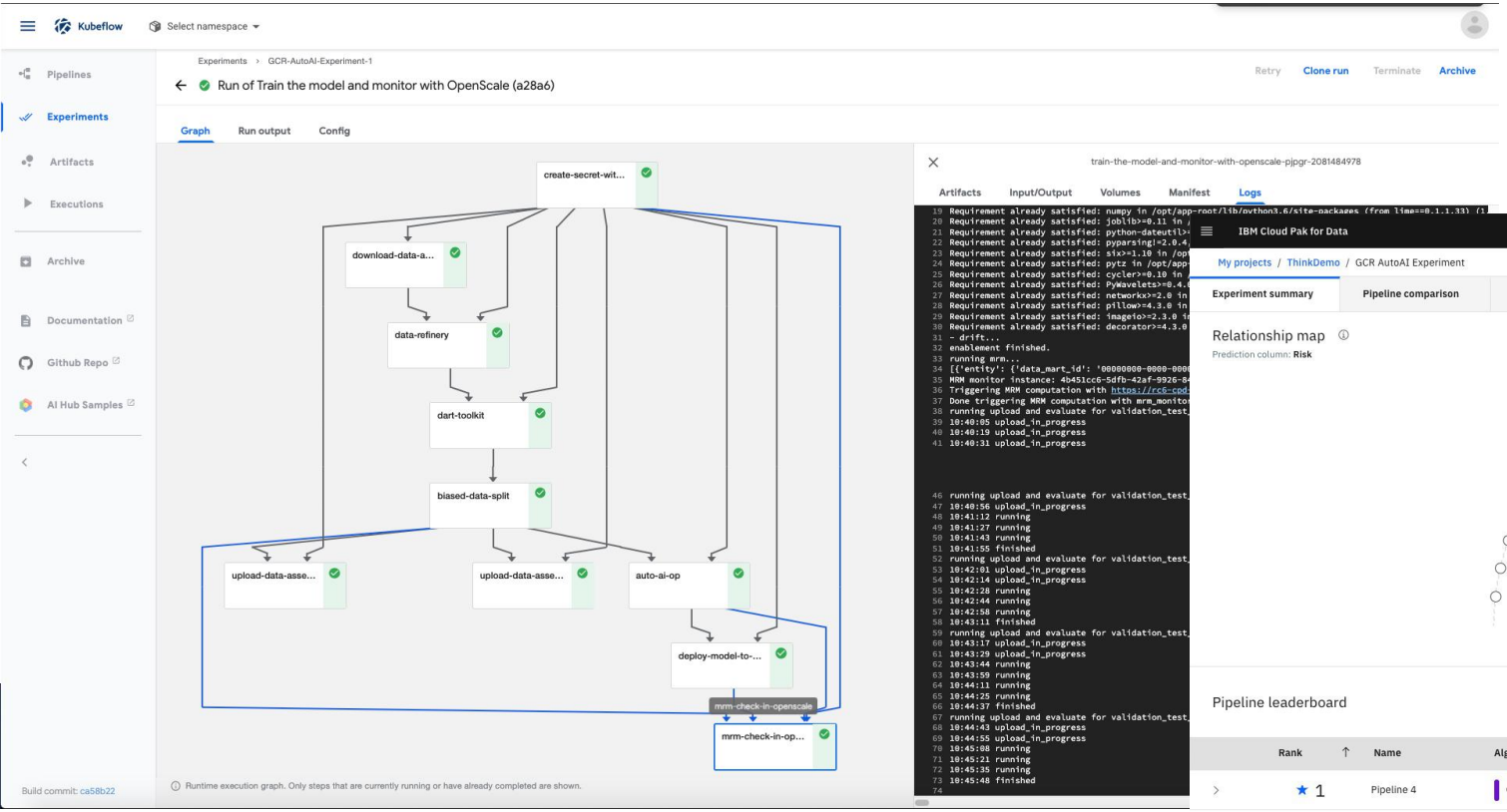


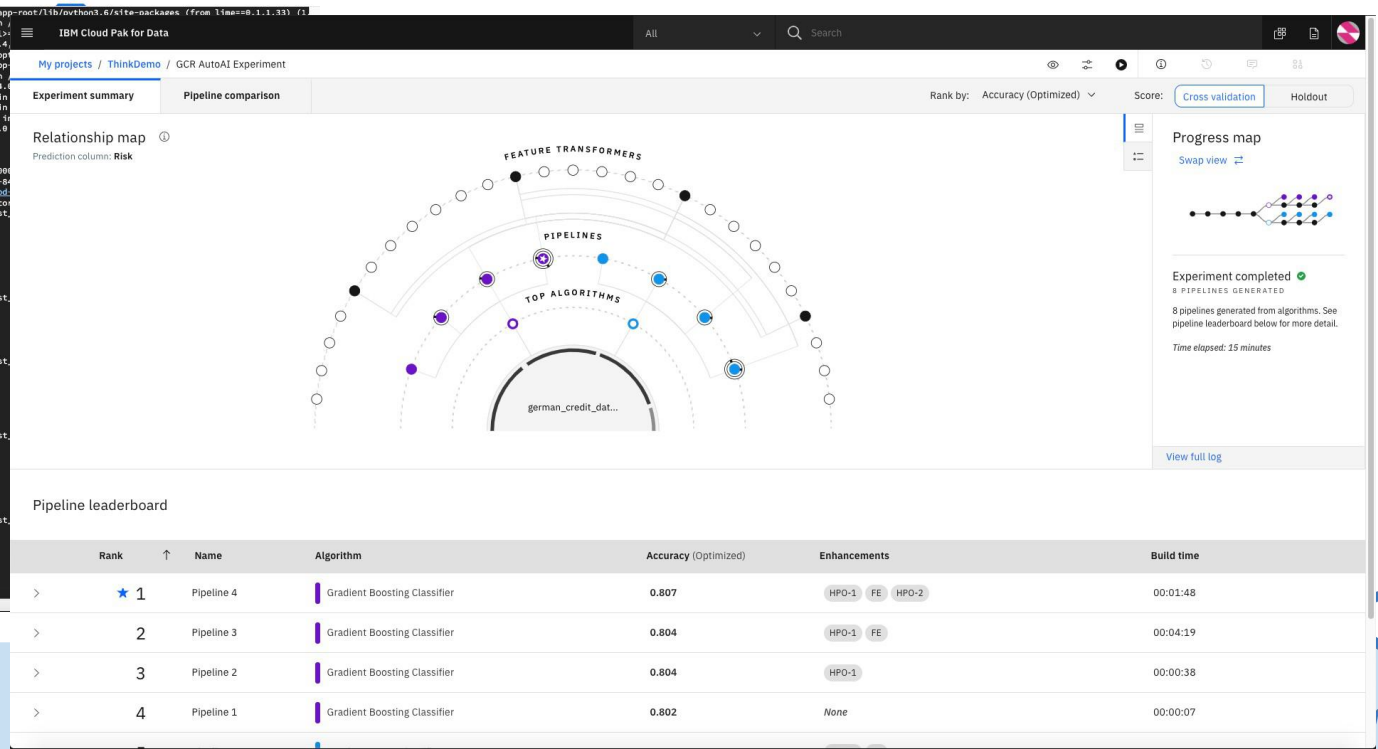
Kubeflow Pipeline Architecture



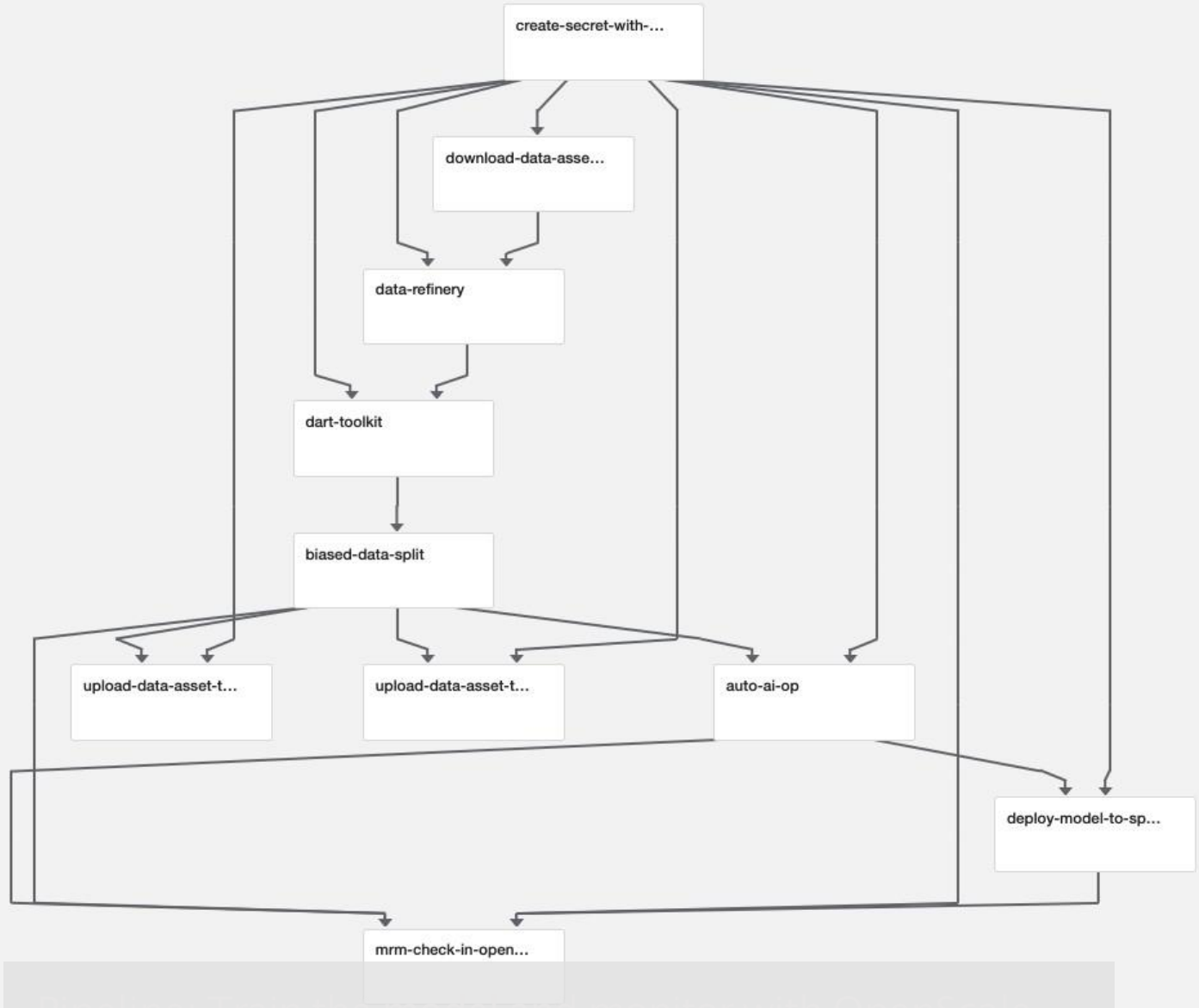
Goals

- 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

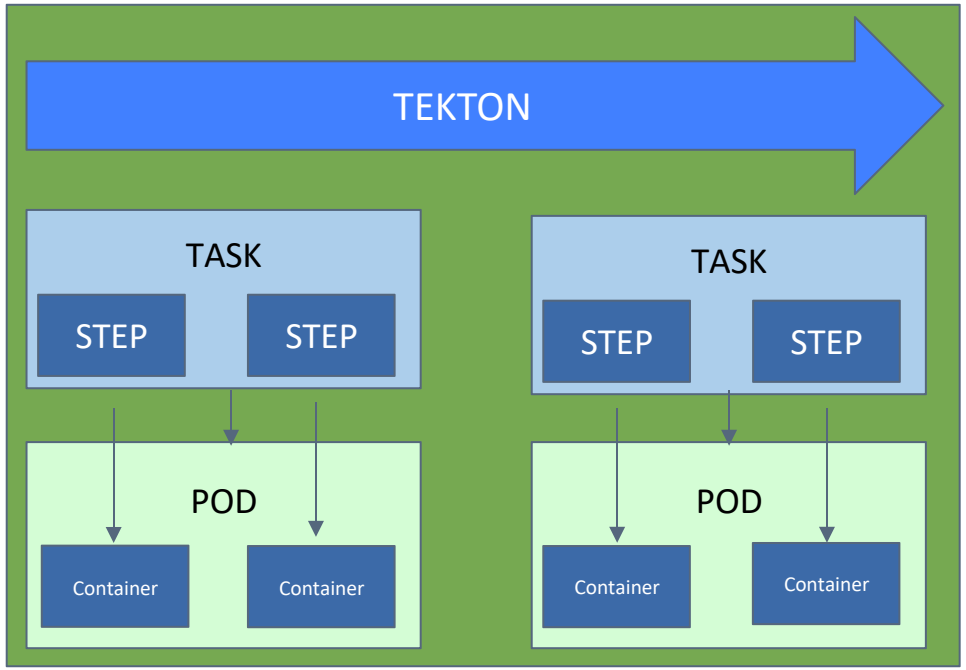
catalog_name DataCatalog

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

pre_production_space_uid 1dd2aaec-781a-4712-a7ff-ae1862cf7a84

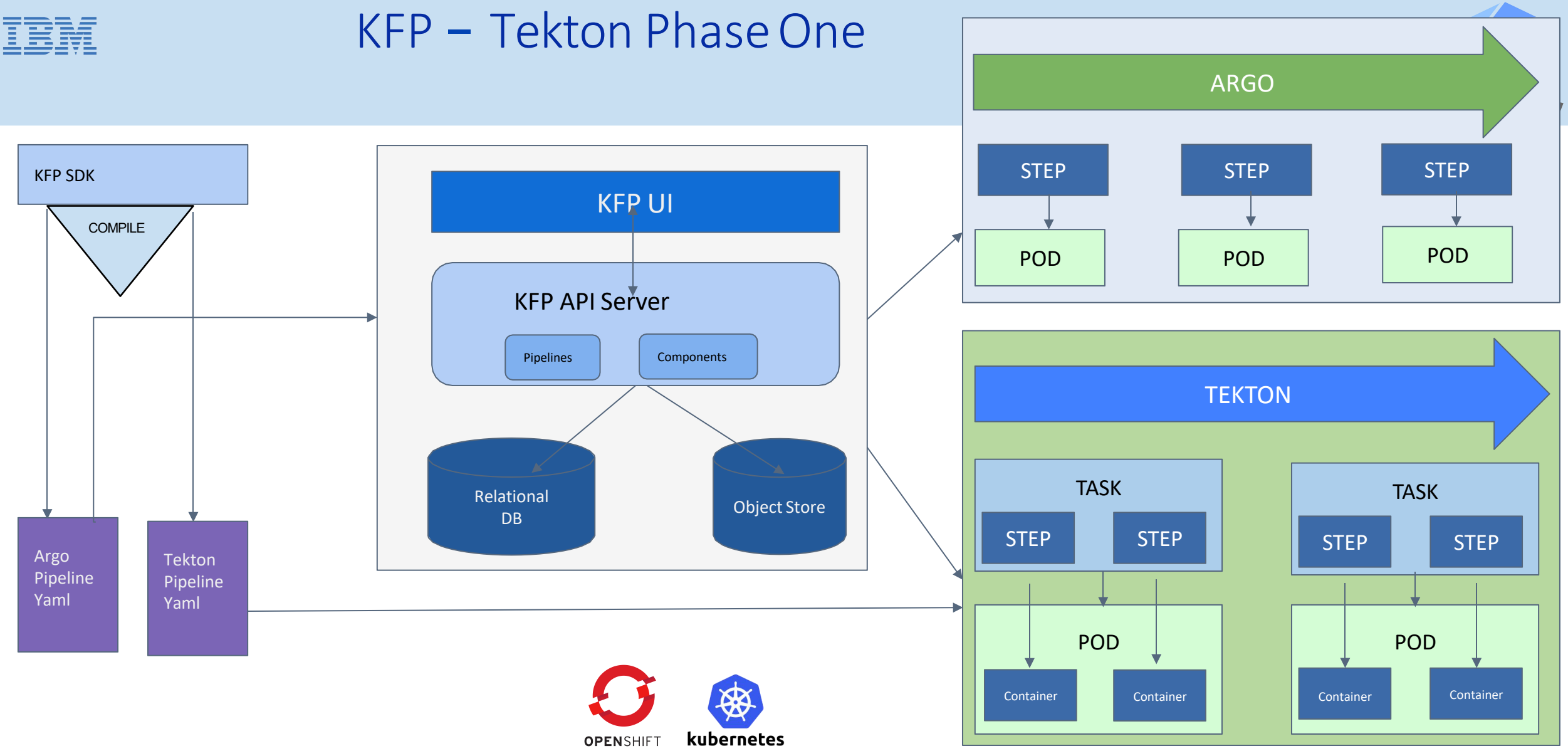
Tekton

- 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.

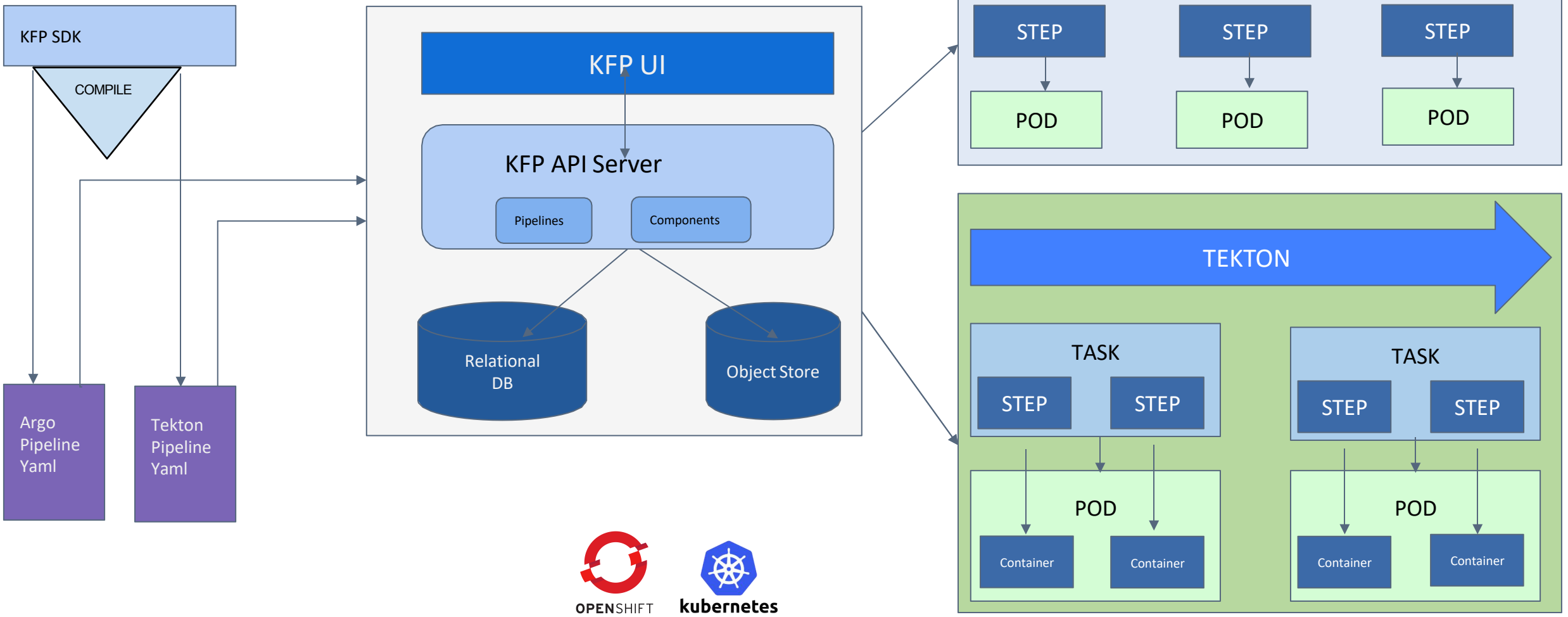
KFP – Tekton Phase One



Pluggable Components

Spark | Watson Studio | WML | Open Scale | Kubeflow Training | Seldon | AIF360 | ART | KATIB | KFSERVING | ...

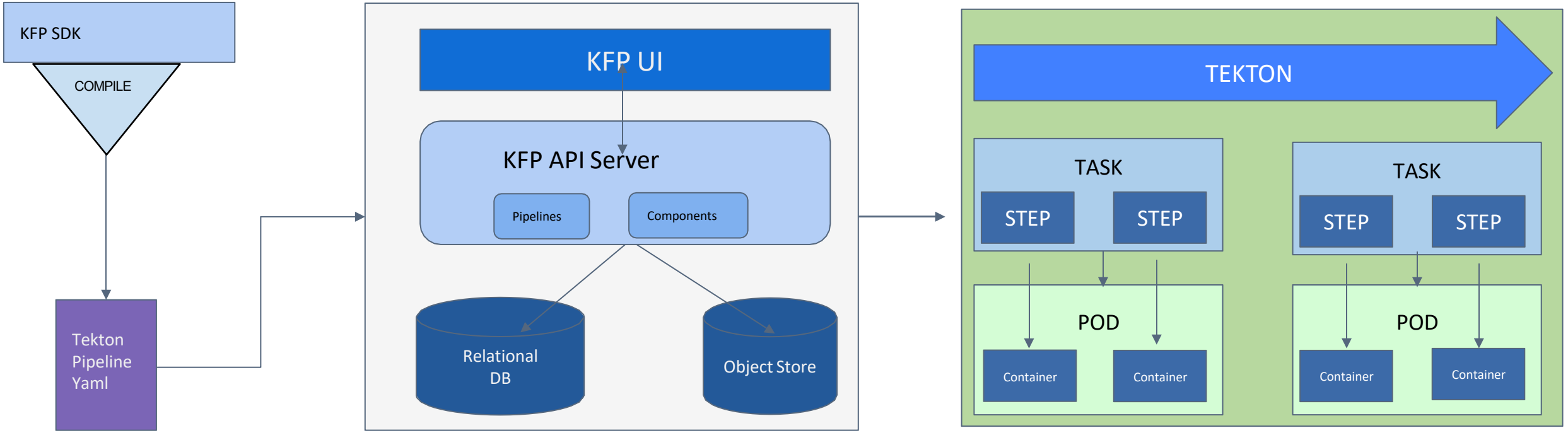
KFP – Tekton Phase Two



Pluggable Components

Spark | Watson Studio | WML | Open Scale | Kubeflow Training | Seldon | AIF360 | ART | KATIB | KFSERVING | ...

Kubeflow Pipelines with Tekton: Delivered



OPENSHIFT kubernetes

Pluggable Components

Spark | Watson Studio | WML | Open Scale | Kubeflow Training | Seldon | AIF360 | ART | KATIB | KFSERVING | ...

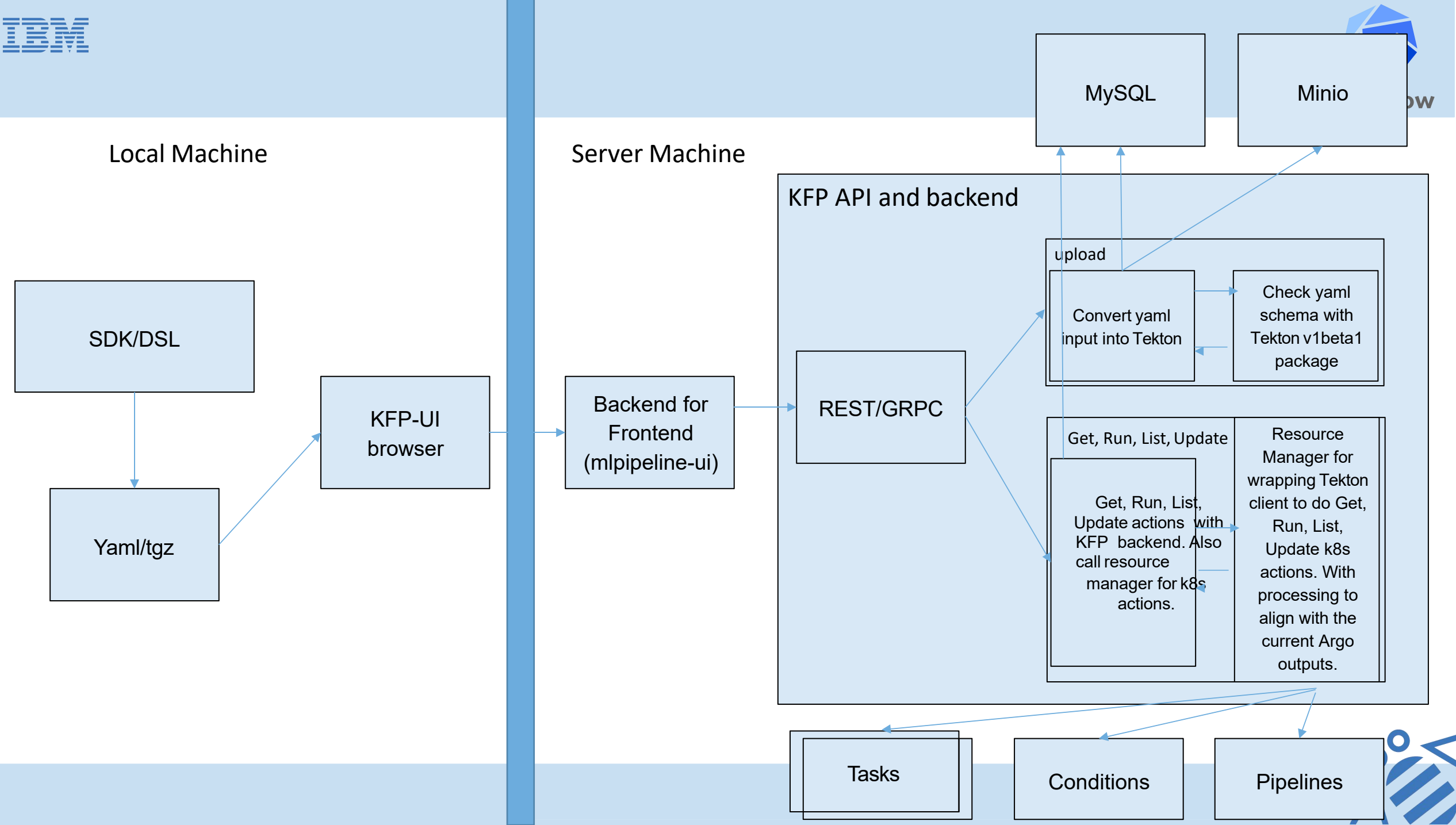
DSL features implemented

- Pipeline DSL features with native Tekton implementation
  - pod_annotations and pod_labels
  - Retries
  - Volumes
  - Timeout for Tasks and Pipelines
  - RunAfter
  - Input Parameters
  - ContainerOp
  - Affinity, Node Selector, and Tolerations
- Pipeline DSL features with custom Tekton implementation
  - Features with same behavior as Argo
    - InitContainers
    - Conditions
    - ResourceOp, VolumeOp, and VolumeSnapshotOp
    - Output Parameters
    - Input Artifacts
    - Output Artifacts
  - Features with limitations
    - ParallelFor - Tracking issue
    - Variable Substitutions - Tracking issue
    - ImagePullSecrets - Tracking issue
  - Features with different behavior than Argo
    - Sidecars - Tracking issue
- Pipeline features that are unavailable on Tekton
  - Exit Handler - Tracking PR

Pipeline samples we are running

- MNIST End to End example with Kubeflow components
- Hyperparameter tuning using Katib
- Trusted AI Pipeline with AI Fairness 360 and Adversarial Robustness 360 components
- Training and Serving Models with Watson Machine Learning
- Lightweight python components example
- The flip-coin pipeline
- Nested pipeline example

https://github.com/kubeflow/kfp-tekton/blob/master/samples/README.md


Local Machine
Server Machine
MySQL
Minio
KFP API and backend
SDK/DSL
Yaml/tgz
KFP-UI browser
Backend for Frontend (mlpipeline-ui)
REST/GRPC
upload
Convert yaml input into Tekton
Check yaml schema with Tekton v1beta1 package
Get, Run, List, Update
Get, Run, List, Update actions with KFP backend. Also call resource manager for k8s actions.
Resource Manager for wrapping Tekton client to do Get, Run, List, Update k8s actions. With processing to align with the current Argo outputs.
Tasks
Conditions
Pipelines

In [12]:
```
kfp.Client(host="169.62.93.163").run_pipeline(experiment_id="74f7f363-96f8-487e-8632-4980b0971c7a",
                                              job_name="sample-job",
                                              pipeline_id="e684bc9e-cb30-4a3e-88f7-5c768202e6b7")
```

Run link here

Out[12]:
```
{'created_at': datetime.datetime(2020, 5, 22, 0, 7, 46, tzinfo=tzutc()),
 'description': None,
 'error': None,
 'finished_at': datetime.datetime(1970, 1, 1, 0, 0, tzinfo=tzutc()),
 'id': '752ed34b-4ade-4654-b7d7-829618edd530',
 'metrics': None,
 'name': 'sample-job',
 'pipeline_spec': {'parameters': None,
                   'pipeline_id': 'e684bc9e-cb30-4a3e-88f7-5c768202e6b7',
                   'pipeline_manifest': None,
                   'pipeline_name': 'tekton-parameters',
                   'workflow_manifest': '{"kind":"PipelineRun","apiVersion":"tekton.dev/v1beta1","metadata":{"nam
e":"pipelinerun-with-taskspec-to-echo-message","creationTimestamp":null},"spec":{"pipelineSpec":{"tasks":[{"name":"ec
ho-message","taskSpec":{"params":[{"name":"MESSAGE","type":"string","default":"Hello '
                                  'World!"}],"steps":[{"name":"echo","image":"ubuntu","resources":{},"scrip
```

Experiments

+ Create run | + Create experiment | Compare runs | Clone run | Archive | Refresh

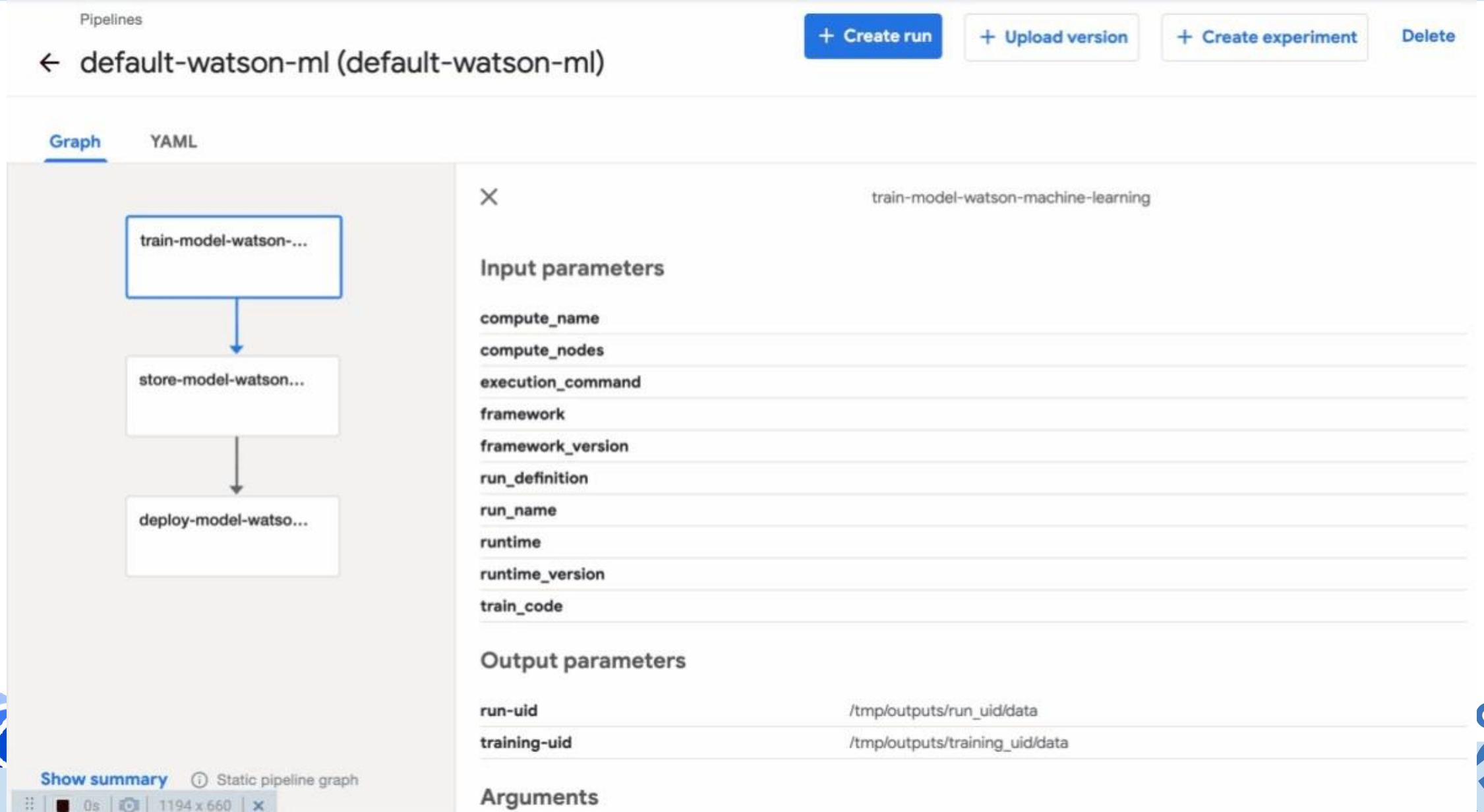
All experiments | All runs

Filter experiments

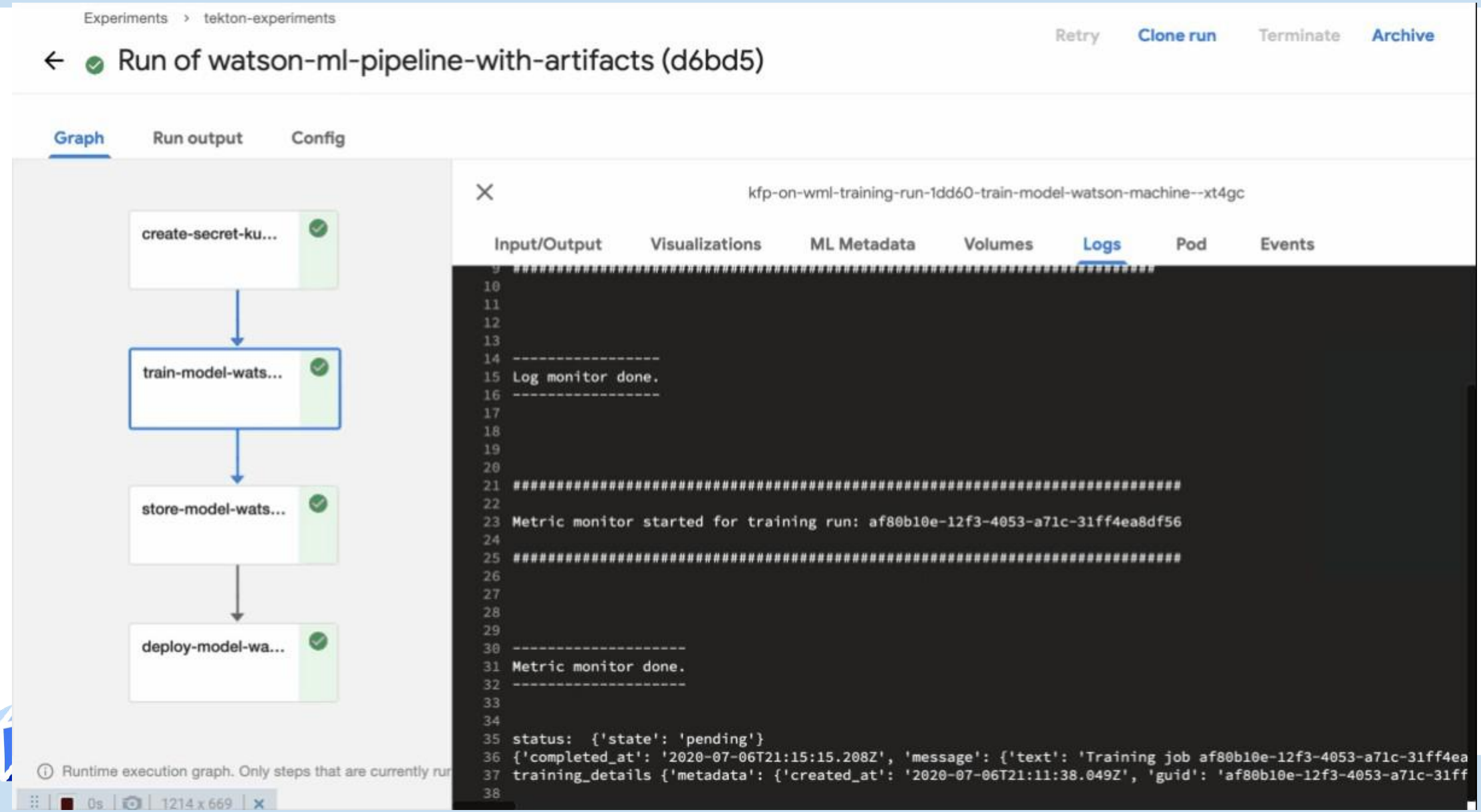
Experiment name	Description	Last 5 runs
Default	All runs created without specifying an experiment will be grouped here.	

Run name	Status	Duration	Pipeline Version	Recurring Run	Start time
sample-job	?	-	tekton-parameters	-	5/21/2020, 5:07:46 PM

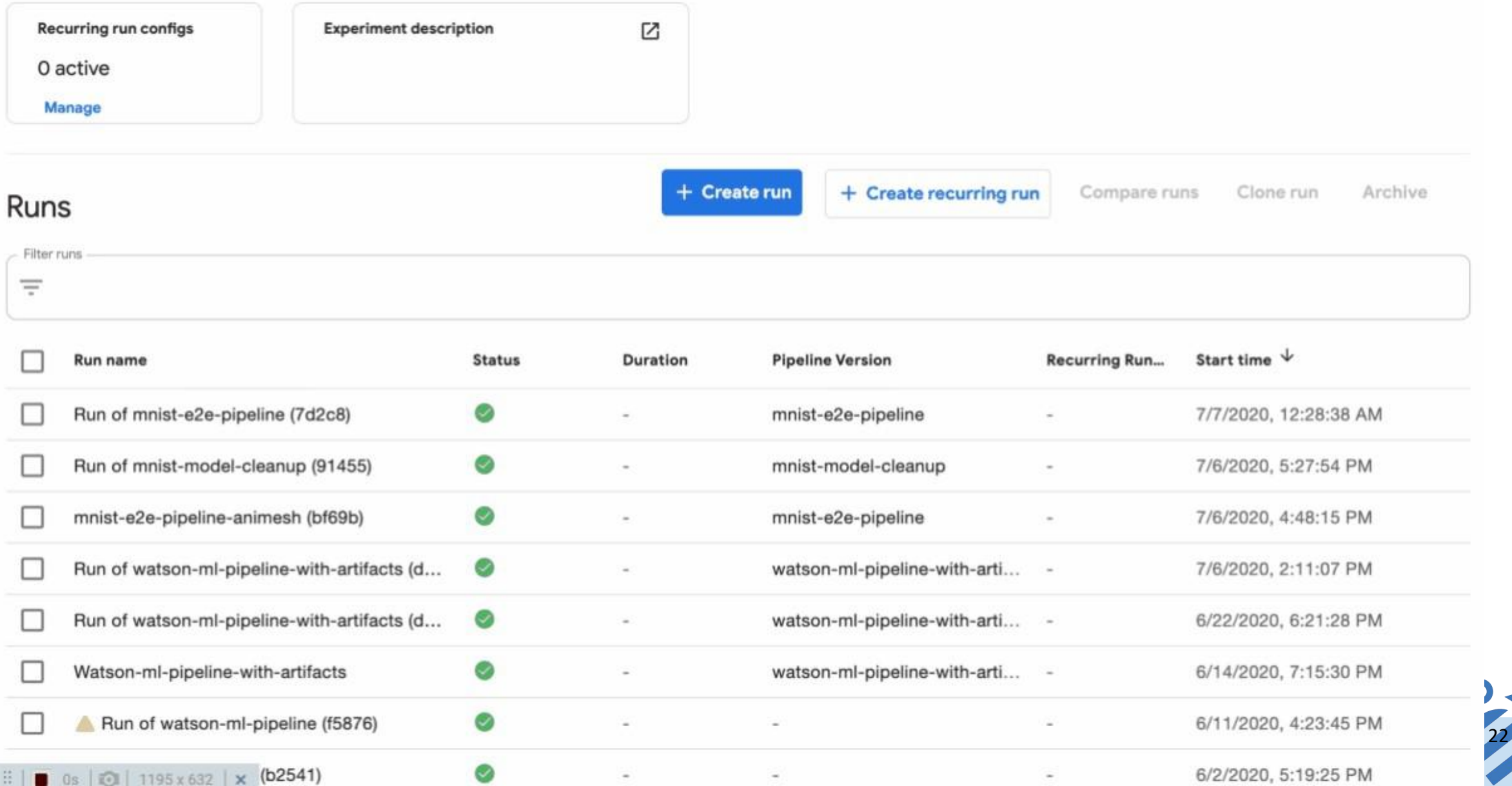
Same KFP Experience: DAG, backed by Tekton YAML



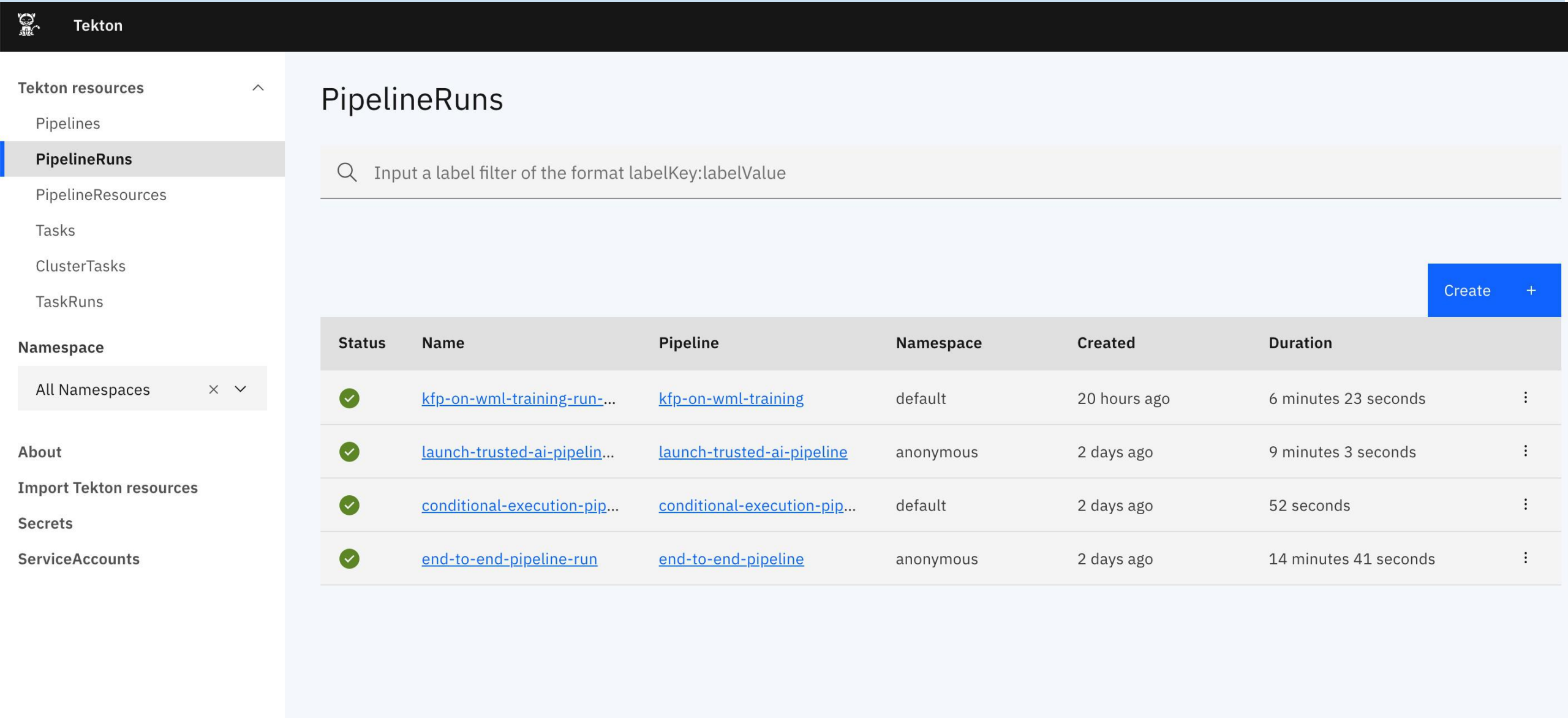
Same KFP Exp: Logs, Lineage Tracking and Artifact Tracking



End to end Kubeflow Components : With KFP-Tekton



Compiled Pipelines on Tekton



Running Pipelines on Tekton

Tekton

Tekton resources

- Pipelines
- **PipelineRuns**
- PipelineResources
- Tasks
- ClusterTasks
- TaskRuns

Namespace

default

About

Import Tekton resources

Secrets

ServiceAccounts

kfp-on-wml-training-run-p7n6f 20 hours ago

Rerun

Succeeded Tasks Completed: 4, Skipped: 0

- create-secret-kubernete...
- train-model-watson-mac...
  - train-mode... Completed
- store-model-watson-ma...
- deploy-model-watson-m...

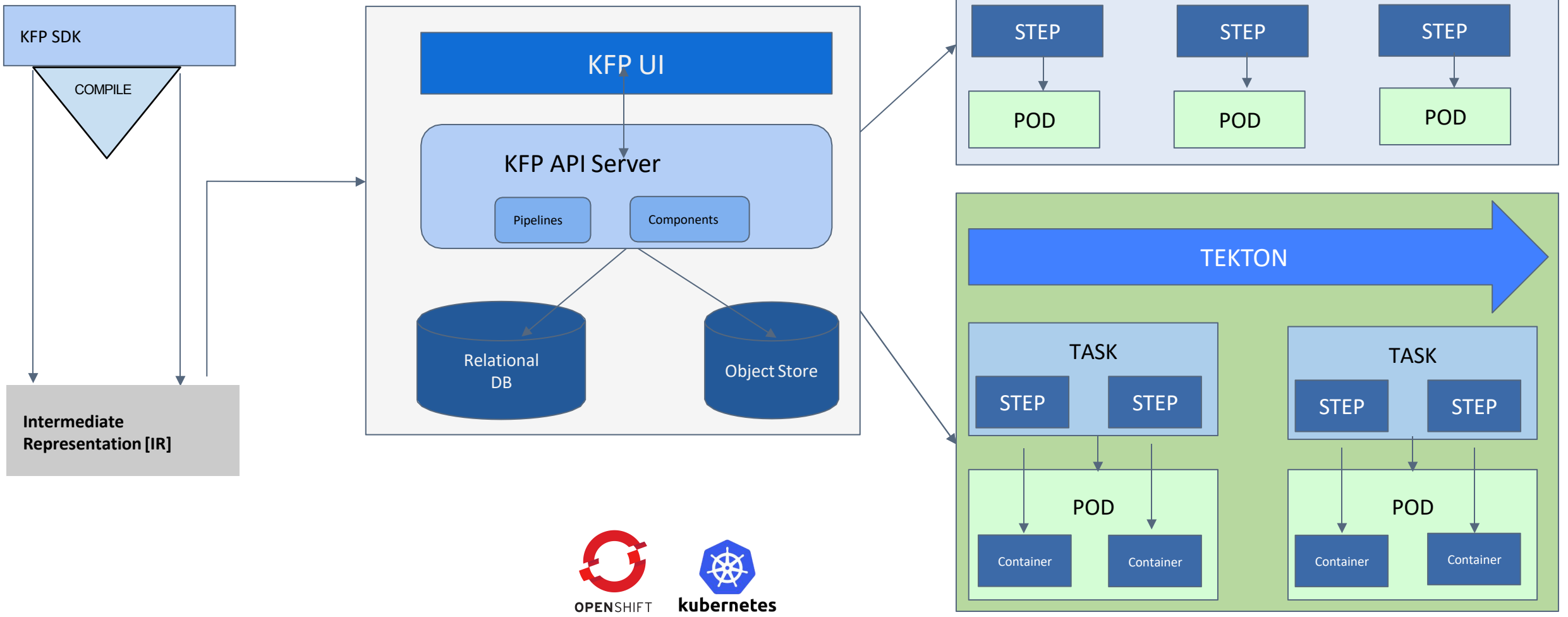
train-model-watson-machine-learning Completed

Logs | Status | Details

```
training_id {'metadata': {'created_at': '2020-05-07T23:57:46.868Z', 'guid': 'b200eef4-3dde-4b4e-a521-fe751735932c' ,
get status {'state': 'running'}

##################################################################################

Log monitor started for training run: b200eef4-3dde-4b4e-a521-fe751735932c

##################################################################################



-----------------
Log monitor done.
-----------------



##################################################################################

Metric monitor started for training run: b200eef4-3dde-4b4e-a521-fe751735932c
```

DEMO

Future: KFP – Tekton Phase Three



Pluggable Components

Spark | Watson Studio | WML | Open Scale | Kubeflow Training | Seldon | AIF360 | ART | KATIB | KFSERVING | ...

Useful Links

Main Open Source Github Repository:
https://github.com/kubeflow/kfp-tekton

IBM internal Slack channels #kfp-tekton
#kubeflow

The Kubeflow external Slack workspace is kubeflow.slack.com

To join, click here
https://join.slack.com/t/kubeflow/shared_invite/zt-cpr020z4-PfcAue_2nw67~iIDy7maAQ