

DATA+AI
SUMMIT 2022

Build Metadata and Lineage Driven Pipelines in Kubernetes



Yi-Hong Wang
Software Developer, IBM



Tommy Li
Software Developer, IBM

ORGANIZED BY  databricks

Agenda

- Introduce Kubeflow Pipelines
- Metadata in Kubeflow Pipelines
- Benefits of Metadata and lineage driven pipeline
- Metadata enhancement in Kubeflow Pipelines v2
- Abstraction Layer to support agnostic backends
- Summary

Enterprises are still struggling to scale AI beyond experimentation

88% of corporate AI initiatives are struggling to move beyond test stages

Source: Artificial Intelligence, The Next Digital Frontier. McKinsey Global Institute, 2017

Client Quotables

"I have no quantification of the business impact of my AI solutions"

"My data scientists have developed some models, but I do not know if they always achieve the best possible solution"

"I have an analytics team that has executed multiple PoCs, but none of that has made it into production"

"We've deployed multiple algorithms, but we have not seen any improvement in our business KPIs"

"We find it difficult finding and hiring the right AI talent"

"My business users **do not trust** the results of my AI applications, and they do not get used"

How to deliver AI at Scale...in Production

Best practices for building accurate models are well understood...



... but less so for building productive Data Science solution at scale.

Holistic Architecture

Application Logic
Technical Integration
Model Management
Tracing, Logging, Metrics

Effective Engineering

Standards
Pipelines
Automation

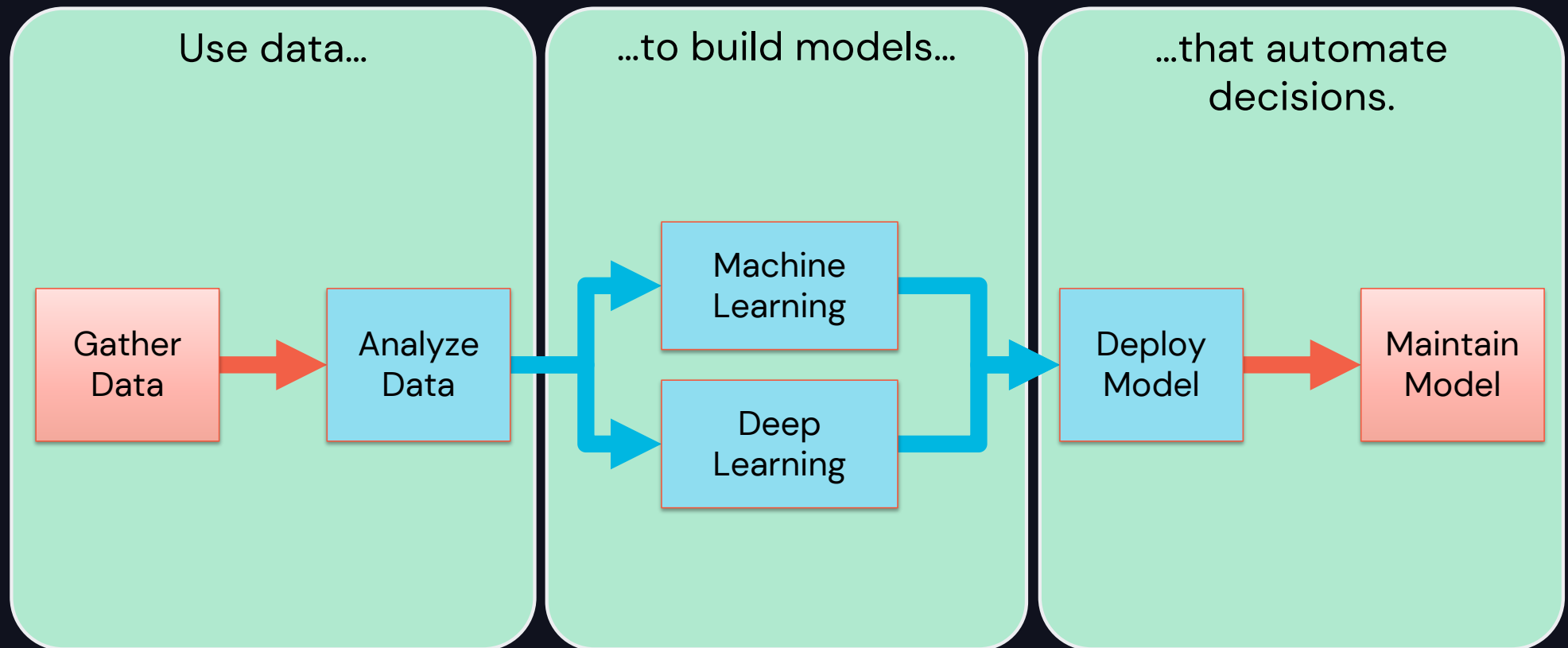
Smooth Operations

Technical Monitoring
Model Monitoring
Maintenance Strategy

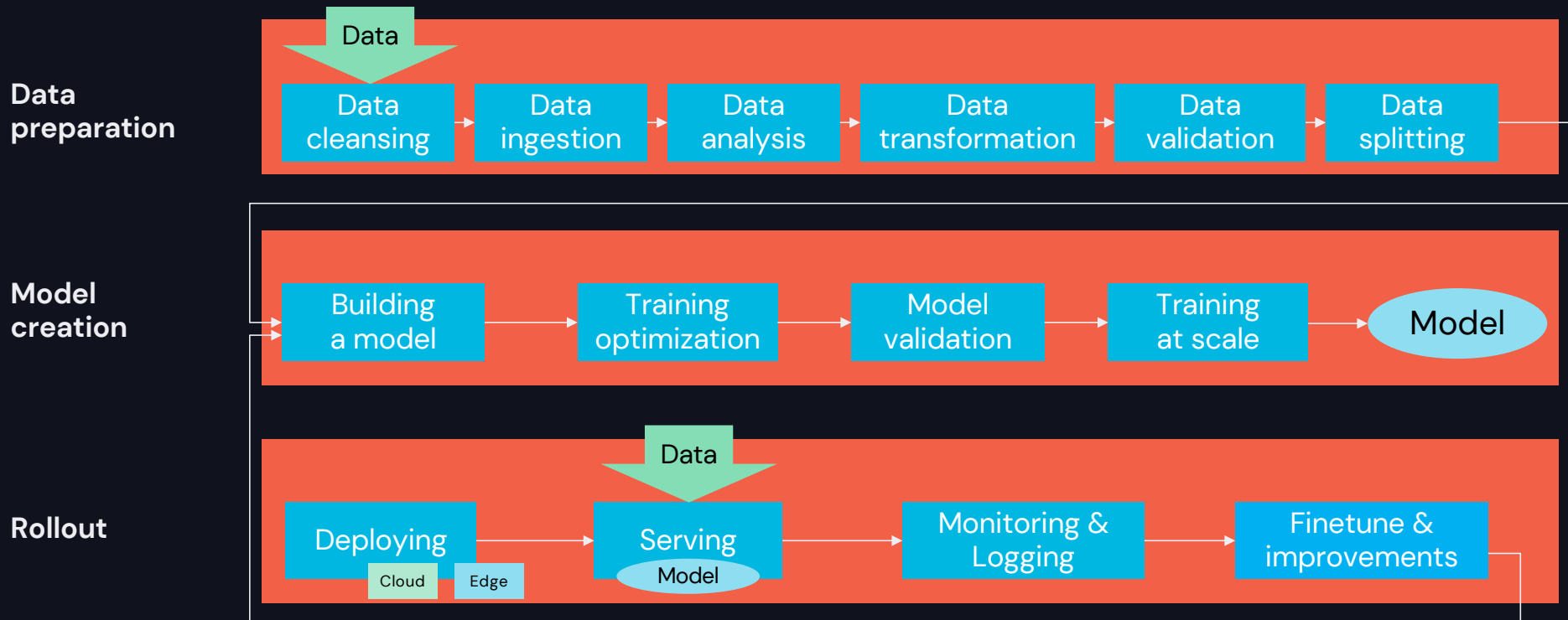
High-Performing Team

Targeted Project Approach

Pillars of AI Lifecycle – Datasets, Models...



...and Pipelines



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

DATA+AI
SUMMIT 2022

The screenshot displays the Kubeflow Pipelines interface. At the top, there are buttons for '+ Create run', '+ Upload version', '+ Create experiment', and 'Delete'. Below this, a pipeline graph is shown for '[Demo] TFX - Taxi Tip Prediction Model Trainer ([Demo] TFX - ...'. The graph includes components like 'csvexamplegen', 'statisticsgen', 'schemagen', 'examplevalidator', 'transform', 'trainer', 'evaluator', 'modelvalidator', and 'pusher'. Below the graph, there is a table listing various pipelines.

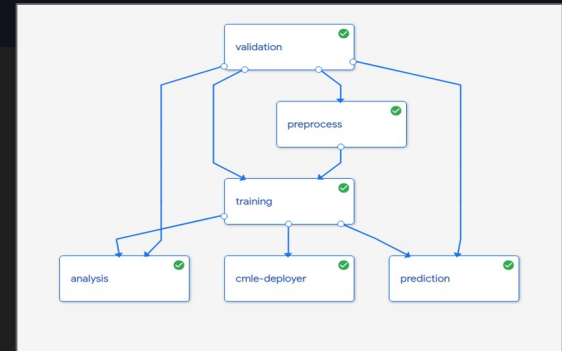
Pipeline name	Description	Uploaded on
[Sample] Basic - Condition	A pipeline shows how to use dsl.Condition. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/core/condition/condition.py	02/01/2019, 11:24:37
[Sample] Basic - Exit Handler	A pipeline that downloads a message and print it out. Exit Handler will run at the end. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/core/exit_handler/exit_handler.py	02/01/2019, 11:24:36
[Sample] Basic - Immediate	A pipeline with parameter values hard coded. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/core/immediate/immediate.py	02/01/2019, 11:24:34
[Sample] Basic - Parallel Join	A pipeline that downloads two messages in parallel and print the concatenated result. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/core/parallel_join/parallel_join.py	02/01/2019, 11:24:33
[Sample] Basic - Sequential	A pipeline with two sequential steps. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/core/sequential/sequential.py	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 BLI dataset. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/advanced/taxi_tip/taxi_tip.py	02/01/2019, 11:24:30
[Sample] ML - XGBoost - Train	A trainer that does end-to-end distributed training for XGBoost models. For source code, refer to https://github.com/kubeflow/pipelines/blob/master/samples/advanced/xgboost_train/xgboost_train.py	02/01/2019, 11:24:29

Define Pipeline with Python SDK

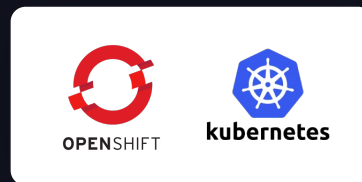
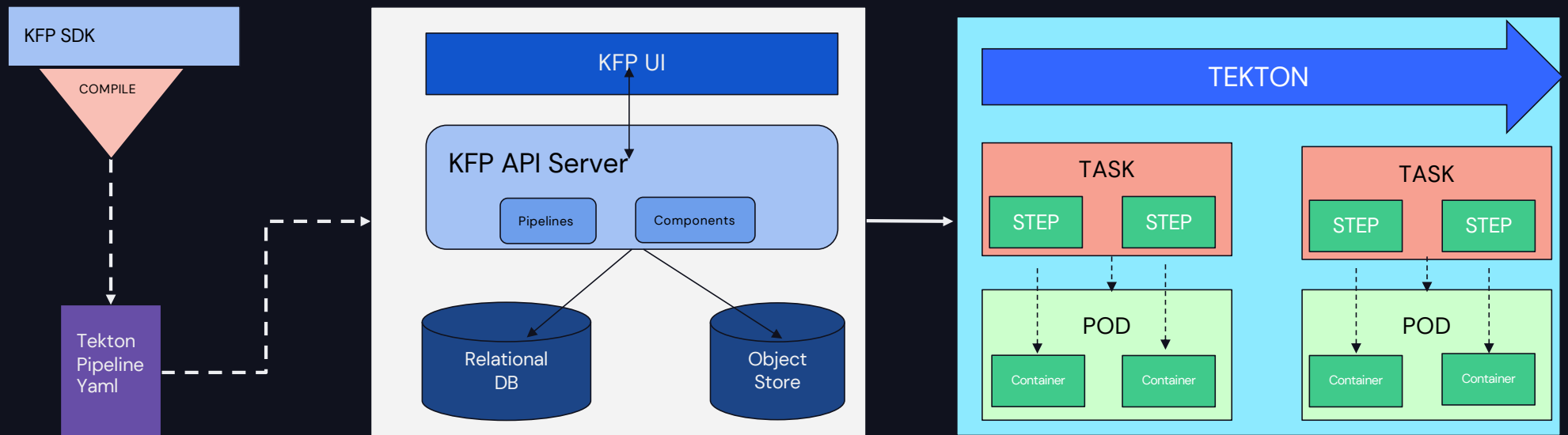
```
@dsl.pipeline(name='taxi-cab-classification-pipeline-example')
def taxi_cab_classification(
    output_dir: str, project: str,
    train_data: str = 'gs://bucket/train.csv',
    evaluation_data: str = 'gs://bucket/eval.csv',
    target: str = 'tips',
    learning_rate: int = 0.1, hidden_layer_size: str = '100,50', steps: int = 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)
```

```
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'})
```



Kubeflow Pipelines with Tekton hits v1.0



Pluggable Components

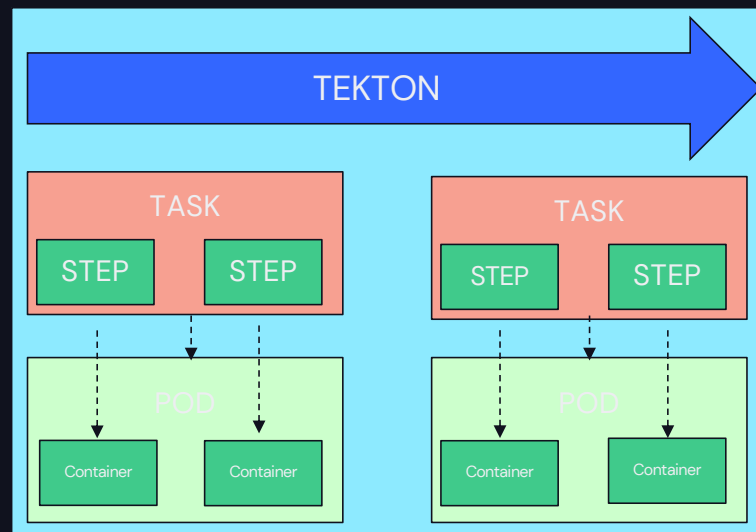


DATA+AI
SUMMIT 2022

<https://developer.ibm.com/blogs/kubeflow-pipelines-and-tekton-advances-data-workloads/>

Tekton

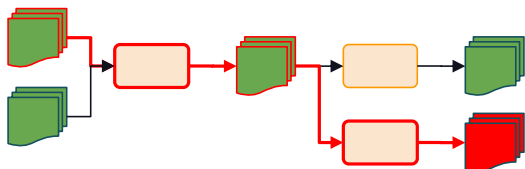
- ❑ The Tekton Pipelines project provides Kubernetes-style resources for declaring CI/CD-style pipelines.
- ❑ Tekton introduces several new CRDs including Task, Pipeline, TaskRun, and PipelineRun.
- ❑ A PipelineRun represents a single running instance of a Pipeline and is responsible for creating a Pod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



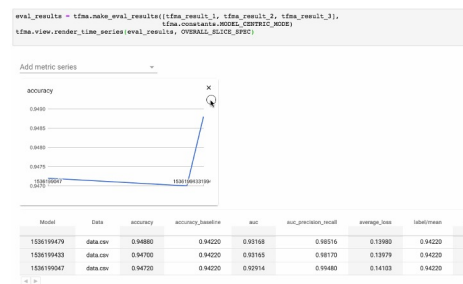
- ❑ A **PipelineRun** defines an execution of a pipeline. It references the Pipeline to run.
- ❑ A **Pipeline** defines the set of Tasks that compose a pipeline.
- ❑ A **TaskRun** defines an execution of a task. It references the task to run.
- ❑ A **Task** defines a set of build Steps such as compiling code, running tests, and building and deploying images.

Benefits of metadata and artifact tracking

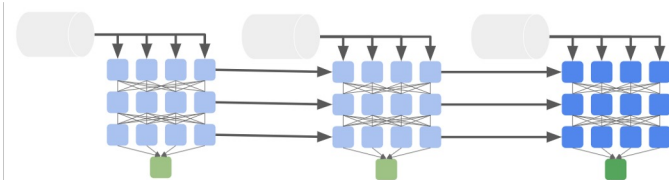
Find out which data a model was trained on



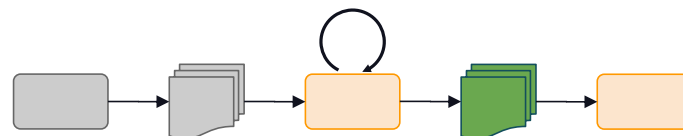
Compare previous model runs



Carry-over state from previous models



Re-use previously computed outputs



Artifact Tracking

The screenshot displays the Kubeflow dashboard interface. On the left is a dark blue sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, and Executions. The main content area is white and shows the 'Artifacts' section for a user named 'user1 (Owner)'. A breadcrumb trail shows '← model'. Below this are two tabs: 'Overview' (selected) and 'Lineage Explorer'. The 'Overview' tab displays the artifact name 'XGBoostModel' and its URI: 'minio://mlpipeline/artifacts/kaggle-house-price-56c0c/train-data/model.tgz'. Under the 'Properties' section, 'Custom Properties' are shown as a JSON object:

```
{  "name": "train-data-model",  "path": "/tmp/outputs/model/data",  "s3": {    "bucket": "mlpipeline",    "key": "artifacts/kaggle-house-price-56c0c/train-data/model.tgz"  }}
```

. At the bottom, a table lists the pipeline name as 'kaggle-house-price-56c0c' and the run ID as 'kaggle-house-price-56c0c'.

user1 (Owner) ▾

Artifacts

← model

Overview Lineage Explorer

XGBoostModel

URI
minio://mlpipeline/artifacts/kaggle-house-price-56c0c/train-data/model.tgz

Properties

Custom Properties

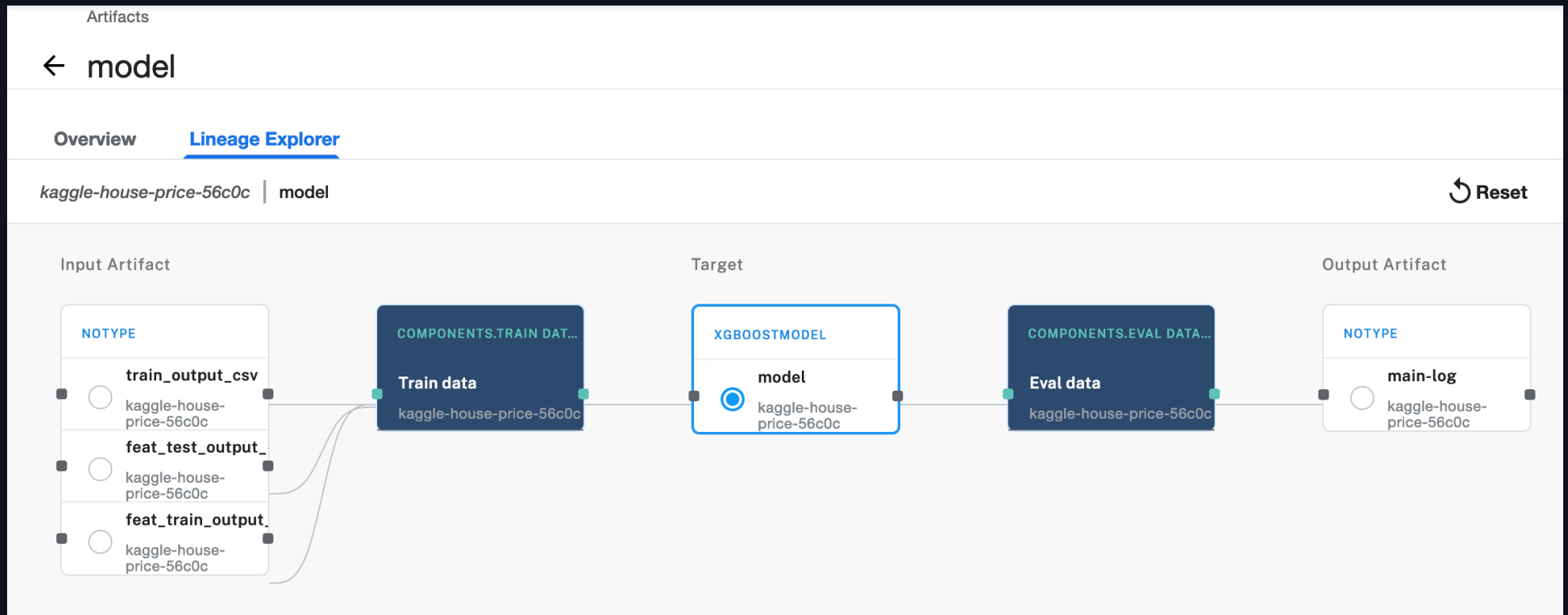
```
argo_artifact                                     name
{
  "name": "train-data-model",
  "path": "/tmp/outputs/model/data",
  "s3": {
    "bucket": "mlpipeline",
    "key": "artifacts/kaggle-house-price-56c0c/train-data/model.tgz"
  }
}
```

pipeline_name	run_id
kaggle-house-price-56c0c	kaggle-house-price-56c0c

Artifacts for a run of the “Kaggle House Price” 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.

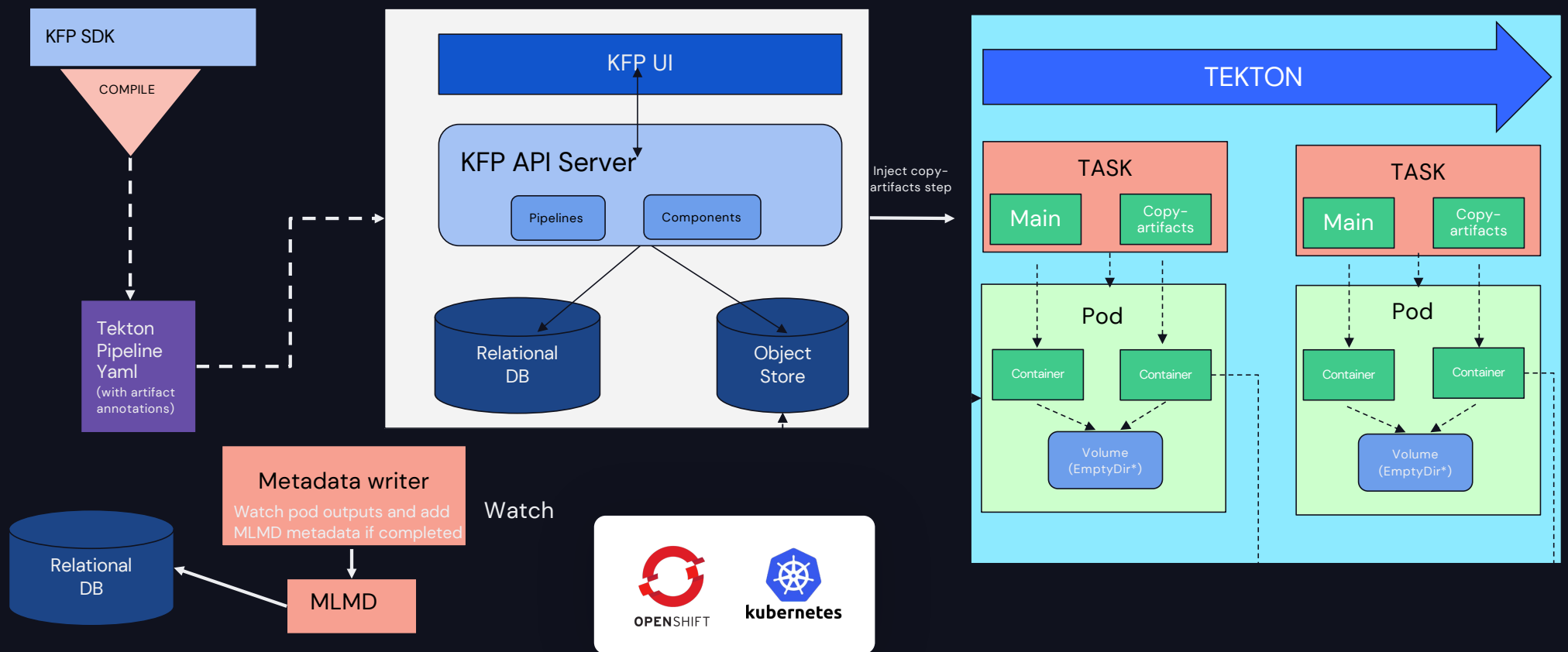


Kubeflow Pipelines on Tekton: Logs, Lineage Tracking and Artifact Tracking

The screenshot displays the Kubeflow Pipelines interface. At the top, the breadcrumb navigation shows 'Experiments > tekton-experiments'. The main header for the selected run is 'Run of watson-ml-pipeline-with-artifacts (d6bd5)', accompanied by action buttons: 'Retry', 'Clone run', 'Terminate', and 'Archive'. Below the header, three tabs are visible: 'Graph', 'Run output', and 'Config'. The 'Graph' tab is active, showing a vertical flow of four steps: 'create-secret-ku...', 'train-model-wats...', 'store-model-wats...', and 'deploy-model-wa...'. Each step is marked with a green checkmark, indicating successful completion. To the right, a modal window titled 'kfp-on-wml-training-run-1dd60-train-model-watson-machine--xt4gc' is open, displaying the 'Logs' tab. The log content includes a separator line, the message 'Log monitor done.', another separator line, a detailed message 'Metric monitor started for training run: af80b10e-12f3-4053-a71c-31ff4ea8df56', and a final 'Metric monitor done.' message. At the bottom of the logs, a JSON object provides status and metadata:

```
status: {'state': 'pending'}  
{'completed_at': '2020-07-06T21:15:15.208Z', 'message': {'text': 'Training job af80b10e-12f3-4053-a71c-31ff4ea8df56'}, 'training_details': {'metadata': {'created_at': '2020-07-06T21:11:38.049Z', 'guid': 'af80b10e-12f3-4053-a71c-31ff4ea8df56'}}}
```

Kubeflow Pipelines with Tekton: Metadata and Artifact tracking



Pluggable Components

DATA+AI
SUMMIT 2022

Spark

Watson
Studio

WML

Open
Scale

Kubeflow
Training

Seldon

AIF360

ART

KATIB

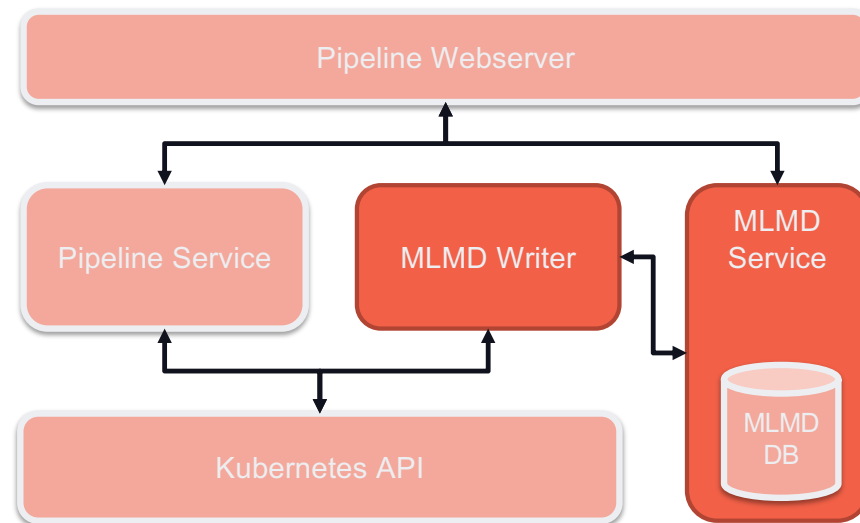
KSERVE

Kubeflow Pipelines v2

Machine Learning Metadata in v1

MLMD service + MLMD writer

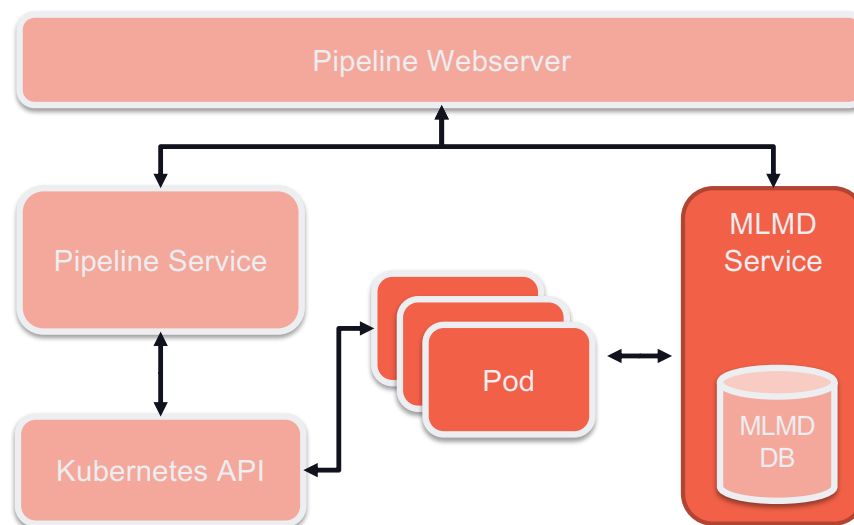
- Asynchronous process
- Preliminary data
- No way to use MLMD to do data passing for Pipeline Run



Machine Learning Metadata in v2

Integrate MLMD with pipeline execution natively

- Integrate MLMD into pipeline execution
- Extend metadata, including pipeline status, parameters, artifacts, etc.
- Use MLMD in pipeline tasks
- Caching key calculation
- Source of truth of the Pipeline Run UI



Pipeline Spec in v1

Platform-dependent Pipeline Spec

- SDK generates Argo/Tekton YAML
- Pipeline UI only understands specific CR

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  annotations:
    pipelines.kubeflow.org/pipeline_spec: '{"description": "Get Most Frequent Word and Save to GCS", "inputs": "generateName: save-most-frequent"'
spec:
  arguments:
    parameters:
      - name: message
      - name: outputpath
  entrypoint: save-most-frequent
  serviceAccountName: pipeline-runner
  onExit: exiting
  templates:
    - dag:
        tasks:
          - arguments:
              parameters:
                - name: message
                  value: '{{inputs.parameters.message}}'
              name: get-frequent
              template: get-frequent
```

Workflow

```
apiVersion: tekton.dev/v1beta1
kind: PipelineRun
metadata:
  name: save-most-frequent
  annotations:
    tekton.dev/output_artifacts: '{"get-frequent": [{"key": "artifacts/$PIPELINERUN/get-frequent/word.tgz", "name": "get-frequent-word", "path": "/tmp/message.txt"}]}'
    tekton.dev/input_artifacts: '{"save": [{"name": "get-frequent-word", "parent_task": "get-frequent"}]}'
    tekton.dev/artifact_bucket: mlpipeline
    tekton.dev/artifact_endpoint: minio-service.kubeflow:9000
    tekton.dev/artifact_endpoint_scheme: http://
    tekton.dev/artifact_items: '{"exiting": [], "get-frequent": [{"word", "${results.word.path}"], "save": []}]'
    sidecar.istio.io/inject: "false"
    pipelines.kubeflow.org/big_data_passing_format: ${workspaces.$TASK_NAME.path}/artifacts/$ORIG_PR_NAME/$TA
    pipelines.kubeflow.org/pipeline_spec: '{"description": "Get Most Frequent Word and Save to GCS", "inputs": [{"name": "message", "type": "String"}, {"name": "outputpath", "type": "String"}], "name": "Save Most Frequent"}'
spec:
  params:
    - name: message
      value: ''
    - name: outputpath
      value: ''
  pipelineSpec:
    params:
```

PipelineRun

Intermediate Representation in v2

Agnostic Pipeline Spec

- SDK generates IR in YAML format
- Easy to interpret
- Speed up Low Code/No Code integration

```
Pipelines
← kaggle-house-price (kaggle-house-price_version_at_2022-06-25T04:28:4

Graph Pipeline Spec
689 root:
690 dag:
691 tasks:
692   download-data:
693     taskInfo:
694       name: download-data
695     inputs:
696       parameters:
697         url:
698           runtimeValue:
699             constant: >-
700               https://github.com/NeoKish/examples/raw/master/house-prices-kaggle-competition/data.zip
701     cachingOptions:
702       enableCache: true
703     componentRef:
704       name: comp-download-data
705   eval-data:
706     taskInfo:
707       name: eval-data
708     inputs:
709       parameters:
710         model_path:
711           taskOutputParameter:
712             producerTask: train-data
713             outputParameterKey: model_path
714         test_data_path:
715           taskOutputParameter:
716             producerTask: featured-data
717             outputParameterKey: feat_test_output_csv
718     dependentTasks:
719       - featured-data
720       - train-data
721     cachingOptions:
722       enableCache: true
723     componentRef:
724       name: comp-eval-data
725   featured-data:
726     taskInfo:
727       name: featured-data
728     inputs:
729       parameters:
730         test_path:
731           taskOutputParameter:
732             producerTask: load-and-preprocess-data
```

New UX for v2

- New v2 IR
- Retrieve Pipeline Run information from MLMD

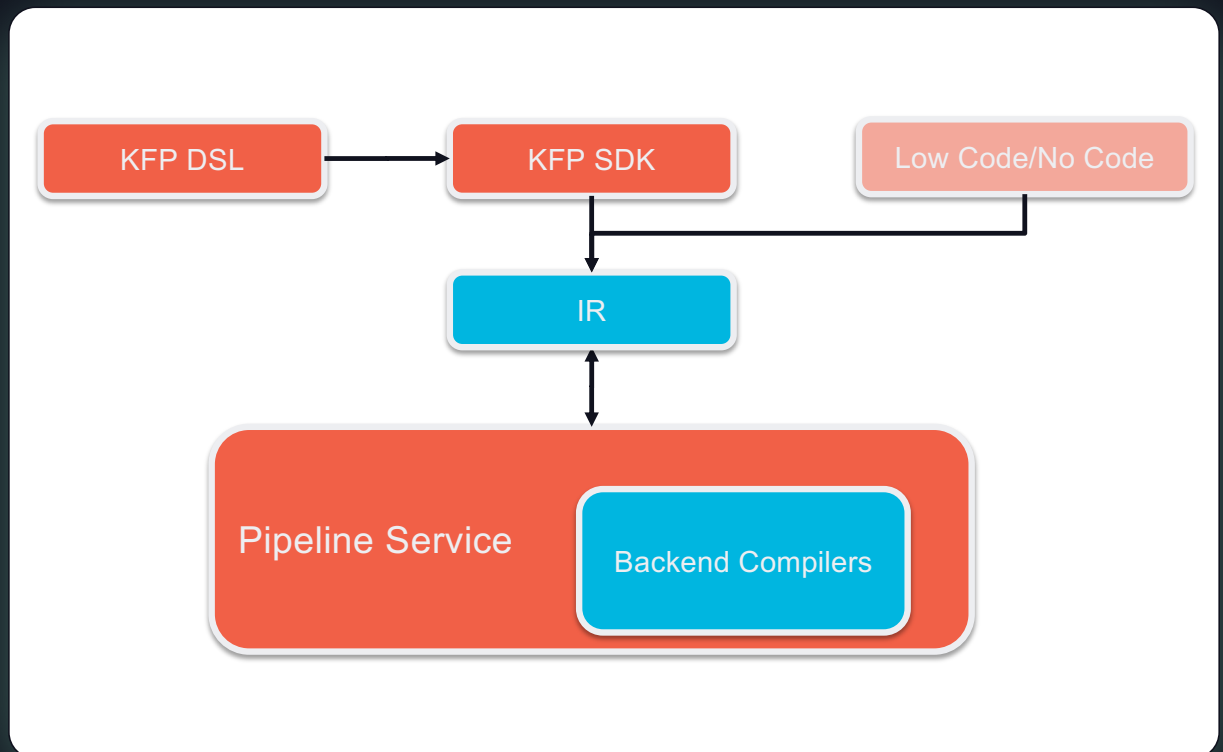


New Orchestration Controllers

Smart Compiler → Smart Runtime

Backend Compiler

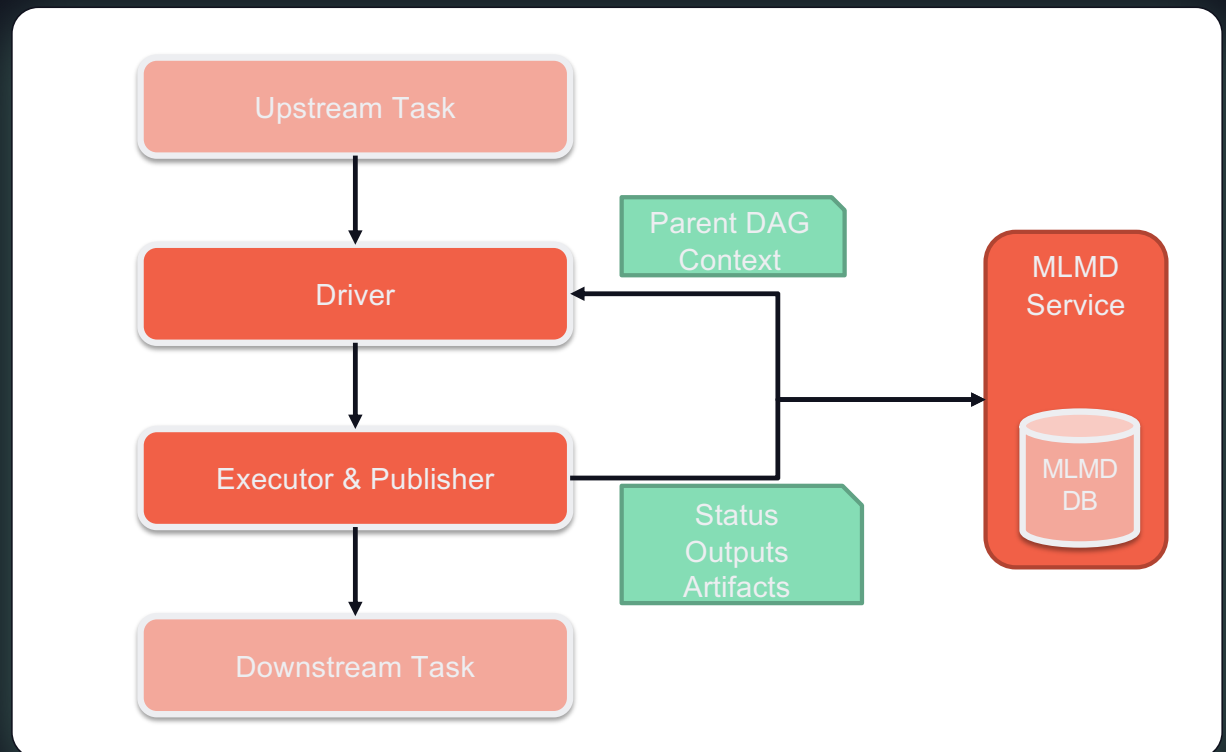
- Encapsulate backend engines
- Hide the platform specific CR from users



Smart Runtime

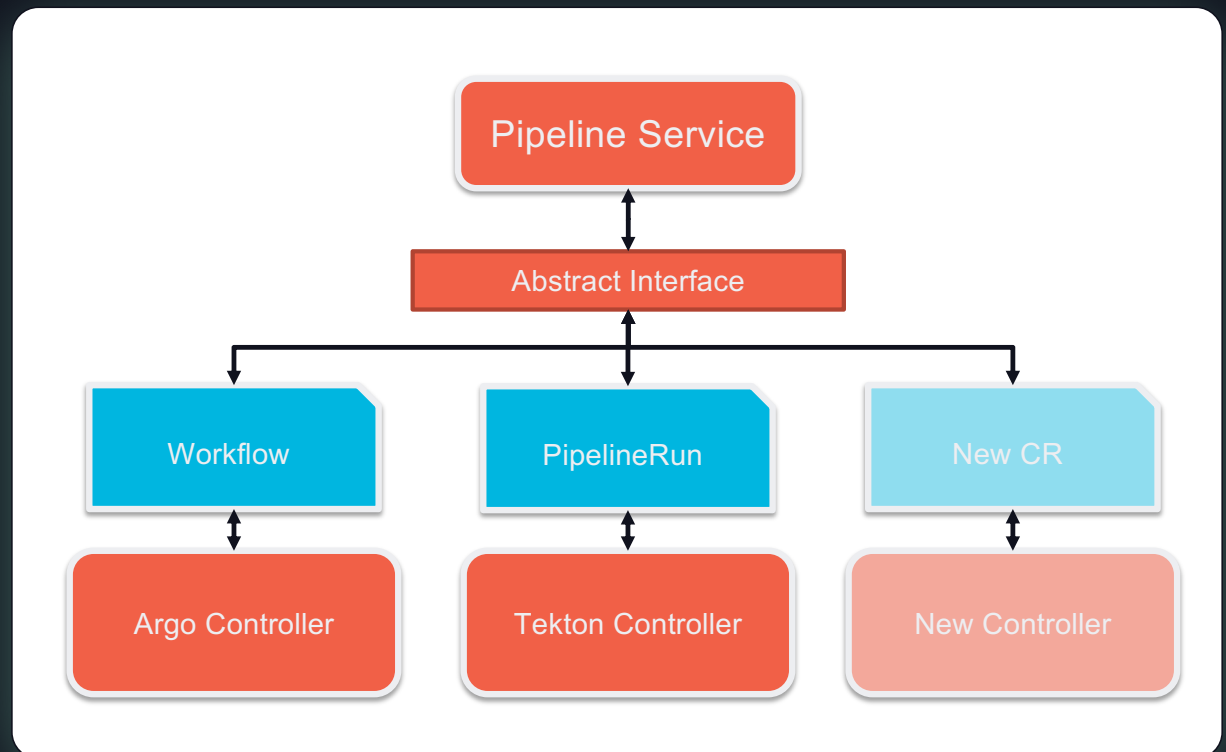
Driver/Executor/Publisher

- Gain more controls of the pipeline execution
- Easier to add features
- Natively integrate with MLMD



Abstraction Layer for Orchestration Engines

- Communicate with orchestration engines with single interface
- Expand the support to other orchestration engines



Components

Roadmap

Components

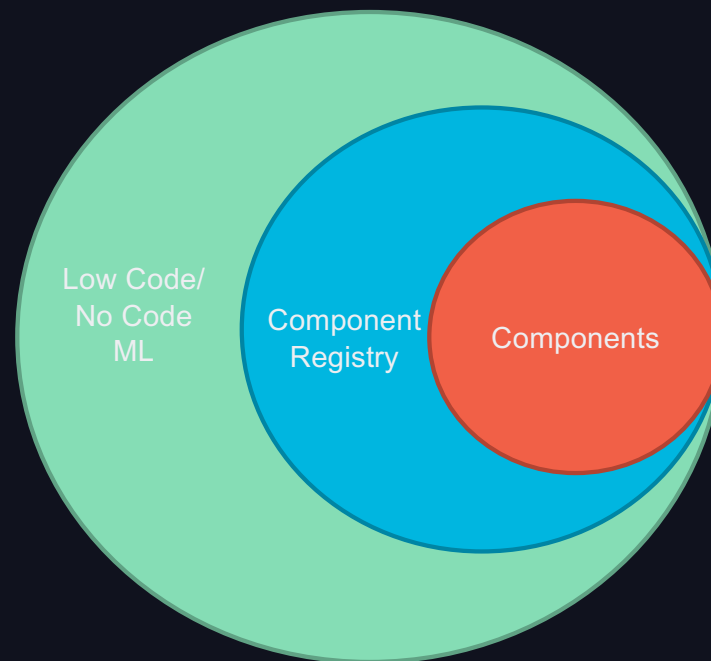
A rich set of components from community and vendors.

Component/Pipeline Registry

KFP SDK can directly load the components from the registry as long as it follows a standardized protocol.

Low code/No code ML

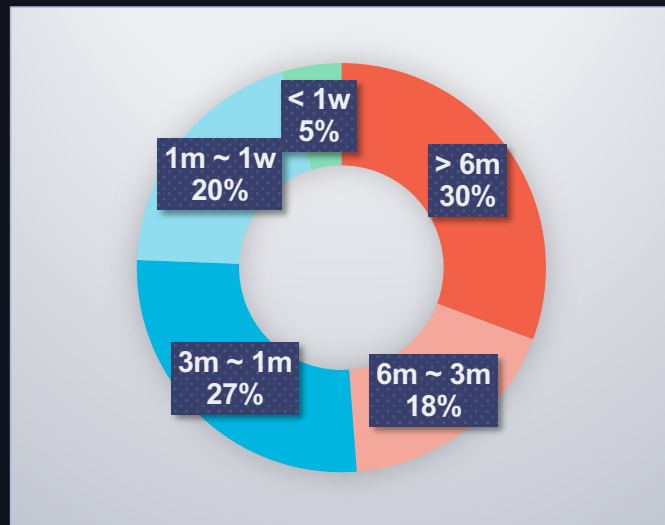
Create E2E ML workflow via drag-and-drop



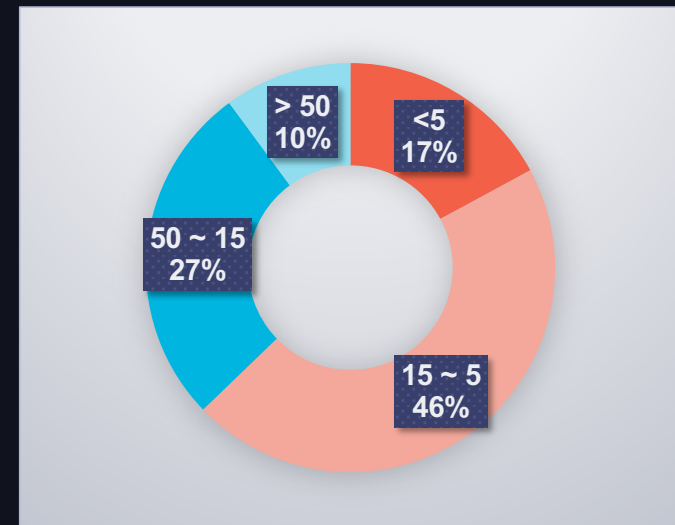
Challenges for ML

Kubeflow Community Survey

Average life of a model in production

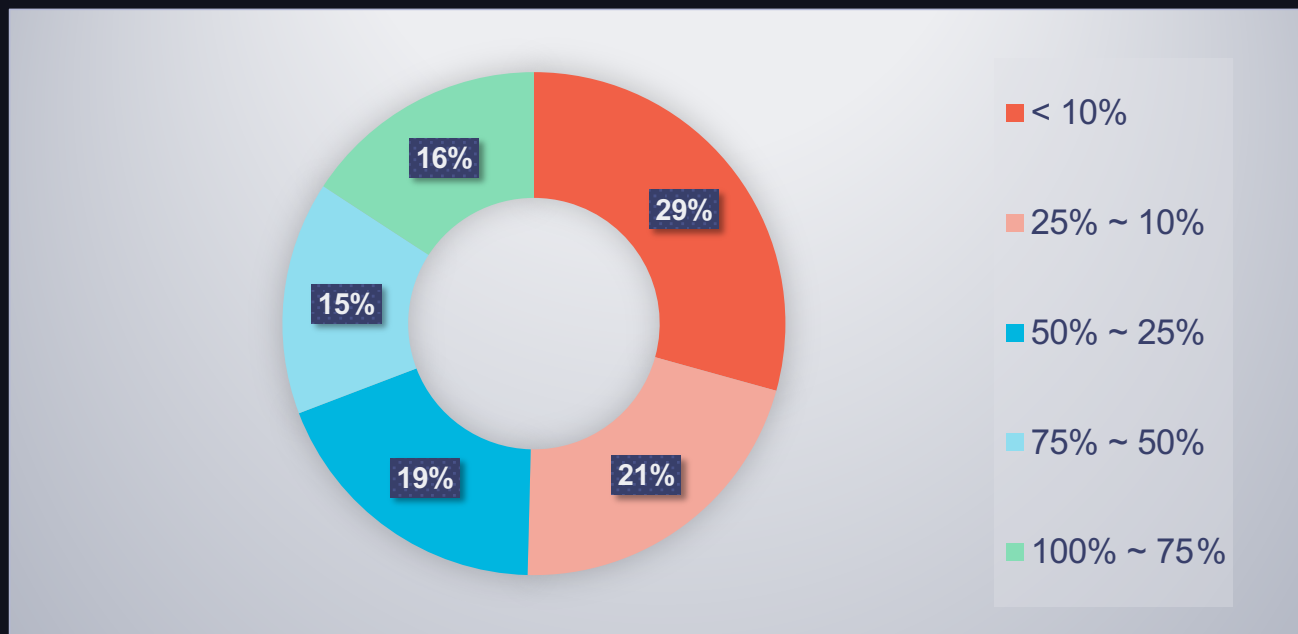


How many iterations does it take to produce a production model



Cont.

What percentage of your 2021 models were successfully deployed into production and are delivering business values?



Summary

- Embrace MLMD as first step toward MLOps
 - https://github.com/kubeflow/pipelines/tree/master/third_party/ml-metadata
- Use IR or component-based strategy to compose ML pipelines
 - <https://www.kubeflow.org/docs/components/pipelines/sdk-v2/component-development/>
- Leverage abstraction layer to bring your orchestration engine to Kubeflow Pipelines
 - https://github.com/kubeflow/pipelines/blob/master/backend/src/common/util/execution_spec.go#L51

References

- Kubeflow Pipelines
<https://github.com/kubeflow/pipelines/>
- Kubeflow Pipelines on Tekton
<https://github.com/kubeflow/kfp-tekton>
- Kubeflow Pipelines v2 Design
 - <http://bit.ly/kfp-v2>

DATA+AI
SUMMIT 2022

Thank you



Yi-Hong Wang
Software Developer



Tommy Li
Software Developer