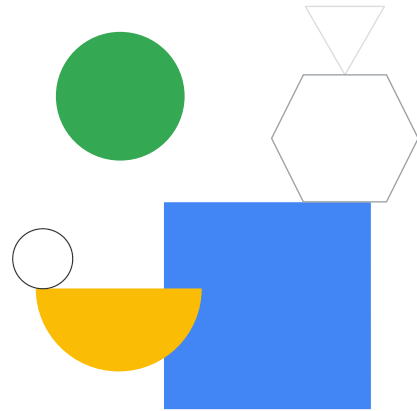




# Production ML Pipelines with Kubeflow



In a previous module, we leveraged pre-trained ML APIs to process natural text. These are great options for seeing if your use case can just use a model that's already created and trained on Google's data. But, you may want a more tailored model trained on your own data. For that we will need a custom model. Let's talk about the different ways of building custom models.



# Module agenda



- 01 Ways to do ML on Google Cloud
- 02 Kubeflow
- 03 AI Hub

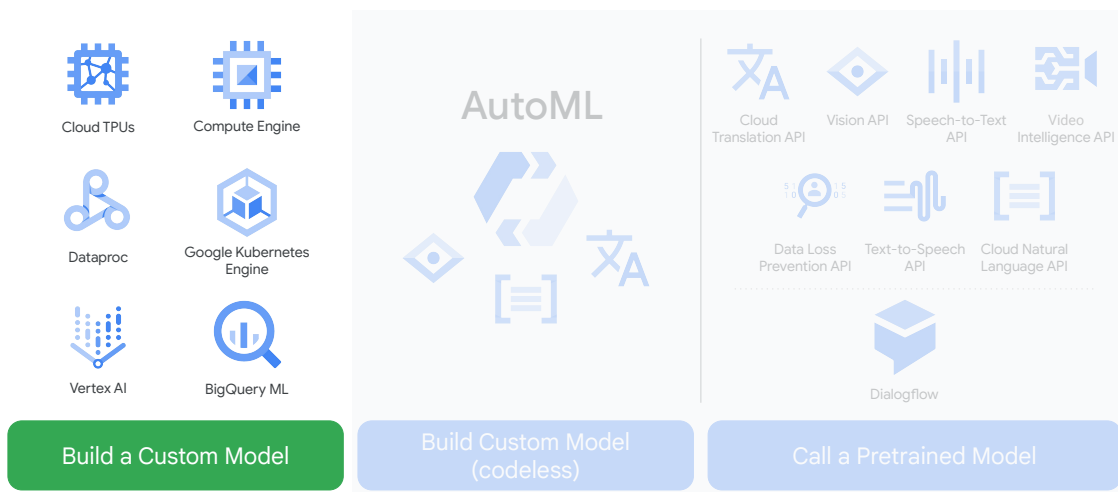
First, we will provide an overview of ways to do ML on Google Cloud. Then, we will talk about a tool, Kubeflow, for deploying machine learning models in a Kubernetes environment. Finally, we will discuss AI Hub, a repository of machine learning resources which can be made publicly available or available for only certain users.



## Ways to do ML on Google Cloud

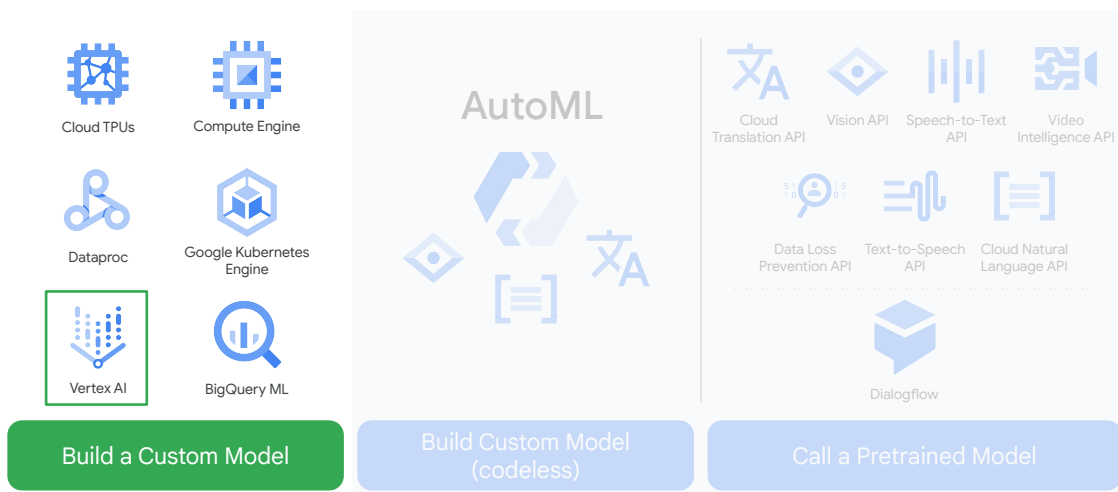
You've already learned that there are three ways you can do machine learning on Google Cloud.

# Create and deploy custom models with Kubeflow



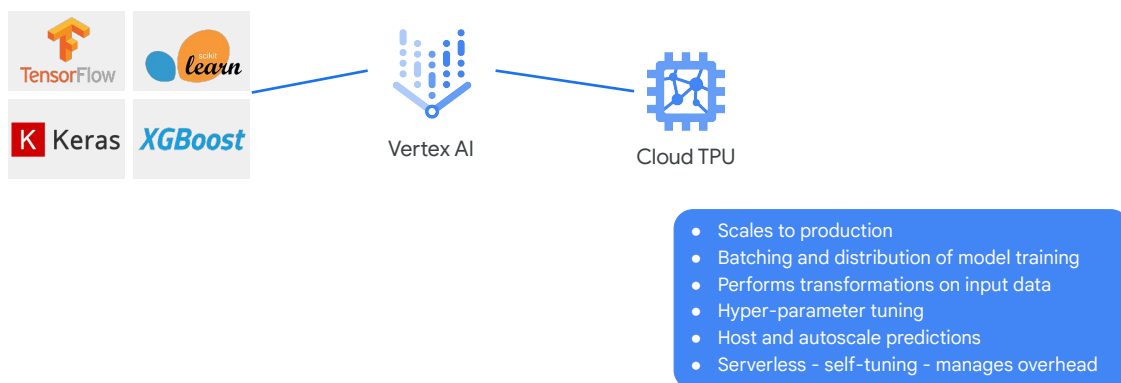
The pretrained models on the right have already been discussed. Now, we're going to visit the other side of the spectrum and build your own custom model and productionalize it on Google Cloud. There are a few ways of doing custom model development, training, and serving.

# Create and deploy custom models with Kubeflow



Let's discuss Vertex AI.

# Vertex AI is a fully managed service for custom machine learning models



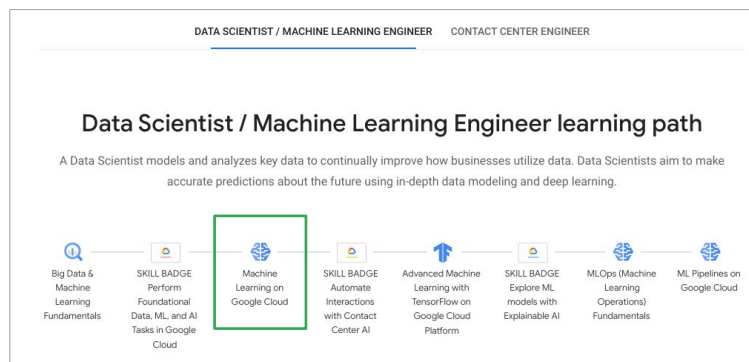
Google Cloud

What is Vertex AI exactly? It's a fully managed service for custom machine learning models, both training and serving predictions. It can scale from the experimentation stage all the way to production. You can also, using the features of TensorFlow, include transformations on input data and perform hyperparameter tuning to choose the best model for your case. You can deploy your models to Vertex AI to serve predictions, which will autoscale to the demands of your clients.

Vertex AI also supports Kubeflow, which is Google's open source framework for building ML pipelines -- and you'll have a lab on this later.

Essentially, Vertex AI is the engine behind doing machine learning at scale on Google Cloud. A data scientist can train and deploy production models from Notebooks with just a few commands.

# In this course, we don't cover writing TensorFlow models, only ways to operationalize them



## Google Cloud Training - Machine Learning and AI

Since we're using Vertex AI and Kubeflow, we will often be thinking about using TensorFlow models. However, this isn't the course to dive into the details of TensorFlow. You can learn more about this in the Machine Learning on Google Cloud course, which is part of the Machine Learning and AI learning path for Data Scientists and Machine Learning Engineers.

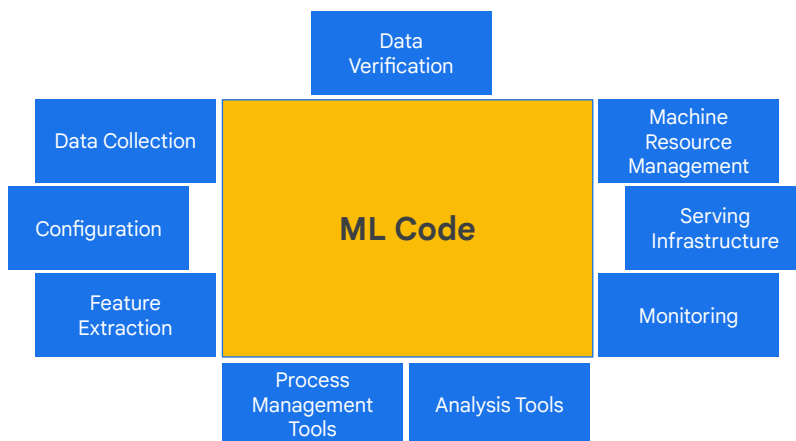


# Kubeflow

Where do Data Engineers come into the picture? Don't forget Data Engineers build data pipelines, and machine learning pipelines are no different. If we want to have a flexible pipeline for all stages of machine learning, Kubeflow is a great option.

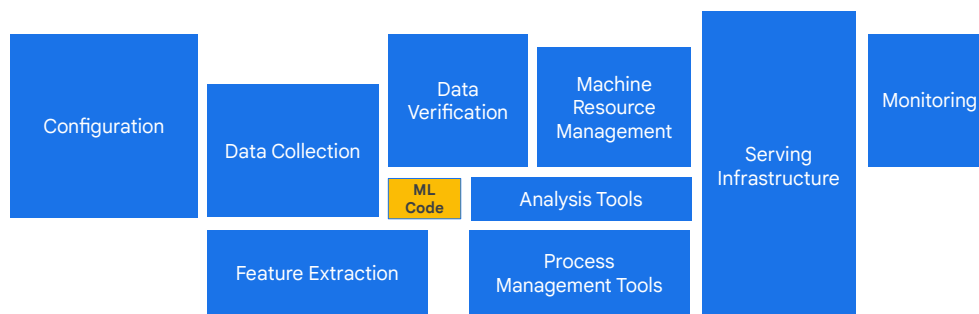


## Perception: ML products are mostly about ML



Many people think that machine learning products are all about the code that ML scientists write locally on their machines. Does this code ensure the data going into it is clean? Can the code auto-scale to clients who want to use it for serving predictions? What if we have to re-train the model, does it go off-line at that point?

## Reality: ML requires lots of DevOps



Source: [Sculley et al.: Hidden Technical Debt in Machine Learning Systems](#)

The truth is, production machine learning systems are large, complicated, distributed systems. There's a lot of DevOps involved for things like monitoring, and process management tools. Google started building Kubeflow to tackle these DevOps challenges using Kubernetes and containers.

## Kubeflow provides a platform for building ML products

- Leverage containers and Kubernetes to solve the challenges of building ML products.
- Kubeflow = Cloud Native, multi-cloud solution for ML.
- Kubeflow provides a platform for composable, portable and scalable ML pipelines.
- If you have a Kubernetes conformant cluster, you can run Kubeflow.

As mentioned, Kubeflow was built to leverage the strengths of Kubernetes. Kubeflow is a purpose-built, multi-cloud ML solution.

# Kubernetes is a great platform for ML

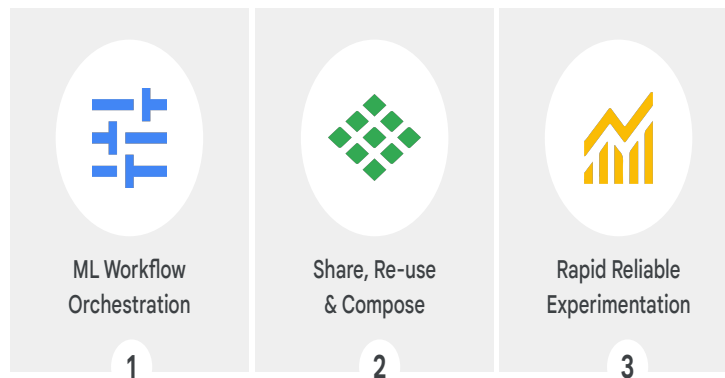
- Containers
- Scaling built in
- Unified architecture
- Easy to integrate building blocks
  - ML APIs
  - Dataflow
- Lots of options for CI/CD
- Portability
  - Dev, On-Prem, Multi-cloud: same stack



Before we dive into the specifics of Kubeflow Pipelines, I should provide additional context.

- Kubeflow Pipelines is a part of the open source project Kubeflow.
- Kubeflow is a platform that provides the tools and scalable services required to develop and deploy ML workloads, all the way from distributed training, to scalable serving, to Notebooks with JupyterHub and workflow orchestration and much more.
- Kubeflow services are built on top of Kubernetes. Kubernetes provides scalability and hybrid portability. You can run Kubeflow anywhere you can run a Kubernetes cluster, and thus applications built on Kubeflow are portable across clouds and on-premise environments. On Google Cloud, you can easily deploy Kubeflow on Google Kubernetes Engine.

The capabilities provided by Kubeflow pipelines can largely be put into three buckets

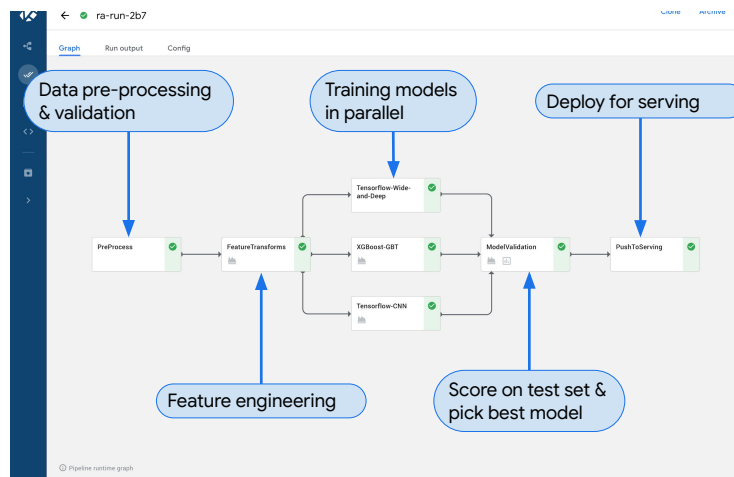


The capabilities provided by Kubeflow pipelines can largely be put into three buckets:

- ML Workflow Orchestration
- Share, Re-use & Compose
- Rapid Reliable Experimentation

You can think of the benefits as similar to those of Cloud Composer but better tailored for ML workloads. Let's see what a pipeline looks like.

## Visual depiction of pipeline topology

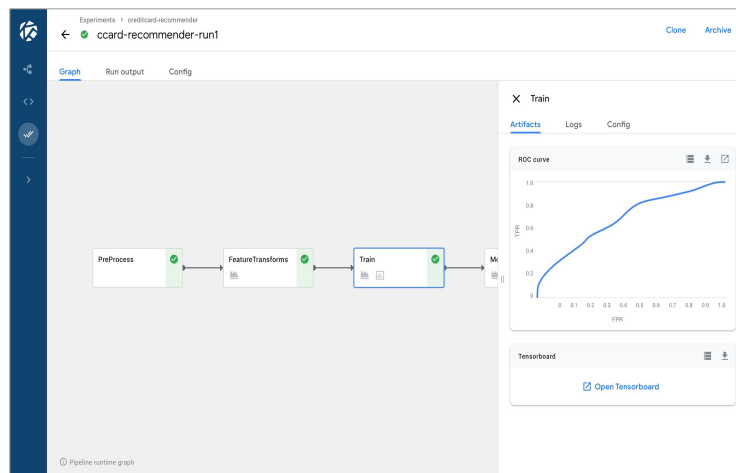


To make things more concrete let's look at a screenshot of an illustrative workflow that was run on Kubeflow Pipelines. This is just an illustrative workflow and users can author and run many different kinds of workflow topologies with different code and tools in the various steps of the workflow.

For each workflow that is run on Kubeflow Pipelines, you get a rich visual depiction of the topology so that you know what was executed as part of the workflow.

- In this workflow, we start with a **data preprocessing and validation** step.
- Followed by **feature engineering**.
- Following that step there is a fork where we **train many different kinds of models**.
- The models that are trained are then **analyzed and compared** on a test dataset.
- Finally, if an improved model is produced, it is **deployed** to a serving endpoint.

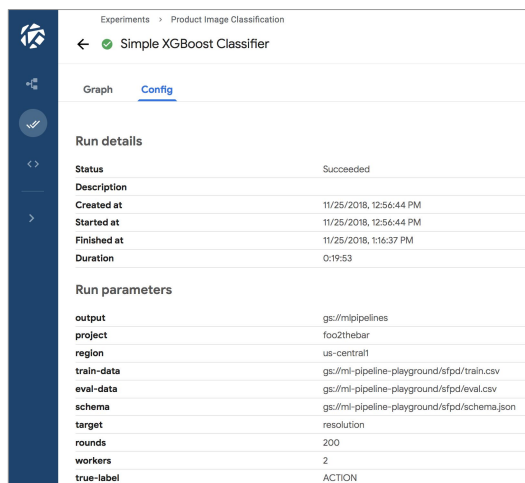
# Rich visualization of metrics



For each step of the workflow, you have rich ML-specific information at your fingertips. Just click on a step and visualize relevant metrics produced by that step, such as an ROC curve for example.

If you did model training, the rich metadata can be visualized with TensorBoard. It is just one click away.

## View all configs, inputs and outputs



The screenshot displays the 'Simple XGBoost Classifier' experiment configuration in the Google Cloud AI Platform console. The interface includes a sidebar with navigation icons and a main content area with tabs for 'Graph' and 'Config'. The 'Config' tab is active, showing 'Run details' and 'Run parameters'.

Run details	
Status	Succeeded
Description	
Created at	11/25/2018, 12:56:44 PM
Started at	11/25/2018, 12:56:44 PM
Finished at	11/25/2018, 1:16:37 PM
Duration	0:19:53

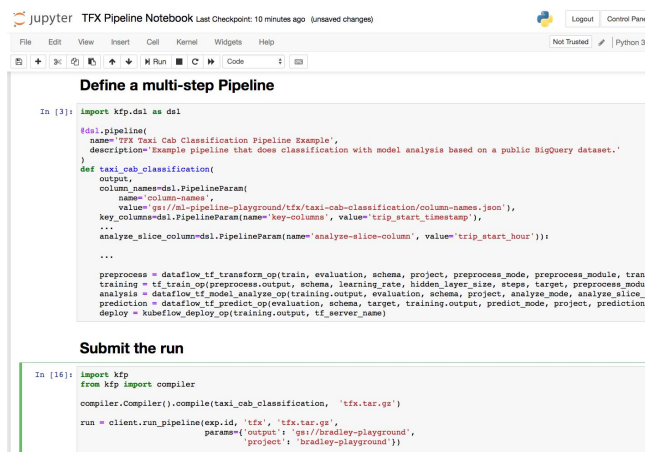
Run parameters	
output	gs://mlpipelines
project	foo2thebar
region	us-central1
train-data	gs://ml-pipeline-playground/tfcd/train.csv
eval-data	gs://ml-pipeline-playground/tfcd/eval.csv
schema	gs://ml-pipeline-playground/tfcd/schema.json
target	resolution
rounds	200
workers	2
true-label	ACTION

For each step of the workflow you can see the precise configuration parameters, inputs and outputs. Thus, for a model trained with Kubeflow Pipelines you never have to wonder, how exactly did I create this model?

Here you can quickly see how long the model training took, where the trained model is, and what data was used for training and evaluation.



# Author pipelines with an intuitive Python SDK



The screenshot shows a Jupyter Notebook interface with the title 'TFX Pipeline Notebook'. The notebook contains two code cells. The first cell, titled 'Define a multi-step Pipeline', defines a pipeline named 'TFX Taxi Cab Classification Pipeline Example' and a function 'taxi\_cab\_classification' that takes a dataset and returns a pipeline. The second cell, titled 'Submit the run', imports the 'kfp' and 'compiler' modules and uses the 'run\_pipeline' function to submit the pipeline to Google Cloud.

```

In [3]: import kfp, dsl

@dsl.pipeline(
    name='TFX Taxi Cab Classification Pipeline Example',
    description='Example pipeline that does classification with model analysis based on a public BigQuery dataset.'
)
def taxi_cab_classification(
    output,
    column_names=dsl.PipelineParam(
        name='column-names',
        value='gs://ml-pipeline-playground/tfx/taxi-cab-classification/column-names.json'),
    key_column=dsl.PipelineParam(name='key-column', value='trip_start_timestamp'),
    ...,
    analyze_slice_column=dsl.PipelineParam(name='analyze-slice-column', value='trip_start_hour')):
    ...

    preprocess = dataflow_tf_transform_op(train, evaluation, schema, project, preprocess_mode, preprocess_module, trans
    training = tf.train_op(preprocess.output, schema, learning_rate, hidden_layer_size, steps, target, preprocess_model)
    analysis = dataflow_tf_model_analyze_op(training.output, evaluation, schema, project, analyze_mode, analyze_slice_c
    prediction = dataflow_tf_predict_op(evaluation, schema, target, training.output, predict_mode, project, prediction
    deploy = kubeflow_deploy_op(training.output, tf_server_name)

In [16]: import kfp
from kfp import compiler

compiler.Compiler().compile(taxi_cab_classification, 'tfx.tar.gz')

run = client.run_pipeline(exp.id, 'tfx', 'tfx.tar.gz',
    param={ 'output': 'gs://bradley-playground',
            'project': 'bradley-playground' })
  
```

You can define the ML workflow using Kubeflow's Python SDK. By defining the workflow we mean specifying each step's inputs and outputs and how the various steps are connected. The topology of the workflow is implicitly defined by connecting the outputs of an upstream step to the inputs of a downstream step. You can also define looping constructs as well as conditional steps.

## Package and share pipelines as zip files

- Upload and execute pipelines via UI (in addition to API/SDK).
- Pipeline steps can be authored as reusable components.

The screenshot shows the 'Run details' form in the Kubeflow UI. It includes fields for 'Pipeline' (xgboost training - confusion matrix), 'Run name' (product-recommender-model), and 'Description/Command' (train XGB model for product recommendation application). Below these are 'Run parameters' for specifying parameters required by the pipeline, including 'project', 'region' (us-central1), 'train-data', 'eval-data', 'schema', 'hyper-parameters', 'epochs', 'workers', and 'train-label'. The 'train-label' field is set to 'ACTION'. At the bottom are 'Create' and 'Cancel' buttons.

Another nice Kubeflow feature is the ability to package pipeline components. This adds an element of portability since you can then move your ML pipelines even between cloud providers.

Kubeflow pipelines separate the work for different parts of the pipeline to enable people to specialize. For example, an ML engineer can focus on feature engineering rather than other parts of creating the model such as hyperparameter tuning. The ML engineer's solutions can then be bundled up and used by a data engineer as part of a data engineering solution. The solution can then appear as a service used by a data analyst to derive business insights.

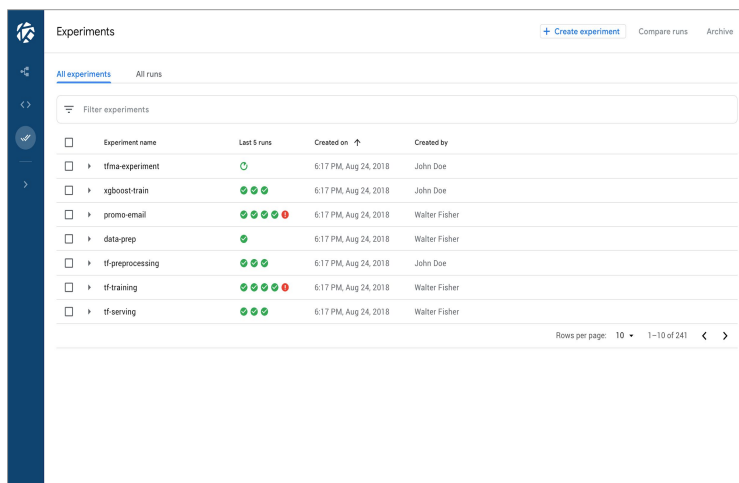
## Rapid, reliable, experimentation

- Every run logged with all config params, inputs, outputs and metrics.
- Easily search and find old runs.
- Clone and re-run or modify.



Perhaps most importantly, Kubeflow allows for quick experimentation with data and modeling.

# View all current and past runs in one place



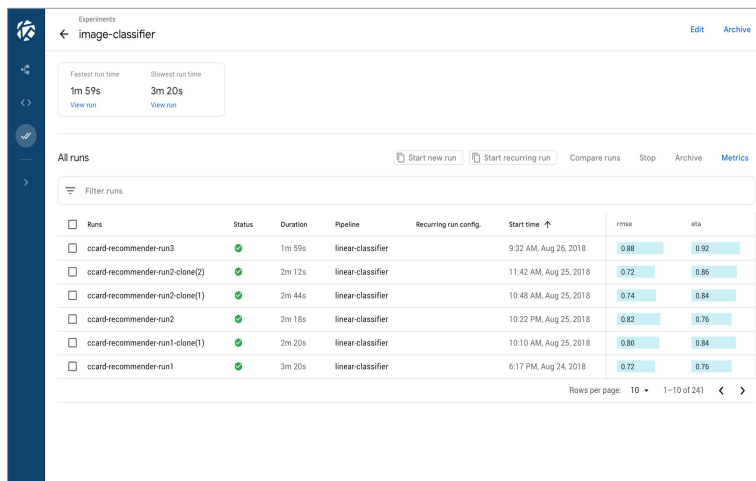
The screenshot displays the 'Experiments' section of the Kubeflow User Interface. It features a sidebar with navigation icons and a main content area with a table of experiment runs. The table has columns for 'Experiment name', 'Last 5 runs', 'Created on', and 'Created by'. The 'Last 5 runs' column uses colored circles to indicate the status of individual runs (green for success, red for failure). The table lists several experiments, including 'tfma-experiment', 'xgboost-train', 'promo-email', 'data-prep', 'tf-preprocessing', 'tf-training', and 'tf-serving'. At the bottom right of the table, there is a pagination control showing 'Rows per page: 10' and '1-10 of 241'.

<input type="checkbox"/>	Experiment name	Last 5 runs	Created on ↑	Created by
<input type="checkbox"/>	tfma-experiment	🟢	6:17 PM, Aug 24, 2018	John Doe
<input type="checkbox"/>	xgboost-train	🟢🟢🟢	6:17 PM, Aug 24, 2018	John Doe
<input type="checkbox"/>	promo-email	🟢🟢🟢🟢🔴	6:17 PM, Aug 24, 2018	Walter Fisher
<input type="checkbox"/>	data-prep	🟢	6:17 PM, Aug 24, 2018	Walter Fisher
<input type="checkbox"/>	tf-preprocessing	🟢🟢🟢	6:17 PM, Aug 24, 2018	John Doe
<input type="checkbox"/>	tf-training	🟢🟢🟢🟢🔴	6:17 PM, Aug 24, 2018	Walter Fisher
<input type="checkbox"/>	tf-serving	🟢🟢🟢	6:17 PM, Aug 24, 2018	Walter Fisher

Rows per page: 10 1-10 of 241

The Kubeflow User Interface provides an easy to explore history of all runs.

# Easy comparison and analysis of runs



Experiments

← image-classifier [Edit](#) [Archive](#)

Fastest run time: 1m 59s [View run](#) Slowest run time: 3m 20s [View run](#)

All runs [Start new run](#) [Start recurring run](#) [Compare runs](#) [Stop](#) [Archive](#) [Metrics](#)

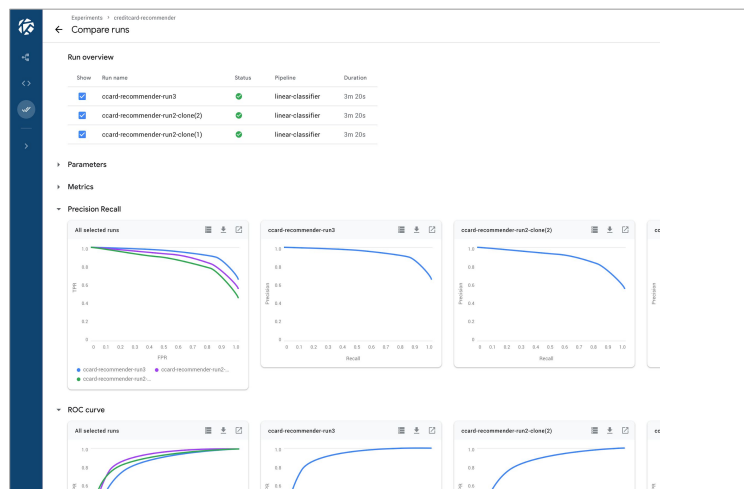
Filter runs

<input type="checkbox"/> Runs	Status	Duration	Pipeline	Recurring run config	Start time ↑	rmae	eta
<input type="checkbox"/> cc-card-recommender-run3	Completed	1m 59s	linear-classifier		9:32 AM, Aug 26, 2018	0.88	0.92
<input type="checkbox"/> cc-card-recommender-run2-clone(2)	Completed	2m 12s	linear-classifier		11:42 AM, Aug 25, 2018	0.72	0.86
<input type="checkbox"/> cc-card-recommender-run2-clone(1)	Completed	2m 44s	linear-classifier		10:48 AM, Aug 25, 2018	0.74	0.84
<input type="checkbox"/> cc-card-recommender-run2	Completed	2m 18s	linear-classifier		10:22 PM, Aug 25, 2018	0.82	0.76
<input type="checkbox"/> cc-card-recommender-run1-clone(1)	Completed	2m 20s	linear-classifier		10:10 AM, Aug 25, 2018	0.80	0.84
<input type="checkbox"/> cc-card-recommender-run1	Completed	3m 20s	linear-classifier		6:17 PM, Aug 24, 2018	0.72	0.76

Rows per page: 10 • 1–10 of 241 < >

And in addition you can quickly compare the results and processing values associated with different runs.

# Easy comparison and analysis of runs



Kubeflow makes it easy to run a number of ML experiments at the same time. For example, if you're doing hyperparameter optimization, you can easily deploy a number of different training instances with different hyperparameter sets. Kubeflow's run overview makes it easy to hone in on the techniques or parameters generating the best results. You can quickly identify what worked and what did not work.

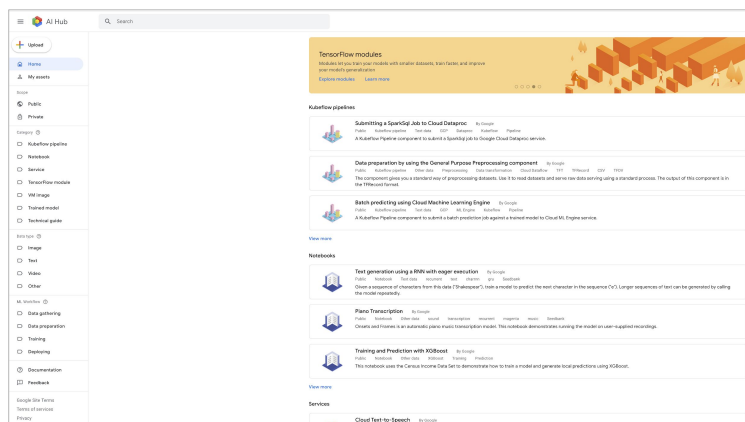


## AI Hub

We mentioned that Kubeflow pipelines can be packaged and shared with other users. This leads us to a discussion of AI Hub.

# AI Hub is a repository for AI assets

Don't reinvent the wheel! Find and deploy ML pipelines.



AI Hub is a repository for ML components. Don't reinvent the wheel! Avoid building some component when someone else has already built it, and most likely, has already optimized it. You can find and deploy not just containerized applications for machine learning, but full ML pipelines on AI Hub.



## AI Hub stores various asset types

- Kubeflow pipelines and components
- Jupyter notebooks
- TensorFlow modules
- Trained models
- Services
- VM images

What asset types can we find on AI Hub? Among the assets stored on AI Hub are entire Kubeflow pipelines, Jupyter notebooks, TensorFlow modules, fully trained models, services, and VM images.

# This is what a typical asset looks like

The screenshot displays the Google Cloud AI Hub interface for a pipeline asset titled "Deploying a trained model to Cloud Machine Learning Engine". The interface includes a sidebar with filters for Scope (Public), Version (1), Category (Kubeflow pipeline), Publisher (Google), Data type (Text), and Labels (GCP, ML Engine, Kubeflow, Pipeline). The main content area provides documentation, including a description, intended use, and runtime arguments table. A "Use this asset" section on the right features a "Download" button, which is highlighted by a blue arrow pointing to a callout box. The callout box contains the text: "One-click deployment of ML pipelines via Kubeflow on Google Cloud as platform for AI, or on premise."

**Documentation**

**Deploying a trained model to Cloud Machine Learning Engine**  
A Kubeflow Pipeline component to deploy a trained model from a Cloud Storage path to a Cloud Machine Learning Engine service.

**Intended use**  
Use the component to deploy a trained model to Cloud Machine Learning Engine service. The deployed model can serve online or batch predictions in a KFP pipeline.

**Runtime arguments:**

Name	Description	Type	Optional	Default
model_uri	The Cloud Storage URI which contains a model file. The commonly used TF model search path (export/exporter) will be used.	GCSPath	No	
project_id	The ID of the parent project of the serving model.	GCPProjectID	No	
model_id	The user-specified name of the model. If it is not provided, the operation uses a random name.	String	Yes	
version_id	The user-specified name of the version. If it is not provided, the operation uses a random name.	String	Yes	
runtime_version	The Cloud Machine Learning Engine's runtime version to use for this deployment. If it is not set, the Cloud ML Engine uses the default stable version, 1.0.	String	Yes	
python_version	The version of Python used in the prediction. If it is not set, the default version is 2.7. Python 3.5 is available when the runtime_version is set to 1.4 and above. Python 2.7 works with all supported runtime versions.	String	Yes	
version	The JSON payload of the new Version.	Dict	Yes	
replace_existing_version	A Boolean flag indicates whether to replace existing version in case of conflict.	Bool	Yes	False
set_default	A Boolean flag indicates whether to set the new version as default version in the model.	Bool	Yes	False
wait_interval	A time-interval to wait for in case the operation has a long run time.	Integer	Yes	30

**Output:**

**Use this asset**  
Download  
Create a Kubeflow Cluster to use this pipeline  
Learn more about how to use pipelines  
Feedback  
Twitter Facebook LinkedIn

Google Cloud

Here you see what a typical asset looks like. You can see information about the pipeline, such as inputs and outputs, and download options.

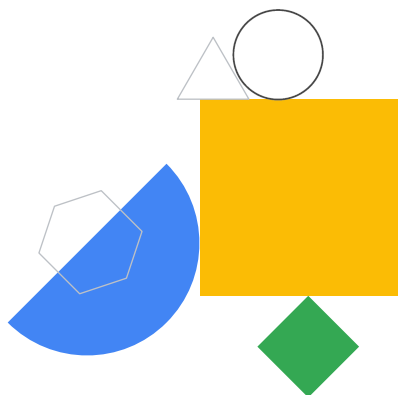
## Assets on AI Hub are collected in two scopes: public assets and restricted assets

- Public scope are available to all AI Hub users.
- Restricted scope contains AI components that you have uploaded and assets that have been shared with you.

The assets on AI Hub are collected into two scopes: public assets and restricted assets. Public assets are available to all AI Hub users. Restricted scope assets contain AI components you have uploaded and those that have been shared with you. For example, you could have assets only available to people within your organization or teams.

# Lab Intro

Running ML Pipelines on Kubeflow



To get a better understanding of how Kubeflow works, let's dive into a lab. In this lab, you learn how to install and use Kubeflow pipelines. Once Kubeflow pipelines are installed, you create and run an experimental end-to-end ML pipeline. When the pipeline is complete, you may examine the pipeline graph, metrics, logs, and parameters.

## Summary

- Use ML on Google Cloud using either:
  - Vertex AI (your model, your data)
  - AutoML (our models, your data)
- Use Kubeflow to deploy end-to-end ML pipelines.
- Don't reinvent the wheel for your ML pipeline!  
Leverage pipelines on AI Hub.

### To summarize:

- Google Cloud has several options to suit your machine-learning needs. Depending on the time and resources you have available, you have the option to use Vertex AI or AutoML.
- You can use Kubeflow to deploy end-to-end ML pipelines
- And remember don't reinvent the wheel for your ML pipeline, leverage pipelines on AI Hub.